Resource-based Corruptions and the Combinatorics of Hidden Diversity

Juan Garay
AT&T Labs – Research
Florham Park, NJ
garay@research.att.com

David Johnson
AT&T Labs – Research
Florham Park, NJ
dsj@research.att.com

Aggelos Kiayias
National and Kapodistrian
University of Athens
Athens, Greece
aggelos@kiayias.com

Moti Yung
Google Inc. and Columbia
University
New York, NY
moti@cs.columbia.edu

ABSTRACT

In the setting of cryptographic protocols, the corruption of a party has traditionally been viewed as a simple, uniform and atomic operation, where the adversary decides to get control over a party and this party immediately gets corrupted. In this paper, motivated by the fact that different players may require different resources to get corrupted, we put forth the notion of *resource-based corruptions*, where the adversary must invest some resources in order to corrupt a player.

If the adversary has full information about the system configuration then resource-based corruptions would provide no fundamental difference from the standard corruption model. However, in a resource "anonymous" setting, in the sense that such configuration is hidden from the adversary, much is to be gained in terms of efficiency and security.

We showcase the power of such *hidden diversity* in the context of secure multiparty computation (MPC) with resource-based corruptions and prove that anonymity it can effectively be used to circumvent known impossibility results. Specifically, if OPT is the corruption budget that violates the completeness of MPC (the case when half or more of the players are corrupted), we show that if hidden diversity is available, the completeness of MPC can be made to hold against an adversary with as much as a $B \cdot OPT$ budget, for any constant $B > 1$. This result requires a suitable choice of parameters (in terms of number of players and their hardness to corrupt), which we provide and further prove other tight variants of the result when the said choice is not available. Regarding efficiency gains, we show that hidden diversity can be used to force the corruption threshold to drop from $1/2$ to $1/3$, in turn allowing the use of much more efficient (information-theoretic) MPC protocols.

ITCS'13, January 9–12, 2012, Berkeley, California, USA.

We achieve the above through a series of technical contributions:

- The modeling of the corruption process in the setting of cryptographic protocols through *corruption oracles* as well as the introduction of a notion of reduction to relate such oracles;
- the abstraction of the corruption game as a combinatorial problem and its analysis; and, importantly,
- the formulation of the notion of *inversion effort preserving* (IEP) functions which is a type of direct-sum property, and the property of *hardness indistinguishability*. While hardness indistinguishability enables the dissociation of parties' identities and the resources needed to corrupt them, IEP enables the discretization of adversarial work into corruption tokens,

all of which may be of independent interest.

Categories and Subject Descriptors

C.2.0 [**COMPUTER-COMM. NETWORKS**]: Security and Protection; E.3 [**DATA ENCRYPTION**]: Public key cryptosystems; G.2.1 [**DISCRETE MATHEMATICS**]: Combinatorics—*Combinatorial algorithms*

Keywords

Cost of corruption, secure multi-party computation, combinatorial analysis, exact hardness, hardness amplification.

1. INTRODUCTION

The notion of computing in the presence of an adversary which controls or gets access to parts of the system is at the heart of modern cryptography. This paradigm has given rise to cryptosystems and protocols which achieve general tasks in the presence of powerful adversaries. A prime example of the paradigm are "completeness theorems" which show that a distributed cryptographic protocol exists where an adversary controlling any minority of the parties, cannot prevent the secure computation of any efficient functionality defined over their inputs [22]. Similar secure multiparty computation (MPC) results hold over secure channels (and no additional cryptography) with an adversary controlling

less than a third of the parties [5, 9]. Furthermore, these results are tight in the sense that when the adversary controls more parties there are functionalities that cannot be securely implemented.

In the works thus far however, the corruption of a party has been viewed as a simple, uniform and atomic operation where the adversary decides to corrupt a party and this party gets corrupted. The only limit the adversary has is on the number of individual party's corruptions and restrictions on when and how it can corrupt.

In this paper we are motivated by the fact that different players may require different resources to get corrupted. Indeed, the price paid for penetrating one organization, measured for example by the amount of money that a corrupt employee might have demanded, may differ from the cost of getting into another organization. Another example of a more cryptographic nature is when two organizations employ different password policies, or utilize VPN's relying on cryptosystems of different strengths. Taking such real world considerations into account gives rise to the notion we put forth of *resource-based corruption*, where the adversary must invest certain resources in order to corrupt a party. The initiation of the study of corruption models which describe situations like the above is our first contribution.

In a setting where the adversary has full information about a system, resource-based corruptions provide no fundamental divergence from the logic of the standard corruption model. The adversary, given an initial corruption budget, will target the largest subset of the system that it can afford and if such subset if large enough, the disruption of various security properties will ensue (as it follows from, e.g., [10]). The interesting case thus arises when the parties act on the system "anonymously" from the point of view of the adversary, in the sense that the association of parties' names and the individual corruption resources they require remains hidden. In this way, it seems plausible that a portion of the corruption budget will have to be wasted to learn the system configuration. Our second contribution is thus the investigation of the quantitative effect of this type of hidden diversity in the context of resource-based corruptions.

The problem the adversary faces can be abstracted as the following combinatorial game. The adversary is given a number of balls ("corruption tokens") and is faced with a sequence of buckets with the objective to fill a certain fraction of them. While it knows all the bucket sizes, it does not know their correspondence to the individual buckets in the sequence. If OPT is the minimum number of balls required to fill a given percentage of the buckets had the adversary been privy to the correspondence between the buckets and their sizes, how many balls as a function of OPT would be required if such hidden diversity is provided? We analyze this setting as "the combinatorics of hidden diversity" and show a number of results concerning the initial partial knowledge given to the adversary and the strategies it can apply. The analysis leads to corruption strategies (algorithms) and corruption impossibilities (lower bounds), under various cases of partial knowledge and size parameters.

Most importantly, we prove that the "value of hidden diversity" can be *unbounded!* Specifically, for any B there are ways to choose buckets' sizes so that the required resources needed by the adversary to reach its target would be $B \cdot OPT$, i.e., an arbitrarily high inflation of the required corruption budget (the cryptographic implications of this are shown below).

To realize the abstractions made above—including the combinatorial game for corruptions—we introduce a formal framework where there is a special entity called a *corruption oracle* that mediates the adversary's corruption capability. We describe a variety of natural corruption oracles and introduce a notion of reduction between them that makes the translation between different corruption games that an adversary might choose feasible. In particular this allows us to capture and quantify the setting where corruptions occur due to the inversion of computationally hard problem instances (what we call "computational corruptions").

Our third contribution is the study of the sufficient conditions under which such computational corruptions can be abstracted as token-based corruptions. First, we formulate the notion of *Inversion Effort Preserving* (IEP) functions, which, at a high level, postulates a complexity lower bound on the problem of solving (inverting) multiple instances simultaneously as a function of the sum of the complexities of solving those instances individually. This notion is a type of generalized direct-sum property. Such direct-sum properties have been studied in other contexts such as communication complexity and complexity of quadratic forms (see, e.g., [16, 27, 15]) and also relate to the notion of hardness amplification [38, 25]. We show that the IEP property holds for random functions as well as exponentiation maps in idealized models such as the random oracle model [3] and the generic group model (e.g., [37]), respectively; we also provide evidence that it holds in the computational model under complexity of factoring assumptions. Second, we formulate the notion of *hardness indistinguishability* which, essentially, expresses the inability of the adversary to distinguish between two or more inversion instances of different complexity. We express this property in the statistical sense, something that facilitates our reductions between corruption models. Effectively, the former notion (IEP) enables the discretization and token abstraction of the computational effort against a sequence of problems, while the latter (indistinguishability) provides the uncertainty the adversary faces while deciding its corruption budget allocation. In turn, these two notions rely on the hardness of the inversion problem of a function when measured in an *exact* sense, a notion we also formalize, investigate and relate to past notions.

Putting everything together, we show how hidden diversity in the context of resource-based corruptions enables us to get around impossibility results. Going back to our motivating example of secure multiparty computation, the impossibility result states that when the adversary has corruption resources sufficient to control half the players, many functionalities have to be given up. Specifically we prove the following (informally stated):

Let OPT be the optimal corruption budget for which the completeness of MPC is violated. Assuming hidden diversity, there exist configurations such that:

- For any B, the completeness of MPC holds against any adversary with less than $B \cdot OPT$ corruption budget assuming a sufficient number n of players (where $n = \Omega(\log(\frac{1}{\epsilon}) \cdot B)$, and ϵ is the probability of error).
- The above assumes the hardness of individual corruptions is not bounded. If, on the other hand, a bound M is imposed, the completeness of MPC holds against any

adversary with less than $\sim (\frac{\sqrt{M}}{\log(\frac{1}{\epsilon})}) \cdot OPT$ corruption budget assuming $n \geq \sqrt{M}$.

These results are expressed formally in Theorems 3.3 and 3.6, respectively. In addition, we provide evidence that the above results are essentially tight. For example, for the second formulation above, when the adversary's budget reaches an amount $\sim \sqrt{M} \cdot OPT$, there is a strategy that always corrupts half the players (Corollary 3.5).

Another way to exploit hidden diversity that we consider is to improve the efficiency of the implementation of MPC (as opposed to increasing the corruption budget the protocol can tolerate). We prove that, assuming the same OPT corruption budget, hidden diversity can force the corruption threshold to drop from 1/2 to 1/3, in turn allowing the use of much more efficient (information-theoretic) MPC protocols (see Theorem 3.7).

The above results demonstrate that there exist settings of player diversity that can be quite beneficial from a security standpoint if the diversity is appropriately hidden. This opens up the possibility for further investigation of the benefits of hidden diversity in various settings where a specific player configuration is assumed (or even can be imposed). We emphasize that our results are relevant in the a setting where (at least some) party corruptions require effort from the adversary; this captures a wide range of attacks, from cryptographic- (e.g., an offline dictionary attack against a password file) to system-based (e.g., a brute force search against address space layout randomization). Evidently, the relevance of our corruption model is conditional to particular system setups (but this is also the case for the setting where corruptions happen "for free" as well as any other conceivable corruption model).

Related work. To our knowledge, this is the first time that hidden diversity in terms of corruption effort is identified and used as a mechanism to boost security guarantees. This can be juxtaposed with results in the theory of cryptography that showed how other types of adversarial uncertainty can be beneficial, perhaps the closest one of which is (sender) anonymity. For example, it has been shown that variants of an anonymous channel can be used to implement unconditionally secure point-to-point channels and a broadcast channel [17, 26], as well as more efficient natural secure computation tasks, such as private information retrieval (PIR) [26]. It is worth noting that assuming sender-anonymity by itself does not fundamentally change the feasibility barriers of secure protocol design, while hidden diversity can circumvent the MPC impossibility results in the resource-based corruption setting.

With respect to resource-based corruptions, the notion of general adversary structures [24], where the adversary can choose different sets of players to corrupt from a collection of possible choices, not necessarily determined by their size, can be cast as an instance of resource-based corruption for an appropriate adversary budget and players' corruption thresholds. The two models, in fact, can be shown to be equivalent; the resource-based corruption model, however, is the one that enables us to reason about hidden diversity which is our main focus.

There is also related work regarding the "lower-bounding" of adversarial effort, including the notion of "moderately hard functions" to fight spam [14], cryptographic key escrow [36, 1], time-lock puzzles and timed-release cryptography (e.g., [33, 6, 20]), and resource fairness [19]. We note that in most such applications, however, the common theme is relating work to time, and thus crafting problems whose solution is hard to parallelize, which is not our concern here. In terms of measuring adversarial effort, our approach is along the same lines as *precise* zero-knowledge [30] where the knowledge gained by a player is measured in terms of its actual computation.

In an independent development, Bellare, Ristenpart and Tessaro [2] recently introduced the notion of *multi-instance* (MI) security, which relates to the notion of IEP functions introduced here. In settings where it is computationally feasible for instances to be compromised ("inverted," in our parlance), such as password-based cryptography, the application where the notion is showcased, it is shown that security can be amplified linearly in the number of instances in the random oracle (RO) model under a proper modeling. Such modeling, called "left-or-right xor" (LORX), is introduced in [2], aiming to capture the level of multi-instance security of a cryptographic primitive (encryption in that paper) with respect to indistinguishability-type challenges. In contrast, and in line with our objectives, the IEP property captures the multi-instance hardness behavior of one-way functions, and, as opposed to LORX, is cast as an intrinsic complexity characteristic that expresses the rate of hardness growth as a function of the number of instances.

As we already mentioned above, the notion of IEP functions also bears some resemblance to hardness amplification and direct-product theorems [38, 25]; we elaborate in more detail on such relation in the full version of the paper [18]. Finally, similar notions to the notion of exact hardness that we put forth in this paper have been considered in the literature; see Section 4.1 and [18] for a comparison to another "more refined than asymptotic" measure of inversion difficulty considered by Haitner, Harnik and Reingold [23].

Organization of the paper. The rest of the paper is organized as follows. In Section 2 we introduce the notion of resource-based corruptions, corruption oracles and reductions between them. In Section 3, we present the combinatorial analysis that allows us to prove the gains of hidden diversity at an abstract, "token" level. Finally, in Section 4 we give the definitions of exact hardness, IEP and hardness indistinguishability, which enable the application of the results to the computational-corruption setting. Candidate instantiations of the Section 4 definitions are presented in Appendix A. Some of the proofs, comparisons to related notions, as well as additional analysis and constructions, are presented in the full version of the paper [18]. We first introduce some basic notation.

Notation. We use $\lambda \in \mathbb{N}$ to denote the security parameter. All quantities are assumed to be functions of λ unless otherwise noted. Let X, Y be random variables with range in $\{0,1\}$. We write $X \overset{\epsilon}{\approx} Y$ if we have that $|\mathbf{Pr}[X=1] - \mathbf{Pr}[Y=1]| < \epsilon$.

2. RESOURCE-BASED CORRUPTIONS

A main goal of this paper is to formulate the cost for an adversary to "corrupt" parties running a cryptographic protocol, in contrast to the traditional approach where the adversary gains control of parties for free. We will be modelling this process in what nowadays is the widely accepted

method to formulate the security of a protocol carrying out a given task: the "trusted-party paradigm" [22]. Recall that in this paradigm, the protocol execution is compared with an ideal process where the outputs are computed by a trusted party that sees all the inputs. A protocol is then said to securely carry out a given task if running the protocol with a realistic adversary amounts to "emulating" the ideal process with the appropriate trusted party. In the perhaps most developed version of the paradigm, due to Canetti [7], the task of distinguishing between the two experiments is assigned to an entity (a polynomial-time interactive Turing machine [ITM]) called *the environment*, and denoted by $\mathcal{Z}$, which is also in charge of producing the inputs for and receiving the outputs from both executions. Besides $\mathcal{Z}$, the other basic entities (also ITMs) involved in the real-world execution are n players $P_1, ..., P_n$, and an adversary $\mathcal{A}$. We will be using a variant of this formulation, similar to [19], meant to capture synchronous communication and exact running-time bounds[1].

2.1 Corruption Oracles and Security Definition

We model the corruption process, under different costs, by the addition of a new entity (ITM) to the real-world execution, which we call the *corruption oracle* ($\mathcal{C}$), and whose essential purpose is to interact with adversary $\mathcal{A}$ and manage its corruption capability. In more detail, the two basic principles in $\mathcal{C}$'s operation are as follows:

- $\mathcal{C}$ is initialized with a random tape and the number of players $n \in \mathbb{N}$ involved in the system. It may return some information about the players to the adversary $\mathcal{A}$.
- For each player P_i, $\mathcal{A}$ may engage $\mathcal{C}$ in a simple (corruption) protocol, call it ρ_i, to determine whether player P_i gets corrupted. The adversary is free to schedule many such corruption protocols concurrently (either statically for a static adversary or dynamically for an adaptive one). If the protocol terminates with $\mathcal{C}$ accepting, then player P_i is declared corrupted, meaning that $\mathcal{A}$ has access to P_i's internal state, and is able to impersonate P_i in all of its subsequent interactions (for a malicious $\mathcal{A}$).

We consider communication with the corruption oracle as part of the adversary's basic step of (significant) computation (see Section 4.1). For succinctness, sometimes we will refer to the adversary $\mathcal{A}$ interacting with corruption oracle $\mathcal{C}$ as $\mathcal{A}^{\mathcal{C}}$. Following [7], we will use the notation $\text{EXEC}_{\pi,\mathcal{A}^{\mathcal{C}},\mathcal{Z}}$ to denote the (binary) distribution ensemble describing $\mathcal{Z}$'s output after interacting with adversary $\mathcal{A}$ running with corruption oracle $\mathcal{C}$, and players running protocol π.

In the ideal process we will model the corruption capability as follows. We consider a "wrapper" functionality $\mathcal{W}^{\mathcal{C}}(\mathcal{F})$ that for any functionality $\mathcal{F}$ and corruption oracle $\mathcal{C}$, "traps" the corruption requests directed to a party participating in it; all other messages and requests it immediately passes to the functionality. This will allow us to provide a corruption interface that is compatible with the corruption-oracle modeling of the real world. In more detail, for each party P_i involved in $\mathcal{F}$ the wrapper $\mathcal{W}^{\mathcal{C}}(\mathcal{F})$ initializes the corruption oracle and considers all parties as "uncorrupted." Mimicking the behavior of the real-world adversary attacking parties, $\mathcal{S}$ can submit corruption messages for party P_i following the protocol ρ_i to $\mathcal{W}^{\mathcal{C}}(\mathcal{F})$, who maintains the state of the corruption oracle. If, at some point, $\mathcal{C}$ terminates the execution of one of the ρ_i protocols and accepts, then $\mathcal{W}^{\mathcal{C}}(\mathcal{F})$ sets the party P_i status to "corrupted," issues a (standard) corruption message for P_i, $(\text{CORRUPT}, P_i)$, to $\mathcal{F}$, and returns the functionality's response, namely, the party's internal state as kept by the functionality, back to $\mathcal{S}$. In addition, we note that once a party is corrupted, the ideal adversary is typically allowed to "re-write" the corrupted party's internal state; $\mathcal{W}^{\mathcal{C}}(\cdot)$ ignores such messages while the party is uncorrupted. Let $\text{EXEC}_{\mathcal{W}^{\mathcal{C}}(\mathcal{F}),\mathcal{S},\mathcal{Z}}$ denote the (binary) distribution ensemble describing $\mathcal{Z}$'s output after interacting with adversary $\mathcal{S}$ and the ideal protocol for $\mathcal{W}^{\mathcal{C}}(\mathcal{F})$ as specified above.

We now proceed to more formally specify how the two executions—real and ideal—would be indistinguishable to the environment. The typical order of quantifiers in simulation-based security definitions ($\forall\mathcal{A}\exists\mathcal{S}\forall\mathcal{Z}$) allows the ideal-world adversary to depend on the real-world adversary that it simulates, but it should be independent of the environment. Following [19], we use a slight weakening of this definition, which is appropriate for the setting of adversaries restricted to performing a specific number of steps of the computation. Specifically, let t denote the number of steps taken by the adversary, for t a monotonically increasing arithmetic function; we consider the "compound" ITM $\langle\mathcal{Z},\mathcal{A}\rangle$ (namely, two ITMs, interacting with each other and possibly with other ITMs in a well-defined manner, treated as a single entity) as being t-bounded. (This refinement will become useful when capturing corruptions as computational effort—cf. Section 4.)

DEFINITION 2.1. *For a given t, a protocol π is said to* securely realize a functionality $\mathcal{F}$ against t-bounded adversaries and corruption oracle $\mathcal{C}$, *if for all $\mathcal{A}$ there exists an ideal-world adversary $\mathcal{S}$, running in time $t+p$, such that for all $\mathcal{Z}$ with $\langle\mathcal{Z},\mathcal{A}\rangle$ being t-bounded,*

$$\text{EXEC}_{\pi,\mathcal{A}^{\mathcal{C}},\mathcal{Z}} \overset{\epsilon}{\approx} \text{EXEC}_{\mathcal{W}^{\mathcal{C}}(\mathcal{F}),\mathcal{S},\mathcal{Z}},$$

where ϵ is some negligible function and p some polynomial.

Next, we introduce the notion of *safety* for a corruption oracle.

DEFINITION 2.2. *For a given t, we say a corruption oracle $\mathcal{C}$ is* t-safe *if for all functionalities $\mathcal{F}$ (including those that guarantee fairness and output delivery to all honest parties), there is a protocol π that securely realizes $\mathcal{F}$ against t-bounded adversaries and $\mathcal{C}$.*

If a corruption oracle is t-safe for all t, then we call it simply safe. For example, it can be derived from [22, 8] that the standard (traditional) corruption oracle allowing less than $1/2$ of the players to be corrupted (to be denoted below as $\mathcal{C}^{\text{std}}_{\frac{1}{2}}$) is safe. Further, in case we want to be explicit about the parameters involved in the proof of safety, we will say that a corruption oracle $\mathcal{C}$ is (t,γ)-safe if it holds that the above definition is satisfied with a simulation that may fail with probability at most $1-\gamma$.

2.2 Cost Measures and Relations between Corruption Oracles

Next, we will start by capturing two cost measures for corruptions, namely, the traditional one (i.e., no cost) and the

[1]We note that these formulations are in fact more powerful than needed for this paper, as we will be focusing on stand-alone protocol execution.

"token" based (i.e., discretized) measure, defining corruption oracles for them, which will already enable us to establish the power of hidden diversity at an abstract level; later on (Section 4) we will provide the final "translation" by presenting the computational-based measure. In addition, we will define the "blinded" version of such oracles (which, looking forward, will effectively model the hardness indistinguishability of functions put forth in Section 4.1), and establish the relations between them.

Standard cryptographic corruption. In this case the corruption oracle is initialized with a threshold $\alpha \in (0,1)$, and maintains a counter ctr initially set to 0. The oracle provides the number of players n to the adversary. The corruption protocol ρ_i consists of a single message $(\text{CORRUPT}, P_i)$ that is to be transmitted by the adversary to the oracle. Given such a message, the oracle checks whether $\mathsf{ctr}+1 < \lceil \alpha \cdot n \rceil$, and in this case declares player P_i corrupted (with the effect described above) and increases ctr by 1. We denote this corruption oracle by $\mathcal{C}_\alpha^{\mathsf{std}}$. For secure multi-party computation applications, typical values of α are $\frac{1}{2}$ (i.e., honest majority), $\frac{1}{3}$ and $1-\frac{1}{n}$ (i.e., at least one honest player).

Token-based corruption. In this case the corruption oracle is initialized with a vector $\overline{s} \in \mathbb{N}^n$ ($\overline{s}$ for bucket "sizes") and a threshold k (total number of tokens); the threshold of player P_i is the value s_i. The oracle gives to $\mathcal{A}$ the vector $\overline{s}$, and maintains counters $\mathsf{ctr}_1, \ldots, \mathsf{ctr}_n$ initially set to 0. Protocol ρ_i here consists of messages of the form $(\text{CORRUPT}, P_i, v)$ sent by $\mathcal{A}$. The oracle checks that $v + \sum_i \mathsf{ctr}_i \leq k$, and in this case it increases counter ctr_i by v; if it happens that $\mathsf{ctr}_i \geq s_i$, then player P_i gets corrupted. We will denote the token-based corruption oracle by $\mathcal{C}_{\overline{s},k}^{\mathsf{tk}}$.

Blinded token-based corruption. This corruption oracle, denoted by $\mathcal{C}_{\overline{s},k}^{\mathsf{btk}}$, is identical to $\mathcal{C}_{\overline{s},k}^{\mathsf{tk}}$ in operation with the following difference: whenever the adversary submits a $(\text{CORRUPT}, P_i, v)$ message, the oracle performs the update operations on player $P_{p(i)}$ where p is a secret random permutation that is selected initially and maintained by the corruption oracle. Otherwise, the blinded token-based corruption oracle behaves as the token-based corruption oracle. Given that this oracle has a private state (permutation p), for technical reasons (to be revealed later) we will also consider a "leaky" version of this corruption oracle that is parameterized by an ITM L and operates on the private permutation p. In this leaky version, the adversary $\mathcal{A}$ may submit a special request given which the corruption oracle will run L on its internal state and return its output to $\mathcal{A}$. We will denote the leaky version of this corruption oracle by $\mathcal{C}_{\overline{s},k}^{\mathsf{btk},L}$.

We now set out to study the relations between corruption oracles defined above. Our main tool is the following definition.

DEFINITION 2.3. *Fix some $\epsilon > 0$ and arithmetic function t. Given two corruption oracles $\mathcal{C}_1$ and C_2, we say that $\mathcal{C}_2$* dominates *$\mathcal{C}_1$ with error ϵ and complexity t, denoted $\mathcal{C}_1 \leq_\epsilon^t \mathcal{C}_2$, provided the following holds: for $n \in \mathbb{N}$ and any n-party protocol π, for any $\mathcal{A}$, there is an adversary $\mathcal{B}$, running in time $t+p$, such that for all $\mathcal{Z}$ with $\langle \mathcal{Z}, \mathcal{A} \rangle$ being t-bounded,*

$$\text{EXEC}_{\pi,\mathcal{A}^{\mathcal{C}_1},\mathcal{Z}} \overset{\epsilon}{\approx} \text{EXEC}_{\pi,\mathcal{B}^{\mathcal{C}_2},\mathcal{Z}},$$

where p is some polynomial.

The main intuition behind this definition is that if $\mathcal{C}_2$ dominates $\mathcal{C}_1$ and an adversary is given a choice between the two, it may always opt for the former. The way we will employ the above relation in our exposition is that if $\mathcal{C}_2$ dominates $\mathcal{C}_1$ and it happens that $\mathcal{C}_2$ is safe, then $\mathcal{C}_1$ is also safe. More specifically, if $\mathcal{C}_2$ is (t,γ)-safe and we have that $\mathcal{C}_1 \leq_\epsilon^t \mathcal{C}_2$, then we have that $\mathcal{C}_1$ is $(t, \gamma-\epsilon)$-safe. It is easy to see that the relation $\leq_\epsilon^t$ is reflexive and transitive and thus constitutes a preorder for any choice of the parameters t, ϵ.

Another easy observation is that $\mathcal{C}_{\gamma_1}^{\mathsf{std}} \leq_\epsilon^t \mathcal{C}_{\gamma_2}^{\mathsf{std}}$ provided that $\gamma_1 \leq \gamma_2$, independently of the values t, ϵ. This stems from the fact that the power of the adversary can only potentially increase in the corruption oracle of the right-hand side. When the relation $\leq_\epsilon^t$ holds for any function t, we will write $\leq_\epsilon$.

From token-based corruptions to standard corruptions. Let $\mathcal{A}$ be any adversary that interacts with $\mathcal{C}_{\overline{s},k}^{\mathsf{X}}$ for some $\overline{s} = (s_1, \ldots, s_n) \in \mathbb{N}^n, k \in \mathbb{N}$ and $\mathsf{X} \in \{\mathsf{tk}, \mathsf{btk}\}$. For some $\alpha \in (0,1)$ and complexity t, we denote by $\mathsf{bad}_{\mathsf{X}}^t[\overline{s}, k, \alpha]$ the supremum of the probability of the event that the number of corrupted players reaches $\lceil \alpha \cdot n \rceil$ in the execution of $M^{\mathcal{C}_{\overline{s},k}^{\mathsf{X}}}$, where M is any t-bounded ITM. We have the following :

LEMMA 2.4. *For any $n, k \in \mathbb{N}, \overline{s} \in \mathbb{N}^n, \alpha \in (0,1)$ and arithmetic function t, we have that $\mathcal{C}_{\overline{s},k}^{\mathsf{X}} \leq_\epsilon^t \mathcal{C}_\alpha^{\mathsf{std}}$, where $\epsilon \leq \mathsf{bad}_{\mathsf{X}}^t[\overline{s}, k, \alpha]$ and $\mathsf{X} \in \{\mathsf{tk}, \mathsf{btk}\}$.*

PROOF. The description of $\mathcal{B}$ is based on $\mathcal{A}$. Specifically, $\mathcal{B}$ will simulate the corruption oracle $\mathcal{C}_{\overline{s},k}^{\mathsf{X}}$ for $\mathcal{A}$ and whenever a player is corrupted according to the corruption oracle it will pass the corresponding corruption message to its own corruption oracle $\mathcal{C}_\alpha^{\mathsf{std}}$. The only problem that may occur in the simulation is when the corruption oracle of $\mathcal{B}$ rejects its corruption request for a certain player while the corresponding player is expected to be corrupted by $\mathcal{A}$. Using the fact that $\mathbf{Pr}[X|\neg B] = \mathbf{Pr}[Y|\neg B]$ implies that $|\mathbf{Pr}[X] - \mathbf{Pr}[Y]| \leq \mathbf{Pr}[B]$ for any three events X, Y, B over the same probability space yields the proof, by taking X to be the output of $\mathcal{A}$'s execution, Y to be the output of $\mathcal{B}$'s execution, and B the probability that $\mathcal{A}$ is able to successfully corrupt a player while $\mathcal{B}$ is denied. □

This lemma would be most useful in case ϵ is negligible (cf. next section).

3. THE COMBINATORICS OF HIDDEN DIVERSITY

In this section we will use the relations we established between the corruption oracles in the previous section in order to derive cryptographic safety bounds at a purely combinatorial level. In particular we will provide bounds for the "bad" event (Lemma 2.4) and negative results—for the adversary—showing how the blinded version of corruption oracles remains safe for ranges of parameters that are unsafe in the regular case, hence demanding from the adversary a substantially higher "budget" to achieve its goal.

We defined two types of token-based corruption oracles, regular and blinded, which are specified by two parameters: $\overline{s}, k$. Let $OPT_\alpha(\overline{s})$ be the minimal number of tokens that need to be invested in order to corrupt a set of players of size at least $\lceil \alpha n \rceil$ in the token-based corruption oracle

model, i.e., $OPT_\alpha(\overline{s}) = \min\{k : \exists C \subseteq \{1,\ldots,n\}$ with $|C| \geq \lceil \alpha n \rceil$ and $\sum_{i\in C} s_i = k\}$. Based on this definition, Lemma 2.4, and our observation that $\mathcal{C}^{\mathsf{std}}_{1/2}$ is safe, we have the following.

THEOREM 3.1. *For any $n \geq 2$, and $\overline{s} \in \mathbb{Z}^n$, the corruption oracle $\mathcal{C}^{\mathsf{tk}}_{\overline{s},k}$ is safe for any $k < OPT_{1/2}(\overline{s})$ and unsafe for any $k \geq OPT_{1/2}(\overline{s})$.*

PROOF. First, regarding cryptographic safety, we observe that as long as $k < OPT_{\frac{1}{2}}(\overline{s})$, it holds that an honest majority of players is always guaranteed. Due to the results of [22], it follows that a protocol can be constructed for any functionality $\mathcal{F}$. On the flip side, if $k \geq OPT_{\frac{1}{2}}(\overline{s})$ this means that there exists a set of players C for which it holds that $k \geq \sum_{i\in C} s_i$ and $|C| \geq \lceil \frac{n}{2} \rceil$. It follows that by corrupting this set of players it is impossible to realize all functionalities (this follows from the impossibility result of Cleve [10]). □

Next, we demonstrate how the blinded token-based corruption oracle remains safe for ranges of parameters that are unsafe in the regular case.

Balls and buckets. The problem at hand can be rephrased as the following game:An adversary wishes to distribute balls (corresponding to corruption tokens) to n buckets, so as to fill at least $\lceil \alpha n \rceil$ of them for some given $\alpha \in (0,1)$. The sizes of the buckets are given in the form of a vector $\overline{s} \in \mathbb{Z}^n$. If the adversary has full information about the buckets, it can achieve its goal by investing $OPT_\alpha(\overline{s})$ balls. Given the characteristics of the $\mathcal{C}^{\mathsf{btk}}_{\overline{s},k}$ oracle, we are interested in the case where the adversary does not know the correspondence between buckets and $\overline{s}$ so it may have to waste a certain number of balls to reach the $\lceil \alpha n \rceil$ threshold. Specifically, we assume that there is a hidden random permutation π that re-labels the buckets so the adversary knows $\overline{s}^\pi = (s_{\pi(1)},\ldots,s_{\pi(n)})$. We note that it should be the case that $\overline{s}$ includes at least 2 different sizes, since otherwise the hidden permutation would not stall the adversary in any way. Specifically, the cardinality of the set $\{s \mid \exists i : s = s_i\}$ is bigger than 1.

3.1 Increased Security from Hidden Diversity

We first consider the case when there are no restrictions on the number of sizes or their values.

THEOREM 3.2. *For any α, $0 < \alpha < 1$, constants $B > 1$ and $\epsilon < 2^{-4}$ and for any $n \geq \log(1/\epsilon) \cdot \max\{1/\alpha, (4B-2)/(1-\alpha)\}$, there exists a vector $\overline{s} \in \mathbb{Z}^n$ such that any adversary that is given $\overline{s}^\pi$ for a random permutation π, and has fewer than $B \cdot OPT_\alpha(\overline{s})$ balls, has probability less than ϵ of filling $\lceil \alpha n \rceil$ buckets.*

PROOF. For a c, $0 < c \leq \alpha$, to be specified later, our instance will have $\lceil cn \rceil$ buckets of size $\lceil \alpha n \rceil + 1$, $\lceil \alpha n \rceil - \lceil cn \rceil$ buckets of size 1, and $n - \lceil \alpha n \rceil$ buckets of size $(\lceil cn \rceil + 2)B\lceil \alpha n \rceil$. An optimal solution will consist of the $\lceil \alpha n \rceil$ smallest buckets, and will have total size $(\lceil cn \rceil + 1)\lceil \alpha n \rceil$. This implies that we cannot afford to fill any of the largest buckets if we are to use fewer than $B \cdot OPT_\alpha(\overline{s}) = B(\lceil cn \rceil + 1)\lceil \alpha n \rceil$ balls, and any algorithm that fills $\lceil \alpha n \rceil$ buckets must find and fill all the buckets of size $\lceil \alpha n \rceil + 1$.

But now note that the only way the adversary can tell whether a bucket has this size or one of the larger ones is to place $\lceil \alpha n \rceil + 1$ balls in the bucket. Thus, if the adversary is to use no more than $B(\lceil cn \rceil + 1)\lceil \alpha n \rceil$ balls, it can test no more than $B(\lceil cn \rceil + 1)$ buckets with size exceeding 1, of which there are $n - \lceil \alpha n \rceil + \lceil cn \rceil$. The probability of finding all $\lceil cn \rceil$ of the mid-size buckets is then at most

$$\left(\begin{array}{c} n - \lceil \alpha n \rceil \\ B(\lceil cn \rceil + 1) - \lceil cn \rceil \end{array} \right) \Big/ \left(\begin{array}{c} n - \lceil \alpha n \rceil + \lceil cn \rceil \\ B(\lceil cn \rceil + 1) \end{array} \right).$$

Setting $X = n - \lceil \alpha n \rceil$ and $Y = B(\lceil cn \rceil + 1) - \lceil cn \rceil$, this is equal to

$$\left(\begin{array}{c} X \\ Y \end{array} \right) \Big/ \left(\begin{array}{c} X + \lceil cn \rceil \\ Y + \lceil cn \rceil \end{array} \right)$$

$$= \frac{(Y+1)(Y+2)\cdots(Y+\lceil cn \rceil)}{(X+1)(X+2)\cdots(X+\lceil cn \rceil)},$$

which we will be able to argue is sufficiently small if we can chose $c > 0$ such that, say, $Y + \lceil cn \rceil \leq (X + \lceil cn \rceil)/2$. This requires that $B\lceil cn \rceil + B < (n - \lceil \alpha n \rceil + \lceil cn \rceil)/2$, or $(2B-1)\lceil cn \rceil \leq n - \lceil \alpha n \rceil - 2B$. which will be true so long as $(2B-1)(cn+1) \leq n(1-\alpha) - 1 - 2B$, or $(2B-1)(cn) \leq n(1-\alpha) - 4B$, or $c \leq (1-\alpha)/(2B-1) - 4B/(n(2B-1))$. It is easy to see that our bound on n implies also that $n > 8B/(1-\alpha)$ hence all we need for $Y + \lceil cn \rceil < (X + \lceil cn \rceil)/2$ is that $c \leq (1-\alpha)/(4B-2)$. By our construction, we also need $c \leq \alpha$, so it suffices to set $c = \min\{\alpha, (1-\alpha)/(4B-2)\}$.

Given that $Y + \lceil cn \rceil < (X + \lceil cn \rceil)/2$, we then have that $(Y+i)/(X+i) \leq 1/2$ for $1 \leq i\lceil cn \rceil$, and hence the probability of success is at most

$$\frac{(Y+1)(Y+2)\cdots(Y+\lceil cn \rceil)}{(X+1)(X+2)\cdots(X+\lceil cn \rceil)} < (1/2)^{\lceil cn \rceil} \leq (1/2)^{cn}.$$

Setting $(1/2)^{cn} = \epsilon$ and solving for n yields the claimed result. □

The above theorem implies the following result.

COROLLARY 3.3. *For any $B > 1$ and $\epsilon < 2^{-4}$, and for any $n \geq \log(1/\epsilon)(8B-4)$, there exists a vector $\overline{s} \in \mathbb{Z}^n$ such that the corruption oracle $\mathcal{C}^{\mathsf{btk}}_{\overline{s},k}$ is $(1-\epsilon)$-safe provided that $k < B \cdot OPT_{1/2}(\overline{s})$.*

PROOF. We show that $\mathcal{C}^{\mathsf{btk}}_{\overline{s},k} \leq_\epsilon \mathcal{C}^{\mathsf{std}}_{n,\frac{1}{2}}$ for the choice of parameters $n, \overline{s}, k$ of the previous theorem. Cryptographic safety will follow then immediately due to Theorem 3.1.

In order to prove the reduction between the corruption oracles we need to provide an adversary $\mathcal{B}$ operating with $\mathcal{C}^{\mathsf{std}}_{n,1/2}$ for any adversary $\mathcal{A}$ operating with $\mathcal{C}^{\mathsf{btk}}_{\overline{s},k}$. $\mathcal{B}$ will simply simulate the corruption oracle of $\mathcal{A}$ and submit the corruption requests to $\mathcal{C}^{\mathsf{std}}_{n,1/2}$ whenever a corruption with $\mathcal{C}^{\mathsf{btk}}_{\overline{s},k}$ takes place. The simulator will fail with probability at most $\mathsf{bad}_{\mathsf{btk}}[\overline{s}, k, 1/2]$ from Lemma 2.4. Due to the previous theorem we can bound the probability by ϵ, which completes the proof. □

The above results require arbitrarily large bucket sizes (components of $\overline{s}$). In real applications, it is plausible that there might be some upper bound M on the maximum size. We now show how an adversary can exploit this (and other restrictions), and what level of safety may remain.

Consider the following algorithm $H2$ that is a hybrid between two simple bucket-filling strategies: one that continuously picks an empty bucket and fills it, and one that layers

balls horizontally across all buckets. Let $M' = \lfloor \sqrt{M} \rfloor$. Algorithm $H2$ proceeds in two basic steps: (1) While less than $\lceil \alpha n \rceil$ buckets are full and there is an unfull bucket containing fewer than M' balls, choose one with the fewest balls and place a ball in it. (2) While less than $\lceil \alpha n \rceil$ buckets are full, pick an unfull bucket and add balls until it is filled to capacity.

Let $H2_\alpha(\overline{s})$ denote the worst-case number of balls that $H2$ must place in order to fill $\lceil \alpha n \rceil$ buckets. The following upper bound on the behavior of $H2$ is proved in the full version of the paper ([18], Section B.3.4.)

THEOREM 3.4. *For $0 < \alpha < 1, n \in \mathbb{N}$ and any $\overline{s} \in \mathbb{Z}^n$ with maximum size M, $\frac{H2_\alpha(\overline{s})}{OPT_\alpha(\overline{s})} \leq 1 + \frac{\sqrt{M}}{\alpha}$.*

COROLLARY 3.5. *For any $n > 1$, $M > 0$, and any $\overline{s} \in \mathbb{Z}^n$ with maximum level M, the corruption oracle $\mathcal{C}^{\mathsf{btk}}_{\overline{s},k}$ is unsafe whenever $k \geq (1 + 2\sqrt{M}) \cdot OPT_{1/2}(\overline{s})$.*

PROOF. The proof follows as a direct application of the previous theorem. Specifically, we have that due to the previous theorem there is an adversarial strategy that needs at most $1 + 2\sqrt{M}$ balls to fill $\lceil \frac{1}{2} \cdot n \rceil$ buckets. It follows that with this strategy an adversary can corrupt $\lceil \frac{1}{2} \cdot n \rceil$ players always and as we argued in Theorem 3.1, it follows that the corruption oracle is unsafe. □

The above relation between the maximum bucket size and unsafety is essentially tight—once the number of corruption tokens drops below $\sqrt{M}$ by a fixed fraction there exist $\overline{s}$ with maximum size M where with high probability the adversary will fail to fill the target number of buckets. In [18] we show the following.

THEOREM 3.6. *For any $\epsilon > 0$ and any $M \geq 8$, $n > 4$ with $\lceil \sqrt{M} \rceil$ dividing n, there is an $\overline{s} \in \mathbb{Z}^n$ with maximum size M, such that the corruption oracle $\mathcal{C}^{\mathsf{btk}}_{\overline{s},k}$ is $(1 - \epsilon)$-safe provided that $k < \delta \cdot \sqrt{M} \cdot OPT_{1/2}(\overline{s})$ where $\delta = \min\big(2/45, (4/15)/\log(1/\epsilon)\big)$.*

It is worth noting that this theorem becomes trivial if $\delta < 1/\sqrt{M}$, since in that case we would be given fewer than $OPT_{1/2}(\overline{s})$ balls and so even the unblinded corruption oracle would be safe. This in turn implies that the theorem is only meaningful for $\epsilon > 2^{-(4/15)\sqrt{M}}$.

The above results about the cryptographic safety of the blinded token-based corruption oracle are all proved using systems $\overline{s}$ that have three distinct sizes. The fact that we use more than two sizes turns out to be necessary. For systems with just two sizes, we show in [18] that there is an adversarial strategy that violates cryptographic safety given just $2 \cdot OPT_{1/2}(\overline{s})$ corruption tokens; this bound turns out to be tight.

3.2 Increased Efficiency from Hidden Diversity

Next, we explore the question whether hidden diversity can be used to relax the computational effort in secure multiparty computation (MPC). Consider an adversary that is given $k < OPT_{1/2}(\overline{s})$ corruption tokens. Recall that the corruption oracle $\mathcal{C}^{\mathsf{tk}}_{\overline{s},k}$ is safe for exactly this range of adversarial resources, i.e., fully secure MPC can be achieved under computational assumptions [22, 31, 8]. The problem we investigate in this section is what potential benefits can be reaped in the hidden diversity setting assuming the same level of adversarial resources. In [18] we show the following.

THEOREM 3.7. *For any $\beta > 0$, there are constants $N > 1$ and $a < 1$, such that for any $n > N$, there is a vector $\overline{s} \in \mathbb{Z}^n$ with $\mathcal{C}^{\mathsf{btk}}_{\overline{s},k} \leq_{a^n} \mathcal{C}^{\mathsf{std}}_{1/4+\beta}$, provided that $k \leq OPT_{1/2}(\overline{s})$.*

Thus, if we choose $\beta = 1/3 - 1/4 = 1/12$ and an instance $\overline{s}$ with sufficiently large n, an adversary will have probability less than ϵ of corrupting $1/3$ or more participants, where "sufficiently large" here again grows proportionally with $\log(1/\epsilon)$. This result can be used to extend the application domain of information-theoretic protocols for fully secure MPC such as those of [5, 12], which are typically much more efficient than the cryptographic ones, but that in the regular corruption model only tolerate a rate of corruptions less than $1/3$.

The full version of the paper contains additional results in the blinded balls-and-buckets model, including adversarial strategies that bound how much the previous result can be extended and an examination of the case where the number of buckets n is bounded but not the maximum size M [18].

4. COMPUTATIONAL CORRUPTIONS

In this last section we turn to capturing and quantifying the setting where corruptions occur due to the inversion of computationally hard problem instances (what we call "computational corruptions"), and study the sufficient conditions under which such corruptions can be abstracted out as token-based corruptions. We start by defining the corresponding corruption oracles. First, the computational corruption oracle captures the setting where the adversary can corrupt a player by "breaking in," specifically by solving an instance of a computational problem that is otherwise unrelated to the cryptographic protocol (for example, this would be the case when the adversary's offline attack against a cryptographic authentication protocol succeeds). The *blinded* computational corruption oracle extends the above to the setting where the adversary cannot determine the correspondence between players and the instances that need to be solved for a break-in to occur.

Computational corruption oracle. The oracle is initialized with the description of functions $f_1, \ldots, f_n$ and samples $x_1, \ldots, x_n$ from the functions' respective domains. Subsequently, it provides to the adversary the vector $(y_1, \ldots, y_n)$, where $y_i = f_i(x_i)$. The corruption protocol ρ_i here consists of messages of the form $(\text{CORRUPT}, P_i, x)$ provided by $\mathcal{A}$; if it happens that $y_i = f_i(x)$, then player P_i gets corrupted[2]. We denote this oracle by $\mathcal{C}^{\mathsf{cc}}_{\overline{f}}$, where $\overline{f} = (f_1, \ldots, f_n)$.

Blinded computational corruption oracle. This case is similar to computational corruption with the difference that in addition, the corruption oracle is initialized with a random permutation p. The oracle provides to $\mathcal{A}$ the vector $(y_{p(1)}, \ldots, y_{p(n)})$. The corruption protocol messages are as in the case of $\mathcal{C}^{\mathsf{cc}}_{\overline{f}}$. We denote this corruption oracle by $\mathcal{C}^{\mathsf{bcc}}_{\overline{f}}$.

[2]As stated, the corruption protocol is an abstract version of a potentially more complex interaction where, for example, y_i is the public-key of player P_i and corruption takes place by getting ahold of the secret key x_i via running some algorithm against y_i and then authenticating on behalf of P_i.

We will be considering this definition in the setting when the functions $\overline{f}$ are hardness indistinguishable.

Next, we introduce the two notions that play a fundamental role in the complexity interpretation of resource-based corruptions.

4.1 Hardness Indistinguishability and Inversion Effort Preserving Functions

Both notions are related to the hardness of the inversion problem of a function when measured in an *exact* sense. We start by explaining this notion of hardness first.

4.1.1 Exact hardness

Consider any function $f : X \to Y$, where $X = \cup_{\lambda\in\mathbb{N}} X_\lambda$, $Y = \cup_{\lambda\in\mathbb{N}} Y_\lambda$. In what follows, if λ is clear from the context, we may denote X_λ and Y_λ by simply X and Y, respectively.

An inversion algorithm for f with success $p(\lambda)$ in time $t(\lambda)$, is a non-uniform algorithm $\mathcal{A}$ that for any λ, receives input $f(x)$ where x is uniformly distributed over X_λ and returns a value that belongs in $f^{-1}(f(x))$ with probability at least $p(\lambda)$ while being restricted to read at most $t(\lambda)$ symbols of its advice string and run for at most $t(\lambda)$ steps. We write $\mathcal{A}_t$ to denote $\mathcal{A}$ restricted on performing only t steps of computation and reading only t symbols of its advice string (for any λ). We then denote the success probability of $\mathcal{A}$, namely, $\mathbf{Pr}[\mathcal{A}_t(f(x)) \in f^{-1}(f(x))]$, by $p_{\mathcal{A},t}(\lambda)$. For simplicity, in the sequel we may drop λ from $p_{\mathcal{A},t}(\lambda)$.

We remark that the restriction on the number of steps may only bind a specific operation, which would be computational significant compared to the others (such as, for example, modular exponentiation). At times we may specify the input distribution to be other than the uniform distribution over X; e.g., we may consider some $X' \subseteq X$ and let x be distributed uniformly over X'. Unless otherwise noted we will assume that the uniform over X is used.

We are now ready to introduce *exact hardness*, a notion which will later on allow us to compare functions according to their inversion difficulty. In a nutshell, the exact hardness of a function, parameterized by ϵ, is the number of steps that necessarily needs to be surpassed in order to achieve probability of success at least ϵ. More formally:

Definition 4.1. *For any $\epsilon \in (0,1)$ and a function $f : X \to Y$, we define the* exact hardness of f with probability ϵ *to be the maximum $H \in \mathbb{N}$ such that for any $\mathcal{A}$ and $t \leq H$, it holds that $p_{\mathcal{A},t} < \epsilon$. For each $\lambda \in \mathbb{N}$, we denote the maximum such H by $H_{f,\epsilon}(\lambda)$.*

Notions of similar nature have been considered in the literature; a salient difference is that exact hardness fixes ϵ, and then calls for the largest possible t for which the function remains hard to invert. Also, observe that, by definition, there is a (non-uniform) algorithm $\mathcal{A}$ that performs a number of steps $t = H_{f,\epsilon}(\lambda) + 1$ with a similarly sized advice string and satisfies $p_{\mathcal{A},t}(\lambda) \geq \epsilon$ for all λ. This holds since if there was no such algorithm, then $H_{f,\epsilon}$ would not be the maximum possible for that particular value of λ.

Similar notions to the notion of exact hardness that we put forth in this paper have been considered in the literature. Notably, Nissan and Wigderson [32] define the hardness of a Boolean function as the largest size of the circuit whose bias is still bounded by the inverse of its size. In our setting, we are interested in fixing the bound of success to a certain level ϵ and maximizing with respect to that (as opposed to allowing ϵ to vary as the inverse of the circuit size). Bellare and Rogaway [4] define a function to be (t,ϵ)-secure if there is no "t-inverter" (an algorithm bounded by t in number of steps and size) with success probability at least ϵ. Put in this latter context, our notion of exact hardness can be defined by fixing ϵ and calling for the largest possible t for which the function remains (t,ϵ)-secure. Another "more refined than asymptotic" measure of inversion difficulty has been considered by Haitner, Harnik and Reingold [23]; we explore its relation to exact hardness in more detail in the full version of the paper [18].

Next, we show two basic properties of our notion of exact hardness. Specifically, we show that exact hardness is monotonically increasing in the probability of inversion success ϵ, and that there is a natural upper bound for exact hardness that stems from the fact that any function over a finite domain can be subjected to a brute-force attack.

Proposition 4.2. *Let $f : X \to Y$ be any function. For any $\lambda \in \mathbb{N}$ we have:*

1. *For any $0 < \epsilon \leq \epsilon'$, it holds that $H_{f,\epsilon}(\lambda) \leq H_{f,\epsilon'}(\lambda)$.*
2. *For any $\epsilon > 0$, $H_{f,\epsilon}(\lambda) \leq \lceil \epsilon \cdot |X_\lambda| \rceil$.*

Proof. We drop λ for notational simplicity. For the first property, assume for the sake of contradiction that $\epsilon \leq \epsilon'$ and $H_{f,\epsilon} > H_{f,\epsilon'}$. Consider now an algorithm $\mathcal{A}$ running in $t \leq H_{f,\epsilon}$ steps. It follows that it has probability of success less than $\epsilon \leq \epsilon'$. As a result any algorithm $\mathcal{A}$ running in time at most $H_{f,\epsilon}$ has probability of success less than ϵ'. This contradicts that $H_{f,\epsilon'}$ is the maximum integer with this property.

For the second property, we consider the basic step to be the operation of reading a pair (x,y) and an element y' from the respective tapes they reside in, and testing whether $y = y'$. Now let $z \geq 1$ be an integer function and consider a family of advice strings that contain pairs of the type (x_i, y_i), $i = 1,\ldots,z$, that belong to the graph of f following a lexicographic ordering. Consider now the algorithm $\mathcal{A}$ that, given y, scans the advice string, and if it finds (x_i, y_i) such that $y = y_i$ it returns x_i. It is easy to see that the number of steps that $\mathcal{A}$ takes is z (in the worst case—without loss of generality we can assume that $\mathcal{A}$ always takes that many steps). The success probability of $\mathcal{A}$ is $\frac{z}{|X|}$, since the desired output x is selected at random from all inputs.

Recall now that no algorithm running $H_{f,\epsilon}$ steps can have probability of success at least ϵ. If, for the sake of contradiction, we assume that for sufficiently large values of the security parameter it holds that $\lceil \epsilon \cdot |X| \rceil < H_{f,\epsilon}$, it follows that $H_{f,\epsilon} \geq \lceil \epsilon \cdot |X| \rceil + 1$; thus, setting $z = \lceil \epsilon \cdot |X| \rceil + 1$, we have that $\mathcal{A}$ performs $z \leq H_{f,\epsilon}$ operations and has success probability of success $\frac{z}{|X|} > \epsilon$, which is a contradiction. □

It is worth noting that in the specification of $H_{f,\epsilon}$ there is no presumption of a "minimal hardness" for the inversion problem of the function f; rather, $H_{f,\epsilon}$ is meant to capture the intrinsic property of f that corresponds to the minimum computational effort (measured in basic operations depending on the computational model) that is required to invert the function with a certain probability of success. If one assumes that this value is sufficiently high, then the function would be presumed to be one-way. To this end, we now show the basic relations between exact hardness and one-wayness. Specifically, for one-wayness we have that for any ϵ that is

bounded away from 0 by an inverse polynomial, the exact hardness of the function to beat ϵ should exceed any polynomial function. Similarly, for *weak* one-way functions, we have that there is some threshold, which is an inverse polynomial away from 1, that in order to be reached the exact hardness should exceed any polynomial function.

PROPOSITION 4.3. *Let $f : X \to Y$ be a polynomial-time computable function. Then:*

1. *f is a one-way function if and only if $\forall c_1, c_2 : \epsilon = \lambda^{-c_1}$ implies $H_{f,\epsilon}(\lambda) > \lambda^{c_2}$ for sufficiently large λ.*
2. *f is a weak one-way function if and only if $\exists c_1\ \forall c_2$: $\epsilon = 1 - \lambda^{-c_1}$ implies $H_{f,\epsilon}(\lambda) > \lambda^{c_2}$ for sufficiently large λ.*

PROOF. The proofs of the two statements are similar so we only prove the first one. For the forward direction, we assume that the function is one-way and $\exists c_1, c_2$ for which if $\epsilon = \lambda^{-c_1}$, then $H_{f,\epsilon} \leq \lambda^{c_2}$. Then we have that there is some algorithm that runs in $\lambda^{c_2} + 1$ steps (which is a polynomial in λ) and has success probability at least λ^{-c_1} for infinitely many λ. This contradicts one-wayness.

For the backward direction, consider an algorithm $\mathcal{A}$ that attempts to invert f and runs in λ^{c_2} steps. Also let λ^{c_1} be any polynomial. Now suppose that the probability of inversion success is at least λ^{-c_1} for infinitely many choices of λ (i.e., the function f fails to be one-way). This contradicts the fact that $H_{f,\epsilon} > \lambda^{c_2}$, which states that in order to reach probability of success λ^{-c_1} one has to exceed λ^{c_2} steps. □

It is natural to ask how easy is to calculate $H_{f,\epsilon}$ for a function f. Evidently, any inversion algorithm for f provides an upper bound for exact hardness, while any lower bound argument on the complexity of the inversion problem of f is a lower bound for $H_{f,\epsilon}$. While finding a formula for $H_{f,\epsilon}$ might be hard for a given function f, for certain functions in idealized computational models, such as random functions [3] or exponentiation maps in the generic group model (e.g., [37]), obtaining exact formulae is in fact possible; we provide such results in Appendix A. Furthermore, under cryptographic assumptions, reasonable ranges for $H_{f,\epsilon}$ can be stated; we do so for factoring-related assumptions also in Appendix A.

4.1.2 The IEP property

Next, we consider the setting where instead of just one, a *set* of functions are to be inverted, and would like, in particular, to have a measure for their "combined" hardness, as a function of the functions' individual hardnesses.

DEFINITION 4.4. *Let $\epsilon > 0, n \in \mathbb{N}$ and τ be a monotonically increasing function. We say a sequence of functions $\{f_i\}_{i=1,\ldots,n}$ is τ-inversion effort preserving (τ-IEP) if for any subset $S = \{i_1, \ldots, i_m\} \subseteq [n]$, it holds that $H_{f^S,\epsilon} \geq \tau(\sum_{i\in S} H_{f_i,\epsilon})$, where $f^S(x_1, \ldots, x_m) \stackrel{\text{def}}{=} \langle f_{i_1}(x_1), \ldots, f_{i_k}(x_m)\rangle$.*

Note that, trivially, $H_{f^{[n]},\epsilon} \geq \max_i H_{f_i,\epsilon}$, since any algorithm that inverts a sequence of instances will have to spend at least as much time as the time needed to invert the most difficult one. Nevertheless, it is not guaranteed that the work performed for the solution of any single instance cannot be used to speed up the solution of other instances. Essentially, the IEP property above says that the speed-up that can occur is lower bounded by a value that is determined by the sum of the individual exact-hardness functions, calibrated by a given function (τ). In the extreme case, if τ is the identity function, then the instances have to be solved entirely independently and there is no algorithm that can proceed towards solving a subset of the instances simultaneously with a joint strategy.

Our IEP property is related to the notions of direct sum and direct product in complexity-theoretic models. A direct sum result holds in a model for a problem if solving one instance x of a problem costs c, then when given k independent instances $x_1, ..., x_k$, no significant gain is achievable and the model requires about ck cost to solve all instances. A direct product result, on the other hand, holds if a fixed probability p to solve a single instance correctly is given, then the probability of solving a m-vector of instances drops exponentially with m (as in, for example, Yao's "concatenation lemma" [38]). We provide further context on this relation in [18].

The IEP property can be proven to hold in idealized models such as the random oracle model or the generic group model (we demonstrate this in Appendix A); furthermore, it is reasonable to assume that it holds for standard cryptographic functions such as multiplication of primes, provided that a suitably function τ is chosen (also Appendix A).

4.1.3 Hardness indistinguishability

To introduce this notion, we define first the notion of *indistinguishability* between two functions. At a high level, it is the realization of this property that will provide the hiding of the functions' hardness, "blinding" the adversary as to what functions to attack first.

DEFINITION 4.5. *For $\epsilon > 0$, two functions $f_1 : X_1 \to Y_1, f_2 : X_2 \to Y_2$ are statistically indistinguishable if the random variables $f_1(x_1), f_2(x_2)$ have statistical distance less than ϵ when x_i is uniformly drawn from X_i, for $i = 1, 2$.*

We observe that for all but at most an ϵ fraction of $y_2 \in f_2(X_2)$ it holds that there is some x_1 with $f_1(x_1) = y_2$. The above definition is particularly interesting to us in the setting where, say, $H_{f_1,\epsilon} < H_{f_2,\epsilon}$ for some ϵ; i.e., the functions behave differently with respect to the exact hardness of the inversion problem. In such case we will talk about *hardness indistinguishability*. Given that an instance can be solved almost always in more than one way, hardness indistinguishability ensures that the hardness level can be *equivocated*[3].

The definition extends to the case of a sequence of functions in a straightforward way. We will call a sequence of functions indistinguishable when every pair of functions is indistinguishable. We give two constructions of such functions, one generic (that requires the extension of the function's outputs) and one more efficient that depends on dense public-key cryptosystems [35]. Note that since indistinguishability is required in the statistical sense, there is no need to

[3]This property becomes handy when designing simulators for reductions between corruption models (cf. Section 4.2). We note that the reason for considering statistical as opposed to computational indistinguishability is that the former provides to the simulator the ability to "equivocate" even when the adversary is able to invert some of the functions, as we allow in our corruption model.

consider how the decision problem's computational complexity amortizes over a sequence of indistinguishable instances (as it is unattainable except with probability ϵ).

Our first construction is general and relies on the existence of *regular* one-way functions see e.g., [21].

Construction #1. Let $f_1 : X_1 \to Y_1$ and $f_2 : X_2 \to Y_2$ be any two regular functions with exact hardness $H_{f_i,\epsilon}$ for $i = 1, 2$. We define the functions f_1', f_2' as follows $f_1' : X_1 \times Y_2 \to Y_1 \times Y_2$ and $f_2' : Y_1 \times X_2 \to Y_1 \times Y_2$ so that $f_1'(x_1, y_2) = (f_1(x_1), y_2)$ and $f_2'(y_1, x_2) = (y_1, f_2(x_2))$. Observe that $H_{f_i',\epsilon} = H_{f_i,\epsilon}$ for $i = 1, 2$. Moreover the domains of f_1', f_2' are efficiently sampleable (assuming those of f_1, f_2 are). The proof of the following proposition is straightforward.

PROPOSITION 4.6. *For any regular f_1, f_2, the functions f_1', f_2' defined as above are (perfectly) hardness indistinguishable.*

The above approach generalizes easily to a sequence of functions (at the expense of increasing linearly the length of the domain and range elements).

Construction #2. A more efficient way to construct indistinguishable functions is using *dense one-way functions* along the lines of dense public-key cryptosystems of [35, 34]. Specifically, a one-way function is called dense if its output is statistically indistinguishable from a random string of a certain length. More formally, if f is the function, it holds that $f(x)$ is statistically close to $\{0,1\}^k$ for some suitable k when x is uniformly distributed. In order to show hardness indistinguishability, given a sequence of dense one-way functions, each function output can be padded with random bits so that all of them match the length of the longest one. Based on the density property, the functions modified as above are pairwise statistically indistinguishable.

We now briefly sketch how a dense one-way function can be constructed. Given a function $f : X \to Y$ (for simplicity assume it is an injection), a dense one way function can be derived by applying a strong randomness extractor that is also collision resistant (see [13]). Specifically, if Ext is such an extractor, we have that $f'(r, x) = (r, \mathsf{Ext}(r, f(x)))$ is a dense one-way function. Indeed, since Ext is a strong extractor and r is a uniformly random seed, it holds that the output of f' is almost uniformly distributed (provided Ext has suitable length). On the other hand, given the collision resistance property of Ext any inversion algorithm against f' can be turned to an algorithm inverting f, hence, f' is a one-way function of exact hardness not much less than that of f. We omit further details.

In Appendix A we study and give bounds for the exact hardness, IEP property and hardness indistinguishability of several candidate functions, namely, random functions, exponentiation maps in the generic group model, and integer multiplication.

4.2 From Computational Corruptions to Token based Corruptions

We now have the tools to consider the relation between the computational corruption oracle and the token-based corruption oracle. Connecting the two relies on whether the computation effort invested by the adversary against corrupting players can be abstracted as discrete token investments. The notion of IEP functions introduced above plays a crucial role here.

THEOREM 4.7. *Let $\epsilon > 0$ and τ a monotonically increasing invertible function. Given a τ-IEP sequence of functions $f_1, \ldots, f_n$, we have that for any t there exist $\overline{s}, k, L$ such that*

$$\mathcal{C}_{\overline{f}}^{\mathsf{cc}} \leq_{\epsilon}^{t} \mathcal{C}_{\overline{s},k}^{\mathsf{tk}} \quad \textit{and} \quad \mathcal{C}_{\overline{f}}^{\mathsf{bcc}} \leq_{\epsilon}^{t} \mathcal{C}_{\overline{s},k}^{\mathsf{btk},L},$$

where $\overline{s} = (s_1, \ldots, s_n)$ is such that $s_i = H_{f_i,\epsilon}$ for $i = 1, \ldots, n$ and $k = \lceil \tau^{-1}(t) \rceil$, and where $L(p)$ operates so that it returns $(y_{p(1)}, \ldots, y_{p(n)})$ with $y_i = f(x_i)$ and x_i a randomly selected input of f_i.

PROOF. We first consider the relation between computational corruption and token-based corruption. The adversary $\mathcal{B}$ will simulate $\mathcal{A}$ as well as the corruption oracle $\mathcal{C}_{\overline{f}}^{\mathsf{cc}}$. During the simulation, $\mathcal{B}$ operates exactly as $\mathcal{A}$ with the following modification: when $\mathcal{A}$ submits a corruption request $(\textsc{Corrupt}, P_i, x)$, $\mathcal{B}$ submits $(\textsc{Corrupt}, P_i, s_i)$ to the corruption oracle.

Corruption requests $(\textsc{Corrupt}, P_i, x)$ for which it holds that $f_i(x) \neq y_i$ are ignored by $\mathcal{B}$.

The only divergence in the simulation is when it may happen that $\mathcal{A}$ issues a corruption request that is granted while the corresponding corruption request of $\mathcal{B}$ is denied. Suppose this happens with probability at least ϵ. This means that $\mathcal{B}$ has used all its tokens $\lceil \tau^{-1}(t) \rceil$, i.e., the set of players C corrupted by $\mathcal{A}$ satisfies that $\sum_{i \in C} s_i > \lceil \tau^{-1}(t) \rceil \geq \tau^{-1}(t)$. Note that since $\mathcal{A}$ manages to corrupt the set of players C with probability at least ϵ it follows that it runs for at least $\tau(\sum_{i \in C} H_{f_i,\epsilon}) + 1$ steps due to the τ-IEP property, i.e., $t \geq \tau(\sum_{i \in C} H_{f_i,\epsilon}) + 1 > \tau(\sum_{i \in C} s_i)$, which is a contradiction.

Regarding the relation of blinded computational corruption and leaky blinded token-based corruption we construct $\mathcal{B}$ as follows. When $\mathcal{B}$ receives $\overline{s}$ from its corruption oracle $\mathcal{C}_{\overline{s},k}^{\mathsf{btk},L}$ it requests to obtain the leak $(y_{p(1)}, \ldots, y_{p(n)})$ and returns this to the adversary $\mathcal{A}$. Otherwise the simulation and proof proceeds in the same fashion as before. □

We next consider the potential advantage that is given by the leaking capability of the leaky blinded-token corruption oracle. We have the following:

THEOREM 4.8. *Let $\epsilon > 0, n, k \in \mathbb{N}, \overline{s} \in \mathbb{N}^n$ be parameters. Given any sequence of statistically indistinguishable functions $\overline{f} = \langle f_1, \ldots, f_n \rangle$ we have $\mathcal{C}_{\overline{s},k}^{\mathsf{btk},L} \leq_{n \cdot \epsilon} \mathcal{C}_{\overline{s},k}^{\mathsf{btk}}$, where L is defined as in Theorem 4.7 and ϵ is an upper bound on the pairwise statistical distances for the sequence $\overline{f}$.*

PROOF. Basically, $\mathcal{B}$ will simulate $\mathcal{A}$ as well as the leak that is received by $\mathcal{A}$. The only issue in the simulation is that $\mathcal{B}$ is not privy to the permutation of its corruption oracle and it will still need to simulate the leak. $\mathcal{B}$ utilizes the sampler for the *hardest* of the $f_1, \ldots, f_n$ functions to obtain n instances $y_1, \ldots, y_n$. The simulator then provides $(y_1, \ldots, y_n)$ to $\mathcal{A}$. The simulator proceeds as follows: whenever the adversary submits a corruption message to the corruption oracle with a certain number of tokens the simulator passes through this request to its own corruption oracle. Due to the fact that the sampling done is at a distance at most ϵ away from regular operation for each pair of functions the behavior of $\mathcal{A}$ cannot result in a change of more than $n \cdot \epsilon$ in the execution experiment. □

With the above results we conclude that under the proper assumptions (namely IEP and hardness indistinguishability) the (blinded) token-based corruption oracle is an accurate abstraction of the (blinded) computational corruption oracle. It then follows from the results of Section 3 that the blinded computational corruption oracle remains t-safe even for values of t that far exceed the computational cost needed to corrupt a majority of player instances, thus establishing the value of the hidden diversity in the computational corruption setting.

5. SUMMARY AND DIRECTIONS FOR FUTURE RESEARCH

Adversaries in modern cryptography have thus far been allowed to "corrupt" parties for free. In this paper, motivated by the fact that different parties may require different resources to get corrupted, we put forth the notion of *resource-based corruptions*, where the adversary must invest some resources in order to corrupt parties. We further showed that in a resource *anonymous* setting (where "anonymous" is in the sense that such resource configuration is hidden from the adversary) much is to be gained in terms of efficiency and security.

We achieved the above through a series of technical contributions. These include the modeling of the corruption process in the setting of cryptographic protocols through *corruption oracles* as well as the introduction of a notion of reduction to relate such oracles; the abstraction of the corruption game as a combinatorial problem and its analysis; and, finally, the formulation of the notion of *inversion effort preserving* (IEP) functions and the property of *hardness indistinguishability*, which we apply to natural cryptographic candidates. An immediate next step is to expand the pool of such candidates.

In showing the advantages of hidden diversity, we have mainly shown that *there exist* instances where it provides significant benefits both in terms of security and efficiency. In the real world, however, systems may or may not mimic these instances and the potential benefits of hidden diversity may vary depending on the diversity distribution that actually exists. Investigating such benefits in a setting where corruption diversity follows a given probability distribution is an interesting research direction. Furthermore, one may consider the setting where a designer intervenes and is allowed to design the system taking diversity into account. This poses the problem of choosing the required corruption effort (bucket sizes), subject to a cost function. Can hidden diversity allow us to reduce cost, or increase security for a given level of expenditure? This is the subject of current research.

6. ACKNOWLEDGEMENTS

The authors are grateful to Ran Gelles, Arjen Lenstra and Alexander May for valuable comments. The work of Aggelos Kiayias was partly supported by EU projects CODAMODA and RECUP.

7. REFERENCES

[1] Mihir Bellare and Shafi Goldwasser. Verifiable partial key escrow. In Richard Graveman, Philippe A. Janson, Clifford Neumann, and Li Gong, editors, *ACM Conference on Computer and Communications Security*, pages 78–91. ACM, 1997.

[2] Mihir Bellare, Thomas Ristenpart, and Stefano Tessaro. Multi-instance security and its application to password-based cryptography. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 312–329. Springer, 2012.

[3] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[4] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with rsa and rabin. In *EUROCRYPT*, pages 399–416, 1996.

[5] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In Janos Simon, editor, *STOC*, pages 1–10. ACM, 1988.

[6] Dan Boneh and Moni Naor. Timed commitments. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254. Springer, 2000.

[7] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Cryptology ePrint Archive, Report 2000/067*, December 2005. Latest version at `http://eprint.iacr.org/2000/067/`.

[8] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In Miller [31], pages 639–648.

[9] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 11–19, New York, NY, USA, 1988. ACM.

[10] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In Juris Hartmanis, editor, *STOC*, pages 364–369. ACM, 1986.

[11] Don Coppersmith. Modifications to the number field sieve. *J. Cryptology*, 6(3):169–180, 1993.

[12] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 445–465. Springer, 2010.

[13] Yevgeniy Dodis. On extractors, error-correction and hiding all partial information. In *Theory and Practice in Information-Theoretic Security, 2005. IEEE Information Theory Workshop on*, pages 74 –79, oct. 2005.

[14] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Ernest F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.

[15] Tomás Feder, Eyal Kushilevitz, Moni Naor, and Noam Nisan. Amortized communication complexity. *SIAM J. Comput.*, 24(4):736–750, 1995.

[16] Ephraim Feig and Shmuel Winograd. On the direct

sum conjecture. *Linear Algebra and its Applications*, 63(0):193 – 219, 1984.

[17] Matthias Fitzi, Juan A. Garay, Ueli M. Maurer, and Rafail Ostrovsky. Minimal complete primitives for secure multi-party computation. *J. Cryptology*, 18(1):37–61, 2005.

[18] Juan A. Garay, David Johnson, Aggelos Kiayias, and Moti Yung. Resource-based corruptions and the combinatorics of hidden diversity. *IACR Cryptology ePrint Archive*, 2012:556, 2012.

[19] Juan A. Garay, Philip D. MacKenzie, Manoj Prabhakaran, and Ke Yang. Resource fairness and composability of cryptographic protocols. *J. Cryptology*, 24(4):615–658, 2011.

[20] Juan A. Garay and Carl Pomerance. Timed fair exchange of standard signatures (Extended Abstract). In Rebecca N. Wright, editor, *Financial Cryptography*, volume 2742 of *Lecture Notes in Computer Science*, pages 190–207. Springer, 2003.

[21] Oded Goldreich, Hugo Krawczyk, and Michael Luby. On the existence of pseudorandom generators. *SIAM J. Comput.*, 22(6):1163–1175, 1993.

[22] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In Alfred V. Aho, editor, *STOC*, pages 218–229. ACM, 1987.

[23] Iftach Haitner, Danny Harnik, and Omer Reingold. Efficient pseudorandom generators from exponentially hard one-way functions. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 228–239. Springer, 2006.

[24] Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptology*, 13(1):31–60, 2000.

[25] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized, and derandomized. *SIAM J. Comput.*, 39(4):1637–1665, 2010.

[26] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography from anonymity. In *FOCS*, pages 239–248. IEEE Computer Society, 2006.

[27] Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3/4):191–204, 1995.

[28] Arjen K. Lenstra, Hendrik W. Lenstra Jr., Mark S. Manasse, and John M. Pollard. The number field sieve. In Harriet Ortiz, editor, *STOC*, pages 564–572. ACM, 1990.

[29] Hendrik W. Lenstra. Factoring integers with elliptic curves. *The Annals of Mathematics*, 126(3):649–673, Nov. 1987.

[30] Silvio Micali and Rafael Pass. Local zero knowledge. In Jon M. Kleinberg, editor, *STOC*, pages 306–315. ACM, 2006.

[31] Gary L. Miller, editor. *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*. ACM, 1996.

[32] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[33] Ron Rivest, Adi Shamir, and David Wagner. Timed-lock puzzles and timed-release crypto. In *MIT/LCS/TR-684*, 1996.

[34] Alfredo De Santis, Giovanni Di Crescenzo, and Giuseppe Persiano. Necessary and sufficient assumptions for non-iterative zero-knowledge proofs of knowledge for all np relations. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 451–462. Springer, 2000.

[35] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *FOCS*, pages 427–436. IEEE Computer Society, 1992.

[36] Adi Shamir. Partial key escrow: A new approach to software key escrow. In *Key Escrow Conference*, 1995.

[37] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.

[38] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *FOCS*, pages 80–91. IEEE Computer Society, 1982.

APPENDIX

A. CANDIDATE IEP AND HARDNESS INDISTINGUISHABLE FUNCTIONS

In this section we study and give bounds for the exact hardness, the IEP property and hardness indistinguishability of three natural cryptographic candidates: random functions, discrete logarithm in the generic group model, and factorization. We note that all the functions we present here can be used in conjunction with constructions #1 and #2 given in Section 4.1.3 to derive hardness indistinguishable functions.

A.1 Random Functions

We first consider the notion of exact hardness and argue that for the inversion problem of a function that is modeled as a random oracle [3], we can calculate exactly the expression for exact hardness. In this model, we only count queries to the random oracle as the basic computational operation, and thus all complexity measures are expressed in terms of such queries.

PROPOSITION A.1. *Let $\epsilon > 0$ and $f : [2^\lambda] \to [2^\lambda]$ be modelled as a random oracle. Then $H_{f,\epsilon}(\lambda) = \epsilon \cdot 2^\lambda - 1$.*

PROOF. When f is modelled as a random oracle the probability $\mathbf{Pr}[\mathcal{A}_t(f(x)) \in f^{-1}(f(x))]$ is taken over all possible choices of x (uniformly selected from $[2^\lambda]$) and the choices of f (thought as a random table with 2^λ entries). Any algorithm that queries f will have probability $(q+1)2^{-\lambda}$ of returning the inverse of the input value $y = f(x)$ if it is allowed q queries. To see this consider the following: if $\mathcal{A}$ asks q distinct queries and finds the inverse among them it can return it for a probability of success equal to 1; otherwise, any $\mathcal{A}$ no matter which strategy it follows will have at best a $\frac{1}{n-q}$ chance of succeeding. Note that if $\mathcal{A}$ is repeating some queries the probability of success

would be worse, but this is something we can assume without loss of generality that it does not happen given that there is no restriction in space for $\mathcal{A}$. The statement of the proposition now follows since the probability of success is $q2^{-\lambda}+(1-q2^{-\lambda})(1/(2^{\lambda}-q))=(q+1)2^{-\lambda}$ by requiring this probability to be less than ϵ and considering a query to f as the basic computational step. □

We now study the inversion-effort-preserving property of random oracles. We show that if the functions are modeled as independent random oracles, then no speed up is possible for such set of instances.

PROPOSITION A.2. *For $n>0$, the sequence of functions $\{f_i\}_{i=1,\ldots,n}$ with $f_i:[2^{\lambda}]\rightarrow[2^{\lambda}]$ is IEP assuming each one of them is modelled as a random oracle.*

PROOF. We will use the fact that querying function f_i gives no information about function f_j for $j\neq i$. As in Proposition A.1, the inversion of function f_i can succeed with probability $(q_i+1)2^{-\lambda}$ as long as q_i queries are made. Also as before, any algorithm inverting f_i with probability ϵ requires $\epsilon\cdot 2^{\lambda}-1$ queries to f_i. We will show that any algorithm running in $t=n\epsilon\cdot 2^{\lambda}-n$ steps cannot invert all of the functions with probability greater than ϵ. We argue as follows. At the end of any execution at the specified number of steps t we can build a table of size q_i for each function f_i; we have that $\sum_{i=1}^{n} q_i\leq t$. By conditioning on the subset A of the functions for which a good query has been made, we obtain that the probability of success can be upper-bounded by the expression

$$\sum_{A\subseteq[n]}\prod_{i\in A}\frac{q_i}{2^{\lambda}}\cdot\prod_{i\notin A}(1-\frac{q_i}{2^{\lambda}})\cdot\prod_{i\notin A}\frac{1}{2^{\lambda}-q_i}$$

$$=\sum_{A\subseteq[n]}\frac{\prod_{i\in A}q_i}{2^{\lambda n}}=\frac{\prod_{i=1}^{n}(q_i+1)}{2^{\lambda n}}\leq\left(\frac{n+\sum_{i=1}^{n}q_i}{n\cdot 2^{\lambda}}\right)^n$$

where the last inequality follows from the arithmetic and geometric means inequality. Now applying the bound $\sum_{i=1}^{n} q_i\leq t=n\epsilon\cdot 2^{\lambda}-n$ and the fact that $\epsilon^n\leq\epsilon$ we bound the probability by ϵ. The result on $H_{f^S,\epsilon}$ (cf. Definition 4.4) now immediately follows. □

A.2 Discrete Logarithm

We now consider the exact hardness of discrete logarithms in the generic group model (see, e.g., [37]). In this setting, any algorithm $\mathcal{A}$ solving the discrete logarithm is given the encodings of two group elements, $\sigma(1)$ and $\sigma(x)$, where $\sigma:\mathbb{Z}_q\rightarrow S$ is a random permutation. Here q is a prime number that is λ-bits.

$\mathcal{A}$ aims to discover x but has no way to internally emulate the group operation. Instead, the algorithm operates with access to an oracle that takes input (r,s) and returns the value $\sigma(r+sx)$; we measure those oracle calls as the basic steps taken by the algorithm.

We denote all the queries of $\mathcal{A}$ to the oracle by $Q_{\mathcal{A}}$. A key observation used in [37] is that as long as it holds that $r+sx=r'+s'x \bmod q$ for two queries $(r,s),(r',s')$ to the oracle, it is possible to reconstruct x. The second key observation is that if this event does not occur then $\mathcal{A}$'s behavior is independent of x. It follows that the success of algorithm $\mathcal{A}$ in solving the discrete logarithm problem is bounded by the probability that $r+sx=r'+s'x$ or $y=x$ for some $(r,s),(r',s')\in Q_{\mathcal{A}}$, where x is uniformly distributed over $\mathbb{Z}_q$ and y is some arbitrarily distributed value (in this setting, y would capture the output of $\mathcal{A}$). If the algorithm performs k queries then we have that the probability of success is bounded by $(\binom{k}{2}+1)/q$ (this stems from the fact that the k queries can be thought of lines over a plane that determine $\binom{k}{2}$ cut points—the probability of equality amounts to hitting one of those cut points). Based on this we state the following.

PROPOSITION A.3. *Let $\epsilon>0$ and q be a λ-bit prime number. Suppose $f:\mathbb{Z}_q\rightarrow S$ is the exponentiation function over a generic multiplicative group. Then $H_{f,\epsilon}\geq\sqrt{2q\epsilon}$.*

PROOF. The proof follows from the argument presented above (due to [37]). Suppose $H_{f,\epsilon}<T=\sqrt{2q\epsilon}$. This means that there is an algorithm that in T steps achieves a probability of success ϵ. But we observe that T is such that $(\binom{T}{2}+1)/q<\epsilon$, hence a contradiction following the above arguments (assuming $T\geq 2$). □

Next, we consider the IEP property in the same setting. We consider the solution of n independent instances of the discrete-logarithm problem over n groups. Each group is assumed to have a separate encoding of elements σ_i that is independently selected. Specifically, an algorithm $\mathcal{A}$ is given the pairs $(\sigma_i(1),\sigma(x_i))$ for $i=1,\ldots,n$ and aims to produce the vector $(x_1,\ldots,x_n)$. $\mathcal{A}$ is allowed to access an oracle that accepts queries of the form (i,r,s) and returns $\sigma_i(r+sx_i)$.

PROPOSITION A.4. *Let $\epsilon>0$, $n>0$ and q be a prime λ-bit number. Suppose $f^{[n]}:(\mathbb{Z}_q)^n\rightarrow S_1\times\ldots\times S_n$ is the coordinate-wise exponentiation function over a sequence of n generic multiplicative groups. Then $H_{f^{[n]},\epsilon}\geq\sqrt{2n\cdot\epsilon^{1/n}\cdot q}$.*

PROOF. Suppose that $H_{f^{(n)},\epsilon}<T=\lceil\sqrt{2n\epsilon^{1/n}q}\rceil$. This means that there is an algorithm $\mathcal{A}$ that in T steps has probability of success at least ϵ. Define by t_i the number of queries posed to the i-th oracle by $\mathcal{A}$. Then it holds that $\sum_{i=1}^{n} t_i=T$ since we count only oracle queries in the running time of $\mathcal{A}$. We let C_i be the event that a collision has occurred at the i-th oracle. Now let $B\subseteq[n]$ be the set of groups over which the event C_i takes place, i.e., for all i, $i\in B$ if and only if C_i is true. Assuming this conditioning we know that the probability of success of algorithm $\mathcal{A}$ is at most $(1/q)^{n-|B|}$ since it essentially has to guess all the other queries. Furthermore, the probability of event B happening is the probability that we get a collision; given the independence of the group encodings this is $\prod_{i\in B}\binom{t_i}{2}/q$. So overall we have that the probability is bounded by

$$q^{-n}\cdot\sum_{B\subseteq[n]}\prod_{i\in B}\binom{t_i}{2}=q^{-n}\cdot\prod_{i=1}^{n}(\binom{t_i}{2}+1)$$

$$\leq(n\cdot q)^{-n}(n+\sum_{i=1}^{n}\binom{t_i}{2})^n,$$

where the last inequality follows from the geometric-arithmetic means inequality. Observe now that $\sum_{i=1}^{n}\binom{t_i}{2}=\frac{1}{2}\cdot(\sum_{i=1}^{n}t_i^2-T)<T(T-1)/2$. From this we obtain that the probability of success ϵ is strictly bounded by $(T^2/2qn)^n$. This is a contradiction since $(T^2/2qn)^n\geq\epsilon$ by definition of T. □

COROLLARY A.5. *Let $\epsilon > 0, n > 0$, and q be a λ-bit prime number. Suppose $f_i : \mathbb{Z}_q \to S_i$ is the exponentiation function over a generic multiplicative group S_i. Then the set of functions $\{f_i\}_{i=1,\ldots,n}$ is τ-IEP for $\tau(\cdot) = (\cdot)^{1/2}$.*

PROOF. By definition we need to show that for any subset $S \subseteq [n]$ we have that $H_{f^S,\epsilon} \geq \tau(\sum_{i \in S} H_{f_i,\epsilon})$. Using Propositions A.3 and A.4 we have that

$$H_{f^S,\epsilon} \geq \sqrt{|S|} \cdot \sqrt{2q\epsilon^{1/n}} \geq$$

$$\sqrt{|S|} \cdot \max_{i \in S} H_{f_i,\epsilon} \geq (\sum_{i \in S} H_{f_i,\epsilon})^{1/2},$$

which completes the proof. □

A.3 Factoring

The complexity to factor an integer N with the best known algorithm (the number field sieve [28]) takes time on the order of $L[1/3, 1.9229]$, where

$$L[\alpha, \beta] = \exp((\beta + o(1))(\log N)^{\alpha}(\log \log N)^{1-\alpha}),$$

and was improved to $L[1/3, 1.902]$ by Coppersmith [11]. Assuming that this result matches the complexity of the factoring problem, this will provide bounds for the exact hardness of the multiplication function. Specifically, if P_λ is the set of all λ-bit prime numbers, we have that if $f_{\mathsf{mult}} : \mathsf{P}_\lambda \times \mathsf{P}_\lambda \to \mathbb{N}$ with $f_{\mathsf{mult}}(p, q) = p \cdot q$, then for any ϵ,

$$\epsilon \cdot L(1/3, 1.902) \leq H_{f,\epsilon} \leq \epsilon \cdot L(1/2, \sqrt{2}) \quad (1)$$

According to the above statement, the exact hardness of factoring is a subexponential function that is bounded from below by a time complexity that is derived from the running time of the number field sieve algorithm. We note that a similar assumption was made in [4] when stating the "exact security" of the RSA function. The upper bound in the statement is derived from Lenstra's elliptic curve factorization algorithm [29]. This algorithm is suitable for expressing an upper bound in the form above since it is an algorithm that repetitively picks an elliptic curve and a point on it, and then tests whether the group law holds by calculating a scalar of the point. The complexity of the algorithm is determined by the fact that after picking around $L(1/2, \sqrt{2})$ curves, then with very high probability a "bad" curve will be found. As a result, if we wish to succeed with probability ϵ, we should sample about $\epsilon \cdot L(1/2, \sqrt{2})$ curves.

Regarding the IEP property, it was shown in [11] that it is possible to amortize cost when factoring n integers and there exists an algorithm with expected time complexity $L[1/3, 2.0068] + n \cdot L[1/3, 1.6386]$. Under the assumption that Coppersmith's algorithm (and small optimizations thereof) is the best possible when trying to factor simultaneously a sequence of moduli, one might be willing to assume that if $f^{[n]}$ is the function that multiplies n pairs of primes to the corresponding moduli, the following lower bound would hold true:

$$H_{f^{[n]},\epsilon} \geq \epsilon \cdot n \cdot L[1/3, \sqrt{2}]. \quad (2)$$

From the above two assumptions, the τ-IEP property for factoring is satisfied by letting $\tau(x) = e^{(\ln x)^{2/3}}$. This holds due to the fact that for any n, ϵ, we have that $\epsilon \cdot n \cdot L(1/3, \sqrt{2}) \geq \tau(\epsilon \cdot n \cdot L(1/2, \sqrt{2}))$.