Partial Fairness in Secure Two-Party Computation

S. Dov Gordon* Jonathan Katz*

Abstract

Complete fairness is impossible to achieve, in general, in secure two-party computation. In light of this, various techniques for obtaining *partial* fairness in this setting have been suggested. We explore the possibility of achieving partial fairness with respect to a strong, simulation-based definition of security within the standard real/ideal world paradigm. We show feasibility with respect to this definition for randomized functionalities where each player may possibly receive a different output, as long as at least one of the domains or ranges of the functionality are polynomial in size. When one of the domains is polynomial size, our protocol is also secure-with-abort. In contrast to much of the earlier work on partial fairness, we rely on standard assumptions only (namely, enhanced trapdoor permutations).

We also provide evidence that our results are, in general, optimal. Specifically, we show a boolean function defined on a domain of super-polynomial size for which it is impossible to achieve both partial fairness and security with abort, and provide evidence that partial fairness is impossible altogether for functions whose domains and ranges all have super-polynomial size.

1 Introduction

In the setting of secure two-parry computation, two parties wish to run some protocol that will enable each of them to learn a (possibly different) function of their inputs while preserving, to the extent possible, security properties such as privacy, correctness, input independence, etc. These requirements, and more, are typically formalized by comparing a real-world execution of the protocol to an *ideal world* where there is a trusted entity who performs the computation on behalf of the parties. Informally, a given protocol is said to be "secure" if for any real-world adversary $\mathcal{A}$ there exists a corresponding ideal-world adversary $\mathcal{S}$ (corrupting the same party as $\mathcal{A}$) such that the result of executing the protocol in the real world with $\mathcal{A}$ is computationally indistinguishable from the result of computing the function in the ideal world with $\mathcal{S}$.

One desirable security property is *fairness* which, intuitively, means that either *both parties* learn the output, or else *neither party* does. In a "true" ideal world — and this is the ideal world used for proving security in the multi-party setting when a majority of parties are honest — fairness is ensured since the trusted party would, in fact, provide output to both parties. Unfortunately, results of Cleve [9] show that complete fairness is impossible to achieve, in general, in the two-party setting. For this reason, the usual treatment of secure two-party computation within the real/ideal world paradigm (see [17]) *weakens* the ideal-world model to one in which fairness is not guaranteed at all. A protocol is then defined to be "secure-with-abort" if it can be simulated (as described above) with respect to this, less-satisfying ideal-world model.

*Dept. of Computer Science, University of Maryland. Email: {gordon,jkatz}@cs.umd.edu. This work was supported by NSF CAREER award #0447075.

Various methods for achieving *partial* fairness have been suggested; we provide an extensive discussion in Section 1.2. With the exception of [16], however, all previous work has departed from the real/ideal paradigm in defining partial fairness; this deficiency is explicitly noted, for example, by Goldreich [17, Section 7.7.1.1]. Our aim is to define, and achieve, a meaningful notion of partial fairness *while staying within the traditional real/ideal paradigm*. Furthermore, many previously-suggested protocols only apply in certain settings (e.g., fair exchange of signatures) or under certain assumptions on the parties' inputs (e.g., that inputs are chosen uniformly at random) but do not give a "general-purpose" solution that can be used for arbitrary functions computed on arbitrary inputs. In contrast, protocols analyzed within the real/ideal paradigm do not suffer from these drawbacks. Finally, we note that much previous work on partial fairness requires strong cryptographic assumptions (e.g., regarding the precise amount of time needed to perform some computation, even using parallelism); we would prefer a solution based on standard assumptions.

As we have remarked already, the most desirable (but, in general, unachievable) definition of security requires computational indistinguishability between the real world and a "true" ideal world where both parties receive output. The standard relaxation of security-with-abort [17] leaves unchanged the requirement of computational indistinguishability, but weakens the ideal world to one in which fairness is no longer guaranteed at all. Katz [21], in a slightly different context, suggested an alternate relaxation: keep the ideal world unchanged, *but relax the notion of simulation* and require instead that the real and ideal worlds be distinguishable with probability at most $\frac{1}{p}+\mathsf{negl}$, for p some specified polynomial[1] (see Definition 1). We refer to a protocol satisfying this definition as being "$\frac{1}{p}$-secure". Cleve [9] and Moran et al. [24] show $\frac{1}{p}$-secure protocols for two-party coin tossing with $\mathcal{O}(p^2)$ and $\mathcal{O}(p)$ rounds, respectively.[2] We are not aware of any other results that satisfy our definition and, in particular, none of the previous approaches for achieving partial fairness appear to yield protocols that are $\frac{1}{p}$-secure. See Section 1.2 for further discussion.

1.1 Our Results

We show general feasibility results for partial fairness in the two-party setting, with respect to the definition of $\frac{1}{p}$-security. Specifically, let $f_n : X_n \times Y_n \to Z_n^1 \times Z_N^2$ be a (randomized) functionality where player 1 (resp., player 2) provides input $x \in X_n$ (resp., $y \in Y_n$) and receives output $z^1 \in Z_n^1$ (resp., $z^2 \in Z_n^2$). (Throughout this paper, n denotes the security parameter.) Assuming the existence of enhanced trapdoor permutations we prove, for arbitrary polynomial p, the existence of a $\frac{1}{p}$-secure protocol for computing f_n as long as at least one of X_n, Y_n, Z_n^1, Z_n^2 is polynomial size (in n). When either X_n or Y_n is polynomial-size, our protocol is also secure-with-abort.

As far as general feasibility results go, we also show that our results are essentially the best possible. First, we show an example of a deterministic, boolean function $f_n : X_n \times Y_n \to \{0, 1\}$, where $|X_n|$ and $|Y_n|$ are both super-polynomial, for which no protocol computing f_n can achieve both security-with-abort and $\frac{1}{p}$-security (for $p > 4$) simultaneously. We also show that if exponentially-strong one-way functions exist, then there exists a deterministic function $f_n : X_n \times Y_n \to Z_n$, with each of X_n, Y_n, Z_n super-polynomial in size, such that f cannot be $\frac{1}{p}$-securely computed for $p > 2$. Taken together, our work thus settles the main open questions in this direction.

[1] This definition is similar in spirit to (but weaker than) the notion of ϵ-zero knowledge [12] and is analogous to the definition used in [18] in the context of password-based key exchange, although there the value of p is fixed by the size of the password dictionary. A similar idea, formalized differently and in a different context, is also used in [1].

[2] Actually, they prove something weaker but one can show that their protocols satisfy our stronger definition.

1.2 Prior Work

There is an extensive literature devoted to the problem of achieving partial fairness when an honest majority is not present, both for the case of specific functionalities like coin tossing [9, 10, 24] and contract signing/exchanging secrets [5, 22, 13, 4, 11], as well as for the case of general functionalities [26, 15, 3, 19, 14, 6, 25, 16]. Prior work (with the exception of [16]), however, does not consider a *simulation-based* security definition of the sort we do here. Moreover, to the best of our knowledge none of the previous approaches (with the exception of [9, 24], that deal only with coin tossing) can be proven $\frac{1}{p}$-secure.

Garay et al. [16] give a simulation-based formalization of "gradual release" [15, 3, 11, 6, 25] within the universal composability framework [8]. The guarantee their protocol provides, informally, is that at any point in the protocol both the adversary and the honest party can obtain their entire output by investing a "similar" amount of work. Somewhat unsatisfying is that the decision of whether an honest party should invest the necessary work and recover the output is not mandated by the protocol, but is somehow supposed to be decided "externally". A gradual release approach also seems problematic in defending against an adversary who runs in polynomial time, but has more computational power than honest parties are able to invest. Finally, we add that the proof of security in [16] relies on a strong, non-standard assumption regarding the precise time required to solve a *specific* computational problem.

Gordon et al. [20] recently showed that, in contrast to the accepted folklore, *complete* fairness is possible in the two-party setting for certain non-trivial functions. Work continuing that direction should be viewed as complementary to our work here: while we do not yet have a complete characterization of what *can* be computed with complete fairness, we know that there certainly do exist some functions that *cannot* be computed with complete fairness [9] and so some relaxation must be considered (at least for some functions). Note further that currently the only (non-trivial) feasibility results for complete fairness in the two-party setting [20] pertain to single-output, deterministic, boolean functions over polynomial-size domains. Our feasibility results here apply to substantially-larger classes of functions.

1.3 Overview of our Approach

We now give an informal description of our feasibility result. Let x denote the input of P_1, let y denote the input of P_2, and let $f : X \times Y \rightarrow Z$ denote the function they are trying to compute. (For simplicity, here we focus on the case when each party receives the same output; the more general case is handled when we formally describe our protocols.) As in [21, 20, 24], our protocols will be composed of two stages, where the first stage can be viewed as a "pre-processing" step and the second stage takes place in a sequence of $r = r(n)$ iterations. The stages have the following form:

First stage The first stage includes the following steps:

1. First, a value $i^* \in \{1, \ldots, r\}$ is chosen according to some distribution.
2. Values $a_1, \ldots, a_r$ and $b_1, \ldots, b_r$ are generated. For $i < i^*$, the value a_i (resp., b_i) is chosen according to some distribution that is independent of y (resp., x). For $i \geq i^*$, however, it holds that $a_i = b_i = f(x, y)$.
3. Each a_i is randomly shared as $a_i^{(1)}, a_i^{(2)}$ with $a_i^{(1)} \oplus a_i^{(2)} = a_i$ (and similarly for each b_i). The stage concludes with P_1 being given $a_1^{(1)}, b_1^{(1)}, \ldots, a_r^{(1)}, b_r^{(1)}$, and P_2 being given

$a_1^{(2)}, b_1^{(2)}, \ldots, a_r^{(2)}, b_r^{(2)}$. (These shares are also authenticated using an information-theoretic MAC, but we omit this in the current description of the protocol.)

Note that, at the end of this stage, each party only has a set of random shares that reveal nothing about the other party's input. This stage can therefore be carried out by any two-party protocol that is secure-with-abort.

Second stage In each iteration i, for $i = 1, \ldots, r$, the parties do the following: First, P_2 sends $a_i^{(2)}$ to P_1 who reconstructs a_i; then P_1 sends $b_i^{(1)}$ to P_1 who reconstructs b_i. (In the real protocol, parties must also verify validity of the shares but we omit this step here.) If a party (say, P_1) aborts or otherwise fails to send a valid message in some iteration i, then the other party (here, P_2) outputs the value reconstructed in the previous iteration (i.e., b_{i-1}). Otherwise, parties reach the end of the protocol and output a_r and b_r, respectively.

The above defines a generic template, but to fully specify the protocol we must specify the distribution of i^* as well as the distribution of the a_i, b_i for $i < i^*$. As in [21, 24], we choose i^* uniformly from $\{1, \ldots, r\}$. (In [20] a geometric distribution was used. That would work in our context as well, but would result in worse round complexity.) For the case when X and Y (the domains of f) are polynomial size, we follow [20] and set $a_i = f(x, \hat{y})$ for $\hat{y}$ chosen uniformly from Y, and set $b_i = f(\hat{x}, y)$ for $\hat{x}$ chosen uniformly (and independently) from X. Note that a_i (resp., b_i) is independent of y (resp. x), as desired.

Intuitively, this is partially fair for the following reasons: fairness is only violated if P_1 aborts exactly in iteration i^* (indeed, if it aborts before iteration i^* then neither party learns the "correct" value of the function, while if it aborts subsequently then both parties learn the correct value). But *even if P_1 knows the value of $z = f(x, y)$*, it cannot determine when iteration i^* occurs with certainty because $a_i = z$ with some noticeable probability α even when $i < i^*$. In [24] (which deals with coin tossing), the distribution of a_i for $i < i^*$ is identical to the distribution of a_{i^*}, and so it is relatively straightforward to see that P_1 cannot abort in iteration i^* except with probability at most $1/r$, where r is the number of iterations. In our case things are complicated by the fact that the distribution of a_i for $i < i^*$ is different from the distribution of a_{i^*}, and furthermore the adversary may know (partial information about) the correct result $a_{i^*} = f(x, y)$. Nevertheless, we use a combinatorial argument (see Lemma 1) to prove that the adversary cannot abort in iteration i^* except with probability at most $1/\alpha r$. Taking $\alpha^* = \min_{z \in f(x,Y)}\{\Pr[a_i = z \mid i < i^*]\}$, we conclude that setting $r = p/\alpha^*$ suffices to achieve $\frac{1}{p}$-security. As long as Y is polynomial-size, $\alpha^* \geq 1/|Y|$ is noticeable and we get a protocol with polynomially-many rounds.

The above approach does not work without additional modifications in the case when Y has super-polynomial size. Essentially, this is because it could be the case that $f(x, y) = z$ for some value z for which the probability that $f(x, \hat{y}) = z$ when $\hat{y}$ is chosen uniformly from Y is negligible. In this case, conditioned on $a_i = z$ is it overwhelmingly likely that $i = i^*$, and this gives a strategy for the adversary to abort exactly in iteration i^* with overwhelming probability. (Namely, abort in the first iteration when $a_i = z$.) To fix this, we must ensure that every possible output in Z (the range of f) occurs with noticeable probability. We do this by changing the distribution of a_i (for $i < i^*$) as follows: with probability $1 - 1/q$ choose a_i as before, but with probability $1/q$ choose a_i uniformly from Z. (We will set q in a moment, but it will be polynomial in n.) Defining $\alpha^* = \min_{z \in Z}\{\Pr[a_i = z \mid i < i^*]\}$ we now have $\alpha^* \geq 1/q|Z|$, which is noticeable when $|Z|$ is polynomial. Furthermore, as before, setting $r = p/\alpha^* \leq pq|Z|$ suffices to ensure that the adversary cannot abort in iteration i^* except with probability at most $1/p$.

By changing the distribution of a_i, however, we introduce a new problem: if a malicious P_2 aborts in some iteration prior to i^*, then the output of the honest P_1 in the real world cannot necessarily be simulated in the ideal world. We show, however, that it *can* be simulated to within statistical difference $O(1/q)$. Taking $q = p$ (along with $r = pq|Z|$) thus gives a protocol with polynomially-many rounds that is $\frac{1}{p}$-secure.

We refer the reader to Section 3 for further details regarding our protocols.

1.4 Organization of the Paper

We present our definitions in Section 2. These are largely standard, except for our definition of $\frac{1}{p}$-security (Definition 1). The reader may also want to quickly review our notation for functionalities. In Section 3 we describe protocols showing feasibility of $\frac{1}{p}$-secure computation for a large class of functionalities. Our impossibility results are given in Section 4.

2 Definitions

We denote the security parameter by n.

2.1 Preliminaries

A function $\mu(\cdot)$ is $\mathsf{negligible}$ if for every positive polynomial $p(\cdot)$ and all sufficiently large n it holds that $\mu(n) < 1/p(n)$. A *distribution ensemble* $X = \{X(a,n)\}_{a\in\mathcal{D}_n,\, n\in\mathbb{N}}$ is an infinite sequence of random variables indexed by $a \in \mathcal{D}_n$ and $n \in \mathbb{N}$, where $\mathcal{D}_n$ is a set that may depend on n. For a fixed function p, two distribution ensembles $X = \{X(a,n)\}_{a\in\mathcal{D}_n,\, n\in\mathbb{N}}$ and $Y = \{Y(a,n)\}_{a\in\mathcal{D}_n,\, n\in\mathbb{N}}$ are *computationally* $\frac{1}{p}$*-indistinguishable*, denoted $X \overset{1/p}{\approx} Y$, if for every non-uniform polynomial-time algorithm D there exists a negligible function $\mu(\cdot)$ such that for every n and every $a \in \mathcal{D}_n$

$$\left|\Pr[D(X(a,n)) = 1] - \Pr[D(Y(a,n)) = 1]\right| \le \frac{1}{p(n)} + \mu(n).$$

Two distribution ensembles are *computationally indistinguishable*, denoted $X \overset{\mathrm{c}}{\equiv} Y$, if for every $c \in \mathbb{N}$ they are computationally $\frac{1}{n^c}$-indistinguishable.

The statistical difference between two distributions $X(a,n)$ and $Y(a,n)$ is defined as

$$\mathsf{SD}\left(X(a,n),\, Y(a,n)\right) = \frac{1}{2} \cdot \sum_s \left|\Pr[X(a,n) = s] - \Pr[Y(a,n) = s]\right|,$$

where the sum ranges over s in the support of either $X(a,n)$ or $Y(a,n)$. Two distribution ensembles $X = \{X(a,n)\}_{a\in\mathcal{D}_n,\, n\in\mathbb{N}}$ and $Y = \{Y(a,n)\}_{a\in\mathcal{D}_n,\, n\in\mathbb{N}}$ are *statistically close*, denoted $X \overset{\mathrm{s}}{\equiv} Y$, if there is a negligible function $\mu(\cdot)$ such that for every n and every $a \in \mathcal{D}_n$, it holds that $\mathsf{SD}\left(X(a,n), Y(a,n)\right) \le \mu(n)$.

2.2 Two-Party Computation

Functionalities. In the two-party setting, a *functionality* $\mathcal{F} = \{f_n\}_{n\in\mathbb{N}}$ is a sequence of randomized processes for which f_n can be computed in time $\mathsf{poly}(n)$. We view f_n as a (randomized) mapping $f_n : X_n \times Y_n \to Z_n^1 \times Z_n^2$, where X_n (resp., Y_n) denotes the valid inputs of the first (resp., second) party and we assume that elements of X_n, Y_n, Z_n^1, and Z_n^2 can be described by strings of length $\mathsf{poly}(n)$. We write $f_n = (f_n^1, f_n^2)$ if we wish to emphasize the two outputs of f_n, but stress that if f_n^1 and f_n^2 are randomized then the outputs of f_n^1 and f_n^2 are correlated random variables. If $\Pr[f_n^1(x,y) = f_n^2(x,y)] = 1$ for all x, y, then we call f_n a *single-output functionality* and write it as $f_n : X_n \times Y_n \to Z_n$.

In the rest of the paper, we frequently drop the explicit dependence on n. It is important to keep in mind, however, that the domain and range of f depend n, since the complexity of our protocols will depend on their sizes.

Two-party computation. A two-party protocol for computing a functionality $\mathcal{F} = \{(f^1, f^2)\}$ is a protocol running in polynomial time and satisfying the following functional requirement: if party P_1 begins by holding 1^n and input $x \in X$, and party P_2 holds 1^n and input $y \in Y$, then the joint distribution of the outputs of the parties is statistically close to $(f^1(x,y), f^2(x,y))$.

In what follows, we define what we mean by a *secure* protocol. Our definition uses the standard real/ideal paradigm of [17] (based on [23, 2, 7]), except that we will sometimes require only $\frac{1}{p}$-indistinguishability rather than indistinguishability. We consider *active* adversaries, who may deviate from the protocol in an arbitrary manner, and static corruptions.

Security of protocols (informal). The security of a protocol is analyzed by comparing what an adversary can do in a real protocol execution to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an *ideal* computation involving an incorruptible *trusted party* to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted party exists) can do no more harm than if it was involved in the above-described ideal computation.

Execution in the ideal model. The parties are P_1 and P_2, and there is an adversary $\mathcal{A}$ who has corrupted one of them. An ideal execution for the computation of $\mathcal{F} = \{f_n\}$ proceeds as follows:

Inputs: P_1 and P_2 hold 1^n and inputs $x \in X_n$ and $y \in Y_n$, respectively; the adversary $\mathcal{A}$ receives an auxiliary input aux.

Send inputs to trusted party: The honest party sends its input to the trusted party. The corrupted party controlled by $\mathcal{A}$ may send any value of its choice. Denote the pair of inputs sent to the trusted party by (x', y').

Trusted party sends outputs: If $x' \notin X_n$ the trusted party sets x' to some default element $x_0 \in X$ (and likewise if $y' \notin Y_n$). Then, the trusted party chooses r uniformly at random and sends $f_n^1(x', y'; r)$ to P_1 and $f_n^2(x', y'; r)$ to party P_2.

Outputs: The honest party outputs whatever it was sent by the trusted party, the corrupted party outputs nothing, and $\mathcal{A}$ outputs any arbitrary (probabilistic polynomial-time computable) function of its view.

We let $\text{IDEAL}_{\mathcal{F},\mathcal{A}(\mathsf{aux})}(x, y, n)$ be the random variable consisting of the output of the adversary and the output of the honest party following an execution in the ideal model as described above.

Execution in the real model. We next consider the real model in which a two-party protocol π is executed by P_1 and P_2 (and there is no trusted party). In this case, the adversary $\mathcal{A}$ gets the inputs of the corrupted party and sends all messages on behalf of this party, using an arbitrary polynomial-time strategy. The honest party follows the instructions of π.

Let π be a two-party protocol computing $\mathcal{F}$. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time machine with auxiliary input aux. We let $\text{REAL}_{\pi,\mathcal{A}(\mathsf{aux})}(x, y, n)$ be the random variable consisting of the view of the adversary and the output of the honest party, following an execution of π where P_1 begins by holding 1^n and input x, and P_2 begins by holding 1^n and input y.

Security as emulation of an ideal execution in the real model. Having defined the ideal and real models, we can now define security of a protocol. Loosely speaking, the definition asserts that a secure protocol (in the real model) emulates the ideal model (in which a trusted party exists). This is formulated as follows:

Definition 1 *Let $\mathcal{F}, \pi$ be as above, and fix a function p. Protocol π is said to* $\frac{1}{p}$-securely compute $\mathcal{F}$ *if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that*

$$\left\{\text{IDEAL}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x, y, n)\right\}_{(x,y)\in X\times Y,\, \mathsf{aux}\in\{0,1\}^*} \stackrel{1/p}{\approx} \left\{\text{REAL}_{\pi,\mathcal{A}(\mathsf{aux})}(x, y, n)\right\}_{(x,y)\in X\times Y,\, \mathsf{aux}\in\{0,1\}^*} .$$

We stress that, in the above definition, we compare the real-world execution of the protocol to an ideal world in which fairness is guaranteed. $\frac{1}{p}$-security thus guarantees, in particular, that fairness is guaranteed in the real world except with probability at most $\frac{1}{p}$.

We also define *security-with-abort* in the standard way [17]. We remark that the notions of $\frac{1}{p}$-security and security-with-abort are incomparable.

2.3 Information-Theoretic MACs

We briefly review the standard definition for information-theoretically secure message authentication codes (MACs). A *message authentication code* consists of three polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$. The *key-generation algorithm* Gen takes as input the security parameter 1^n in unary and outputs a key k. The *message authentication algorithm* Mac takes as input a key k and a message $M \in \{0,1\}^{\leq n}$, and outputs a tag t; we write this as $t = \mathsf{Mac}_k(M)$. The *verification algorithm* Vrfy takes as input a key k, a message $M \in \{0,1\}^{\leq n}$, and a tag t, and outputs a bit b; we write this as $b = \mathsf{Vrfy}_k(M, t)$. We regard $b = 1$ as acceptance and $b = 0$ as rejection, and require that for all n, all k output by $\mathsf{Gen}(1^n)$, all $M \in \{0,1\}^{\leq n}$, it holds that $\mathsf{Vrfy}_k(M, \mathsf{Mac}_k(M)) = 1$.

We say $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ is a *secure m-time MAC*, where m may be a function of n, if no computationally-unbounded adversary can output a valid tag on a new message after seeing valid tags on m other messages. For our purposes, we do not require security against an adversary who adaptively chooses its m messages for which to obtain a valid tag; it suffices to consider a non-adaptive definition where the m messages are fixed in advance. (Nevertheless, known constructions satisfy the stronger requirement.) Formally:

Definition 2 *Message authentication code* $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *is an* information-theoretically secure m-time MAC *if for any sequence of messages $M_1, \ldots, M_m$ and any adversary $\mathcal{A}$, the following is negligible in the security parameter n:*

$$\Pr\left[\begin{array}{l} k \leftarrow \mathsf{Gen}(1^n);\ \forall i : t_i = \mathsf{Mac}_k(M_i); \\ (M', t') \leftarrow \mathcal{A}(M_1, t_1, \ldots, M_m, t_m) \end{array} :\ \mathsf{Vrfy}_k(M', t') = 1 \bigwedge M' \notin \{M_1, \ldots, M_m\}\right].$$

3 $\frac{1}{p}$-Secure Computation of General Functionalities

We begin in Section 3.1 by stating and proving a combinatorial lemma that will form an essential piece of our analysis in the two sections that follow. The reader who is willing to accept the results of that lemma on faith is welcome to skip to Section 3.2 where we demonstrate a $\frac{1}{p}$-secure protocol that works for functionalities defined on polynomial-size domains. A slight modification of this protocol is also secure-with-abort. To keep the exposition as simple as possible, we restrict our attention there to single-output functionalities, though the techniques extend easily to the case where each party receives a different function of the inputs. In Section 3.3 we show how to adapt the protocol for functionalities defined over domains of super-polynomial size (but polynomial range), and also generalize to the case of functionalities generating different outputs for each party.

3.1 A Useful Lemma

In this section, we analyze an abstract game Γ between a challenger and an (unbounded) adversary $\mathcal{A}$. The game is parameterized by a value $\alpha \in (0, 1]$ and an integer $r \geq 1$. Fix (arbitrary) distributions D_1, D_2 such that for every z it holds that

$$\Pr_{a \leftarrow D_1}[a = z] \geq \alpha \cdot \Pr_{a \leftarrow D_2}[a = z]. \tag{1}$$

The game $\Gamma(\alpha, r)$ proceeds as follows:

1. The challenger chooses i^* uniformly from $\{1, \ldots, r\}$, and then prepares values $a_1, \ldots, a_r$ as follows:
   - For $i < i^*$, it chooses $a_i \leftarrow D_1$.
   - For $i \geq i^*$, it sets $a_i \leftarrow D_2$.
2. The challenger and the adversary then interact in a sequence of at most r iterations. In iteration i:
   - The challenger gives a_i to the adversary.
   - The adversary can either abort or $\mathsf{continue}$. In the former case, the game stops. In the latter case, the game continues to the next iteration.
3. $\mathcal{A}$ wins if it aborts the game in iteration i^*. (Since $\mathcal{A}$ can no longer win once iteration i^* has passed, we may simply assume the game stops if that ever occurs.)

Let $\mathsf{Win}(\alpha, r)$ denote the maximum probability with which $\mathcal{A}$ can win the above game.

Lemma 1 *For any D_1, D_2 satisfying Equation (1),* $\mathsf{Win}(\alpha, r) \leq 1/\alpha r$.

Proof: Fix D_1, D_2 satisfying Equation (1). We prove the lemma by induction on r. When $r = 1$ the lemma is trivially true; for completeness, we also directly analyze the case $r = 2$. Since $\mathcal{A}$ is unbounded we may assume it is deterministic. Then, without loss of generality, we may assume the adversary's strategy is determined by a set S in the support of D_2 such that $\mathcal{A}$ aborts in the first iteration iff $a_1 \in S$, and otherwise aborts in the second iteration (no matter what). We have

$$\begin{aligned}
\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins and } i^* = 1] + \Pr[\mathcal{A} \text{ wins and } i^* = 2] \\
&= \frac{1}{2} \cdot \Pr_{a \leftarrow D_2}[a \in S] + \frac{1}{2} \cdot \left(1 - \Pr_{a \leftarrow D_1}[a \in S]\right) \\
&\leq \frac{1}{2} \cdot \Pr_{a \leftarrow D_2}[a \in S] + \frac{1}{2} \cdot \left(1 - \alpha \cdot \Pr_{a \leftarrow D_2}[a \in S]\right) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \left((1 - \alpha) \cdot \Pr_{a \leftarrow D_2}[a \in S]\right) \;\leq\; 1 - \alpha/2,
\end{aligned}$$

where the first inequality is due to Equation (1). One can easily verify that $1 - \alpha/2 \leq 1/2\alpha$ when $\alpha > 0$. We have thus proved $\mathsf{Win}(\alpha, 2) \leq 1/2\alpha$.

Assume $\mathsf{Win}(\alpha, r) \leq 1/\alpha r$, and we now bound $\mathsf{Win}(\alpha, r+1)$. As above, let S denote a set in the support of D_2 such that $\mathcal{A}$ aborts in the first iteration iff $a_1 \in S$. If $\mathcal{A}$ does *not* abort in the first iteration, and the game does not end, then the conditional distribution of i^* is uniform in $\{2, \ldots, r+1\}$ and the game $\Gamma(\alpha, r+1)$ from this point forward is exactly equivalent to the game $\Gamma(\alpha, r)$. In particular, conditioned on the game $\Gamma(\alpha, r+1)$ not ending after the first iteration, the best strategy for $\mathcal{A}$ is to play whatever is the best strategy in game $\Gamma(\alpha, r)$. We thus have

$$\begin{aligned}
\Pr[\mathcal{A} \text{ wins}] &= \Pr[\mathcal{A} \text{ wins and } i^* = 1] + \Pr[\mathcal{A} \text{ wins and } i^* > 1] \\
&= \frac{1}{r+1} \cdot \Pr_{a \leftarrow D_2}[a \in S] + \frac{r}{r+1} \cdot \left(1 - \Pr_{a \leftarrow D_1}[a \in S]\right) \cdot \mathsf{Win}(\alpha, r) \\
&\leq \frac{1}{r+1} \cdot \Pr_{a \leftarrow D_2}[a \in S] + \frac{1}{\alpha(r+1)} \cdot \left(1 - \alpha \cdot \Pr_{a \leftarrow D_2}[a \in S]\right) \cdot \\
&= \frac{1}{\alpha(r+1)}.
\end{aligned}$$

This completes the proof. ∎

3.2 A $\frac{1}{p}$-Secure Protocol for Functions over Polynomial-Size Domains

In this section, we describe an approach that works for functions where at least one of the domains is polynomial-size. Although the protocol we describe would work even when the parties receive different outputs, for simplicity we assume here that the parties compute a single-output function. (We will return to the more general setting in the following section.) We prove the following:

Theorem 3 *Let $\mathcal{F} = \{f_n : X_n \times Y_n \to Z_n\}$ be a sequence of (randomized) functions where $|Y_n| =$ $\mathsf{poly}(n)$. Then, assuming the existence of enhanced trapdoor permutations, for any polynomial p there exists an $\mathcal{O}(p \cdot |Y_n|)$-round protocol that $\frac{1}{p}$-securely computes $\mathcal{F}$.*

Proof: As described in Section 1.3, our protocol Π consists of two stages. Let p be an arbitrary polynomial, and set $r = p \cdot |Y_n|$. We will implement the first stage using a sub-protocol for computing a randomized functionality $\mathsf{ShareGen}_r$ defined in Figure 1. ($\mathsf{ShareGen}_r$ is parameterized by a polynomial r.) This functionality returns shares to each party, authenticated using an information-theoretically secure r-time MAC, and in the second stage of the protocol the parties exchange these shares in a sequence of r iterations as described in Figure 2.

$\mathsf{ShareGen}_r$

Inputs: The security parameter is n. Let the inputs to $\mathsf{ShareGen}_r$ be $x \in X_n$ and $y \in Y_n$. (If one of the received inputs is not in the correct domain, a default input is substituted.)

Computation:

1. Define values $a_1, \ldots, a_r$ and $b_1, \ldots, b_r$ in the following way:
   - Choose i^* uniformly at random from $\{1, \ldots, r\}$.
   - For $i = 1$ to $i^* - 1$ do:
     - Choose $\hat{y} \leftarrow Y_n$ and set $a_i = f(x, \hat{y})$.
     - Choose $\hat{x} \leftarrow X_n$ and set $b_i = f(\hat{x}, y)$.
   - For $i = i^*$ to r, set $a_i = b_i = f(x, y)$.
2. For $1 \leq i \leq r$, choose $(a_i^{(1)}, a_i^{(2)})$ and $(b_i^{(1)}, b_i^{(2)})$ as random secret sharings of a_i and b_i, respectively. (E.g., $a_i^{(1)}$ is random and $a_i^{(1)} \oplus a_i^{(2)} = a_i$.)
3. Compute $k_a, k_b \leftarrow \mathsf{Gen}(1^n)$. For $1 \leq i \leq r$, let $t_i^a = \mathsf{Mac}_{k_a}(i \| a_i^{(2)})$ and $t_i^b = \mathsf{Mac}_{k_b}(i \| b_i^{(1)})$.

Output:

1. Send to P_1 the values $a_1^{(1)}, \ldots, a_r^{(1)}$ and $(b_1^{(1)}, t_1^b), \ldots, (b_r^{(1)}, t_r^b)$, and the MAC-key k_a.
2. Send to P_2 the values $(a_1^{(2)}, t_1^a), \ldots, (a_r^{(2)}, t_r^a)$ and $b_1^{(2)}, \ldots, b_r^{(2)}$, and the MAC-key k_b.

Figure 1: Functionality $\mathsf{ShareGen}_r$.

We analyze our protocol in a hybrid model where there is a trusted party computing $\mathsf{ShareGen}_r$. (In this ideal world, if P_1 is adversarial it can abort the trusted party computing $\mathsf{ShareGen}_r$ before the trusted party sends output to the honest party.) We will prove $\frac{1}{p}$-security of Π in this hybrid model (in fact, we will prove that an execution in the hybrid world has statistical difference at most $\frac{1}{p} + \mathsf{negl}(n)$ from an ideal-world computation of $\mathcal{F}$); it follows as in [7] that if we use a sub-protocol for computing $\mathsf{ShareGen}_r$ that is secure-with-abort, then Π is $\frac{1}{p}$-secure.

Claim 1 *For every non-uniform, polynomial-time adversary $\mathcal{A}$ corrupting P_1 and running Π in a hybrid model with access to an ideal functionality computing $\mathsf{ShareGen}_r$ (with abort), there exists a non-uniform, polynomial-time adversary $\mathcal{S}$ corrupting P_1 and running in the ideal world with access to an ideal functionality computing f_n (with complete fairness), such that*

$$\left\{\mathrm{IDEAL}_{f_n, \mathcal{S}(\mathsf{aux})}(x, y, n)\right\}_{x \in X_n, y \in Y_n, \mathsf{aux} \in \{0,1\}^*} \stackrel{1/p}{\approx} \left\{\mathrm{HYBRID}^{\mathsf{ShareGen}_r}_{\Pi, \mathcal{A}(\mathsf{aux})}(x, y, n)\right\}_{x \in X_n, y \in Y_n, \mathsf{aux} \in \{0,1\}^*}.$$

Proof: We construct a simulator $\mathcal{S}$ that is given black-box access to $\mathcal{A}$. *For readability in what follows, we ignore the presence of the MAC-tags and keys.* That is, we do not mention the fact that $\mathcal{S}$ computes MAC-tags for messages it gives to $\mathcal{A}$, nor do we mention the fact that $\mathcal{S}$ must verify the MAC-tags on the messages sent by $\mathcal{A}$. When we say that $\mathcal{A}$ "aborts", we include in this the event that $\mathcal{A}$ sends an invalid message, or a message whose tag does not pass verification. We also drop the subscript n from our notation.

1. $\mathcal{S}$ invokes $\mathcal{A}$ on the input[3] x', the auxiliary input, and the security parameter n. The simulator also chooses $\hat{x} \in X$ uniformly at random (it will send $\hat{x}$ to the trusted party, if needed).

[3]To simplify notation, we reserve x for the value input by $\mathcal{A}$ to the computation of $\mathsf{ShareGen}_r$.

Protocol 1

Inputs: Party P_1 has input x and party P_2 has input y. The security parameter is n. Let $r = p \cdot |Y_n|$.

The protocol:

1. **Preliminary phase:**
   (a) P_1 chooses $\hat{y} \in Y_n$ uniformly at random, and sets $a_0 = f(x, \hat{y})$. Similarly, P_2 chooses $\hat{x} \in X_n$ uniformly at random, and sets $b_0 = f(\hat{x}, y)$.
   (b) Parties P_1 and P_2 compute $\mathsf{ShareGen}_r$, using their inputs x and y.
   (c) If P_2 receives $\perp$ from the above computation, it outputs b_0 and halts. Otherwise, the parties proceed to the next step.
   (d) Denote the output of P_1 from π by $a_1^{(1)}, \ldots, a_r^{(1)}$, $(b_1^{(1)}, t_1^b), \ldots, (b_r^{(1)}, t_r^b)$, and k_a.
   (e) Denote the output of P_2 from π by $(a_1^{(2)}, t_1^a), \ldots, (a_r^{(2)}, t_r^a)$, $b_1^{(2)}, \ldots, b_r^{(2)}$, and k_b.
2. **For $i = 1, \ldots, r$ do:**

   P_2 sends the next share to P_1:
   (a) P_2 sends $(a_i^{(2)}, t_i^a)$ to P_1.
   (b) P_1 receives $(a_i^{(2)}, t_i^a)$ from P_2. If $\mathsf{Vrfy}_{k_a}(i \| a_i^{(2)}, t_i^a) = 0$ (or if P_1 received an invalid message, or no message), then P_1 outputs a_{i-1} and halts.
   (c) If $\mathsf{Vrfy}_{k_a}(i \| a_i^{(2)}, t_i^a) = 1$, then P_1 sets $a_i = a_i^{(1)} \oplus a_i^{(2)}$ (and continues running the protocol).

   P_1 sends the next share to P_2:
   (a) P_1 sends $(b_i^{(1)}, t_i^b)$ to P_2.
   (b) P_2 receives $(b_i^{(1)}, t_i^b)$ from P_1. If $\mathsf{Vrfy}_{k_b}(i \| b_i^{(1)}, t_i^b) = 0$ (or if P_2 received an invalid message, or no message), then P_2 outputs b_{i-1} and halts.
   (c) If $\mathsf{Vrfy}_{k_b}(i \| b_i^{(1)}, t_i^b) = 1$, then P_2 sets $b_i = b_i^{(1)} \oplus b_i^{(2)}$ (and continues running the protocol).
3. If all r iterations have been run, party P_1 outputs a_r and party P_2 outputs b_r.

Figure 2: Generic protocol for computing a function f_n.

2. $\mathcal{S}$ receives the input x of $\mathcal{A}$ to the computation of the functionality $\mathsf{ShareGen}_r$. (If $x \notin X$ a default input is substituted.)
3. $\mathcal{S}$ sets $r = p \cdot |Y|$, and chooses uniformly-distributed shares $a_1^{(1)}, \ldots, a_r^{(1)}$ and $b_1^{(1)}, \ldots, b_r^{(1)}$. Then, $\mathcal{S}$ gives these shares to $\mathcal{A}$ as its output from the computation of $\mathsf{ShareGen}_r$.
4. If $\mathcal{A}$ sends abort to the trusted party computing $\mathsf{ShareGen}_r$, then $\mathcal{S}$ sends $\hat{x}$ to the trusted party computing f, outputs whatever $\mathcal{A}$ outputs, and halts. Otherwise (i.e., if $\mathcal{A}$ sends $\mathsf{continue}$), $\mathcal{S}$ proceeds as below.
5. Choose i^* uniformly from $\{1, \ldots, r\}$
6. For $i = 1$ to $i^* - 1$:
   (a) $\mathcal{S}$ chooses $\hat{y} \in Y$ uniformly at random, computes $a_i = f(x, \hat{y})$, and sets $a_i^{(2)} = a_i^{(1)} \oplus a_i$. It gives $a_i^{(2)}$ to $\mathcal{A}$. (Note that a fresh $\hat{y}$ is chosen in every iteration.)
   (b) If $\mathcal{A}$ aborts, then $\mathcal{S}$ sends $\hat{x}$ to the trusted party, outputs whatever $\mathcal{A}$ outputs, and halts.

7. For $i = i^*$ to r:
   (a) If $i = i^*$ then $\mathcal{S}$ sends x to the trusted party computing f and receives $z = f(x, y)$.
   (b) $\mathcal{S}$ sets $a_i^{(2)} = a_i^{(1)} \oplus z$ and gives $a_i^{(2)}$ to $\mathcal{A}$.
   (c) If $\mathcal{A}$ aborts, then $\mathcal{S}$ then outputs whatever $\mathcal{A}$ outputs, and halts. If $\mathcal{A}$ does not abort, then $\mathcal{S}$ proceeds.
8. If $\mathcal{A}$ has never aborted (and all r iterations are done), then $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs and halts.

Ignoring the possibility of a MAC forgery, we claim that the statistical difference between an execution of $\mathcal{A}$, running Π in a hybrid world with access to an ideal functionality computing $\mathsf{ShareGen}_r$, and an execution of $\mathcal{S}$, running in an ideal world with access to an ideal functionality computing f, is at most $1/p$. (Thus, taking into account the possibility of a MAC forgery makes the statistical difference at most $1/p + \mu(n)$ for some negligible function μ.) To see this, let y denote the input of the honest P_2 and consider three cases depending on when the adversary aborts:

1. $\mathcal{A}$ aborts in round $i < i^*$. Conditioned on this event, the view of $\mathcal{A}$ is identically distributed in the two worlds (and is independent of y), and the output of the honest party is $f(\hat{x}, y)$ for $\hat{x}$ chosen uniformly in X.

2. $\mathcal{A}$ aborts in round $i > i^*$, or never aborts. Conditioned on this event, the view of $\mathcal{A}$ is again distributed identically in the two worlds, and in both worlds the output of the honest party is $f(x, y)$.

3. $\mathcal{A}$ aborts in round $i = i^*$: here the distributions are *not* identical in the two worlds. Although the view of $\mathcal{A}$ is identical in both worlds, the output of the honest party is not: in the hybrid world the honest party will output $f(\hat{x}, y)$, for $\hat{x}$ chosen uniformly in X, while in the ideal world the honest party will output $f(x, y)$.

   However, we use Lemma 1 to claim that this event occurs with probability at most $1/p$. To see this, let D_1 denote the distribution of a_i for $i < i^*$, and let D_2 denote the distribution of a_{i^*}. By construction of the protocol, we have

$$\begin{aligned} \Pr_{a \leftarrow D_1}[a = z] &\stackrel{\text{def}}{=} \Pr_{\hat{y} \leftarrow Y}[f(x, \hat{y}) = z] \\ &\geq \frac{1}{|Y|} \cdot \Pr[f(x, y) = z] \quad = \quad \frac{1}{|Y|} \cdot \Pr_{a \leftarrow D_2}[a = z]. \end{aligned}$$

   Taking $\alpha = 1/|Y|$ and applying Lemma 1, we see that $\mathcal{A}$ aborts in iteration i^* with probability at most $1/\alpha r = |Y|/|Y|p = 1/p$.

This completes the proof of the claim. ■

Claim 2 *For every non-uniform, polynomial-time adversary $\mathcal{A}$ corrupting P_2 and running Π in a hybrid model with access to an ideal functionality computing $\mathsf{ShareGen}_r$ (with abort), there exists a non-uniform, polynomial-time adversary $\mathcal{S}$ corrupting P_2 and running in the ideal world with access to an ideal functionality computing f (with complete fairness), such that*

$$\left\{\text{IDEAL}_{f,\mathcal{S}}(x, y, n)\right\}_{x \in X_n, y \in Y_n, n \in \mathbb{N}} \stackrel{\text{c}}{\equiv} \left\{\text{HYBRID}_{\Pi,\mathcal{A}}^{\mathsf{ShareGen}_r}(x, y, n)\right\}_{x \in X_n, y \in Y_n, n \in \mathbb{N}}.$$

$\mathsf{ShareGen}'_{p,r}$

Inputs: The security parameter is n. Let the inputs to $\mathsf{ShareGen}'_{p,r}$ be $x \in X_n$ and $y \in Y_n$. (If one of the received inputs is not in the correct domain, a default input is substituted.)

Computation:

1. Define values $a_1, \ldots, a_r$ and $b_1, \ldots, b_r$ in the following way:
   - Choose i^* uniformly at random from $\{1, \ldots, r\}$
   - For $i = 1$ to $i^* - 1$ do:
     - Choose $\hat{x} \leftarrow X$ and set $b_i = f^2(\hat{x}, y)$.
     - With probability $\frac{1}{p}$, choose $z \leftarrow Z_n^1$ and set $a_i = z$
     - With the remaining probability $1 - \frac{1}{p}$, choose $\hat{y} \leftarrow Y$ and set $a_i = f^1(x, \hat{y})$.
   - For $i = i^*$ to r, set $a_i = f^1(x, y)$ and $b_i = f^2(x, y)$.
2. For $1 \leq i \leq r$, choose $(a_i^{(1)}, a_i^{(2)})$ and $(b_i^{(1)}, b_i^{(2)})$ as random secret sharings of a_i and b_i, respectively. (E.g., $a_i^{(1)}$ is random and $a_i^{(1)} \oplus a_i^{(2)} = a_i$.)
3. Compute $k_a, k_b \leftarrow \mathsf{Gen}(1^n)$. For $1 \leq i \leq r$, let $t_i^a = \mathsf{Mac}_{k_a}(i \| a_i^{(2)})$ and $t_i^b = \mathsf{Mac}_{k_b}(i \| b_i^{(1)})$.

Output:

1. Send to P_1 the values $a_1^{(1)}, \ldots, a_r^{(1)}$ and $(b_1^{(1)}, t_1^b), \ldots, (b_r^{(1)}, t_r^b)$, and the MAC-key k_a.
2. Send to P_2 the values $(a_1^{(2)}, t_1^a), \ldots, (a_r^{(2)}, t_r^a)$ and $b_1^{(2)}, \ldots, b_r^{(2)}$, and the MAC-key k_b.

Figure 3: Functionality $\mathsf{ShareGen}'_{p,r}$.

The simulation is identical to the one in Claim 1, though the proof is much simpler (and we can prove a stronger notion of security) since P_1 always "gets the output first" in every iteration. ■

Achieving security-with-abort. As written, the protocol is not secure-with-abort. However, it is easy to modify it so that it is (without affecting $\frac{1}{p}$-security): simply have $\mathsf{ShareGen}_r$ set $b_{i^*-1} = \perp$, where $\perp$ is some distinguished value outside the range of f. Although this allows a malicious P_2 to identify exactly when iteration i^* occurs, this does not affect security (as reflected by Claim 2).

3.3 A $\frac{1}{\mathbf{p}}$-Secure Protocol for Functions over Arbitrary Domains

The protocol from the previous section does not directly apply to functions on arbitrary (i.e., not necessarily polynomial-size) domains, since the round complexity of the protocol is polynomial in the smaller domain. In this section we demonstrate how to extend the protocol so as to handle arbitrary domains, as long as the range of the function is polynomial size (for at least one of the parties). For completeness, we now also explicitly take into account the case when parties obtain different outputs. Intuition for the changes we introduce is given in Section 1.3.

Theorem 4 *Let $\mathcal{F} = \{f_n : X_n \times Y_n \to Z_n^1 \times Z_n^2\}$ be a sequence of (randomized) functions, with $|Z_n^1| = \mathsf{poly}(n)$. Then, assuming the existence of enhanced trapdoor permutations, for any polynomial p there exists an $\mathcal{O}\left(p^2 \cdot |Z_n^1|\right)$-round protocol that $\frac{1}{p}$-securely computes $\mathcal{F}$.*

Proof: Our protocol Π is, once again, composed of two stages. The second stage will be identical to the second stage of the previous protocol (see Figure 2), except that the number of iterations r

will now be set to $r = p^2 \cdot |Z_n^1|$. The first stage will generate shares using a new sub-routine $\mathsf{ShareGen}'_{p,r}$, parameterized by both p and r, as described in Figure 3.

We will again analyze our protocol in a hybrid model, but where there is now a trusted party computing $\mathsf{ShareGen}'_{p,r}$. (Once again, P_1 can abort the computation of $\mathsf{ShareGen}'_{p,r}$ even in the ideal world.) We will prove $\frac{1}{p}$-security in this hybrid model, which implies that if the parties use a secure-with-abort protocol for computing $\mathsf{ShareGen}'_{p,r}$ then the entire protocol Π is $\frac{1}{p}$-secure.

Claim 3 *For every non-uniform, polynomial-time adversary $\mathcal{A}$ corrupting P_1 and running Π in a hybrid model with access to an ideal functionality computing $\mathsf{ShareGen}'_{p,r}$ (with abort), there exists a non-uniform, polynomial-time adversary $\mathcal{S}$ corrupting P_1 and running in the ideal world with access to an ideal functionality computing $\mathcal{F}$ (with complete fairness), such that*

$$\left\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x,y,n)\right\}_{x\in X_n, y\in Y_n, \mathsf{aux}\in\{0,1\}^*} \stackrel{1/p}{\approx} \left\{\mathrm{HYBRID}^{\mathsf{ShareGen}'_{p,r}}_{\Pi,\mathcal{A}(\mathsf{aux})}(x,y,n)\right\}_{x\in X_n, y\in Y_n, \mathsf{aux}\in\{0,1\}^*}.$$

Proof: The simulator is essentially the same as the simulator used in the proof of Claim 1, except that in step 6(a) the distribution on a_i (for $i < i^*$) is changed to the one used by $\mathsf{ShareGen}'_{p,r}$. The analysis is similar, too, except for bounding the probability that $\mathcal{A}$ aborts in iteration i^*. To bound this probability we will again rely on Lemma 1, but now distribution D_1 (i.e., the distribution of a_i for $i < i^*$) is different. Let y denote the input of P_2. Note that, by construction of $\mathsf{ShareGen}'_{p,r}$, for any $z \in Z_n^1$ we have $\Pr_{a\leftarrow D_1}[a = z] \geq \frac{1}{p} \cdot \frac{1}{|Z_n^1|}$. Regardless of f^1 and y, it therefore holds for all $z \in Z_n^1$ that

$$\Pr_{a\leftarrow D_1}[a = z] \geq \frac{1}{p \cdot |Z_n^1|} \cdot \Pr_{a\leftarrow D_2}[a = z].$$

Setting $\alpha = 1/p \cdot |Z_n^1|$ and applying Lemma 1, we see that $\mathcal{A}$ aborts in iteration i^* with probability at most

$$\frac{1}{\alpha r} = \frac{p \cdot |Z_n^1|}{p^2 \cdot |Z_n^1|} = \frac{1}{p}.$$

This completes the proof. ■

The following claim considers the case of a malicious P_2. We stress that, in contrast to Claim 2, here we claim only $\frac{1}{p}$-indistinguishability.

Claim 4 *For every non-uniform, polynomial-time adversary $\mathcal{A}$ corrupting P_2 and running Π in a hybrid model with access to an ideal functionality computing $\mathsf{ShareGen}'_{p,r}$ (with abort), there exists a non-uniform, polynomial-time adversary $\mathcal{S}$ corrupting P_2 and running in the ideal world with access to an ideal functionality computing $\mathcal{F}$ (with complete fairness), such that*

$$\left\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x,y,n)\right\}_{x\in X_n, y\in Y_n, \mathsf{aux}\in\{0,1\}^*} \stackrel{1/p}{\approx} \left\{\mathrm{HYBRID}^{\mathsf{ShareGen}'_{p,r}}_{\Pi,\mathcal{A}(\mathsf{aux})}(x,y,n)\right\}_{x\in X_n, y\in Y_n, \mathsf{aux}\in\{0,1\}^*}.$$

Proof: The simulator $\mathcal{S}$ in this case is fairly obvious, but we include it for completeness. Once again, for readability we ignore the presence of the MAC-tags and keys.

1. $\mathcal{S}$ invokes $\mathcal{A}$ on the input y', the auxiliary input, and the security parameter n. The simulator also chooses $\hat{y} \in Y$ uniformly at random (it will send $\hat{y}$ to the trusted party, if needed).

2. $\mathcal{S}$ receives the input y of $\mathcal{A}$ to the computation of the functionality $\mathsf{ShareGen}'_{p,r}$. (If $y \notin Y$ a default input is substituted.)

3. $\mathcal{S}$ sets $r = p^2 \cdot |Z^1|$, and chooses uniformly-distributed shares $a_1^{(1)}, \ldots, a_r^{(1)}$ and $b_1^{(1)}, \ldots, b_r^{(1)}$. Then, $\mathcal{S}$ gives these shares to $\mathcal{A}$ as its output from the computation of $\mathsf{ShareGen}'_{p,r}$.

4. Choose i^* uniformly from $\{1, \ldots, r\}$

5. For $i = 1$ to $i^* - 1$:

   (a) $\mathcal{S}$ chooses $\hat{x} \in X$ uniformly at random, computes $b_i = f^2(\hat{x}, y)$, and sets $b_i^{(1)} = b_i^{(2)} \oplus b_i$. It gives $b_i^{(1)}$ to $\mathcal{A}$. (Note that a fresh $\hat{x}$ is chosen in every iteration.)

   (b) If $\mathcal{A}$ aborts, then $\mathcal{S}$ sends $\hat{y}$ to the trusted party, outputs whatever $\mathcal{A}$ outputs, and halts.

6. For $i = i^*$ to r:

   (a) If $i = i^*$ then $\mathcal{S}$ sends y to the trusted party computing f and receives $z = f^2(x, y)$.

   (b) $\mathcal{S}$ sets $b_i^{(1)} = b_i^{(2)} \oplus z$ and gives $b_i^{(1)}$ to $\mathcal{A}$.

   (c) If $\mathcal{A}$ aborts, then $\mathcal{S}$ then outputs whatever $\mathcal{A}$ outputs, and halts. If $\mathcal{A}$ does not abort, then $\mathcal{S}$ proceeds.

7. If $\mathcal{A}$ has never aborted (and all r iterations are done), then $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs and halts.

Ignoring the possibility of a MAC forgery, we claim that the statistical difference between an execution of $\mathcal{A}$, running Π in a hybrid world with access to an ideal functionality computing $\mathsf{ShareGen}'_{p,r}$, and an execution of $\mathcal{S}$, running in an ideal world with access to an ideal functionality computing $\mathcal{F}$, is at most $1/p$. (Thus, taking into account the possibility of a MAC forgery makes the statistical difference at most $1/p + \mu(n)$ for some negligible function μ.) In fact, the view of $\mathcal{A}$ is identical in the two worlds; the only issue is the output of the honest P_1 holding input x. Specifically, if $\mathcal{A}$ aborts in any iteration prior to i^* then, in the ideal world interaction with $\mathcal{S}$, party P_1 outputs $f^1(x, \hat{y})$ for a uniformly-chosen $\hat{y} \in Y$. In the hybrid world, however, the output of P_1 is given by the distribution of a_i (for $i < i^*$) as determined by $\mathsf{ShareGen}'_{p,r}$. However, these two distributions are within statistical difference (at most) $1/p$. The claim follows. ■

■

4 Optimality of Our Results

In this section, we prove two impossibility results showing that the results of the previous section are optimal as far as generic feasibility results go. The first result is unconditional, while the second relies on cryptographic assumptions.

4.1 Impossibility of $\frac{1}{p}$-Security and Security-with-Abort for Functions over Arbitrary Domains

As mentioned earlier, the notions of security-with-abort and $\frac{1}{p}$-security are incomparable. Although we are able to achieve $\frac{1}{p}$-security and security-with-abort *simultaneously* for the case of functionalities where at least one of the domains is polynomial-size (cf. Section 3.2), we were not able to do so for the case of arbitrary domains. We show, in fact, that it is impossible to achieve both of these criteria (in general) in this case.

Consider the equality function EQ defined over domains of super-polynomial size. (Formally, consider $\mathsf{EQ} : X_n \times Y_n \to \{0,1\}$ where $X_n = Y_n = \{0,1\}^{\ell(n)}$ for some $\ell(n) = \omega(\log n)$.) Let Π be some protocol computing EQ, where we assume without loss of generality that P_2 goes first and P_1 goes last. Say Π has $r = r(n)$ iterations for some polynomial r. (An iteration consists of a message sent by P_2 followed by a message from P_1.) We show that Π cannot be simultaneously secure-with-abort and $\frac{1}{p}$-secure for any $p > 4 + \frac{1}{\mathsf{poly}(n)}$.

Let a_0 denote the value that P_1 outputs if P_2 sends nothing, and let a_i, for $1 \le i \le r$, denote the value that P_1 outputs if P_2 aborts after sending its iteration-i message. Similarly, let b_0 denote the value that P_2 outputs if P_1 sends nothing, and let b_i, for $1 \le i \le r$, denote the value that P_2 outputs if P_1 aborts after sending its iteration-i message. We may assume without loss of generality that, for all i, $a_i \in \{0,1\}$ and $b_i \in \{0,1,\perp\}$.

We will consider two experiments involving an execution of Π. In the first, x and y are chosen uniformly and independently from $\{0,1\}^{\ell(n)}$; the parties are given inputs x and y, respectively; and the parties then run protocol Π honestly. We denote the probability of events in this probability space by $\Pr_{\mathrm{rand}}[\cdot]$. In the second experiment, x is chosen uniformly from $\{0,1\}^{\ell(n)}$ and y is set equal to x; these inputs are given to the parties and they run the protocol honestly as before. We denote the probability of events in this probability space by $\Pr_{\mathrm{eq}}[\cdot]$.

Claim 5 *If Π is secure-with-abort, then $\Pr_{\mathrm{rand}}[a_0 = 1 \vee \cdots \vee a_r = 1]$ and $\Pr_{\mathrm{rand}}[b_0 = 1 \vee \cdots \vee b_r = 1]$ are negligible.*

Proof: If, say, $\Pr_{\mathrm{rand}}[a_0 = 1 \vee \cdots \vee a_r = 1]$ were not negligible, then we could consider an adversarial P_2 that runs the protocol honestly but aborts at a random round. This would result in the honest P_1 outputting 1 with non-negligible probability in the real world, whereas this occurs with only negligible probability in the ideal world (when the parties are given independent, random inputs). A similar argument applies to $\Pr_{\mathrm{rand}}[b_0 = 1 \vee \cdots \vee b_r = 1]$. ■

Assume for simplicity that Π has perfect correctness, i.e., that $a_r = b_r = \mathsf{EQ}(x,y)$ when the two parties run the protocol honestly holding initial inputs x and y. (This assumption is not necessary, but allows us to avoid introducing burdensome notation.) Then $\Pr_{\mathrm{eq}}[a_0 = 1 \vee \cdots \vee a_r = 1] = \Pr_{\mathrm{eq}}[b_0 = 1 \vee \cdots \vee b_r = 1] = 1$. Let i^* denote the lowest index, if any, for which $a_{i^*} = 1$. Similarly, let j^* denote the lowest index, if any, for which $b_{j^*} = 1$. Since

$$\Pr_{\mathrm{eq}}[i^* \le j^*] + \Pr_{\mathrm{eq}}[i^* > j^*] = \Pr_{\mathrm{eq}}[i^* \text{ and } j^* \text{ are defined}] = 1,$$

at least one of the two terms on the left-hand side is at least 1/2. We assume $\Pr_{\mathrm{eq}}[i^* \le j^*] \ge 1/2$ in what follows, but the exact same argument (swapping the roles of the parties) applies in case $\Pr_{\mathrm{eq}}[i^* > j^*] \ge 1/2$.

Consider now a third experiment that is a mixture of the previous two. Specifically, in this experiment a random bit b is chosen; if $b = 0$ then the parties are given inputs x and y as in the first experiment (i.e., chosen uniformly and independently at random), while if $b = 1$ then the parties are given (random) $x = y$ as in the second experiment. The parties then run protocol Π honestly. We denote the probability of events in this probability space by $\Pr_3^{\mathsf{real}}[\cdot]$. We use the superscript real to distinguish this from an ideal-world version of this experiment where the bit b is chosen uniformly and the parties are given x and y generated accordingly, but now the parties interact with an ideal party computing EQ *without abort*. We denote the probability of events in this probability space by $\Pr_3^{\mathsf{ideal}}[\cdot]$.

Consider an execution of the third experiment (in either the real or ideal worlds), in the case when P_1 is malicious. Let guess denote the event that P_1 correctly guesses the value of the bit b, and let out_2 denote the output of P_2. A simple calculation shows that

$$\Pr_3^{\mathsf{ideal}}[\mathsf{guess} \wedge \mathsf{out}_2 \neq 1] \leq \frac{1}{2} + \mathsf{negl}(n). \tag{2}$$

Now take the following real-world adversary $\mathcal{A}$ corrupting P_1: upon receiving input x, run Π honestly but compute a_i after receiving each iteration-i message from P_2. There are two cases:

- If, at some point, some $a_i = 1$ then abort the protocol (before sending the iteration-i message on behalf of P_1) and output the guess "$b = 1$".
- If $a_i = 0$ for all i, then simply run the protocol to the end (including the final message of the protocol) and output the guess "$b = 0$".

We have:

$$\begin{aligned}
\Pr_3^{\mathsf{real}}[\mathsf{guess} \wedge \mathsf{out}_2 \neq 1] &= \frac{1}{2} \cdot \Pr_{\mathrm{rand}}[\mathsf{guess} \wedge \mathsf{out}_2 \neq 1] + \frac{1}{2} \cdot \Pr_{\mathrm{eq}}[\mathsf{guess} \wedge \mathsf{out}_2 \neq 1] \\
&\geq \frac{1}{2} \cdot \Pr_{\mathrm{rand}}[a_1 = 0 \wedge \cdots \wedge a_r = 0 \wedge b_r = 0] + \frac{1}{2} \cdot \Pr_{\mathrm{eq}}[i^* \leq j^*] \\
&\geq \frac{1}{2} \cdot (1 - \mathsf{negl}(n)) + \frac{1}{4} \;=\; \frac{3}{4} - \mathsf{negl}(n).
\end{aligned} \tag{3}$$

Taken together, Claim 5 and Equations (2) and (3) show that if Π is secure with abort, then it cannot also be $\frac{1}{p}$-secure for any $p > 4 + \frac{1}{\mathsf{poly}(n)}$.

4.2 Impossibility of $\frac{1}{p}$-Security for General Functions

Our results in Section 3.2 and 3.3 imply that $\frac{1}{p}$-security is achievable for any function $f : X \times Y \to Z^1 \times Z^2$ as long as at least one of X, Y, Z^1, Z^2 are polynomial size. Here, we give evidence that this limitation is inherent by showing, under a reasonable cryptographic assumption, that there is a deterministic, single-output function $f : X \times Y \to Z$ with $|X|, |Y|, |Z| = \omega(\mathsf{poly}(n))$ that cannot be $\frac{1}{p}$-securely computed for any $p > 2 + \frac{1}{\mathsf{poly}(n)}$. The proof is not difficult, and we would not be surprised if similar proofs have appeared previously in the literature; however, we were unable to track down any such references.

The assumption we will need is that exponentially-strong one-way functions exist. (We note, however, that our proof rules out partial fairness for functions over domains of size 2^{n^ϵ} if standard one-way functions exist.) This is a family of functions $\{f_n : \{0,1\}^n \to \{0,1\}^n\}$ such that, for some δ, we have

$$\Pr[\mathcal{A}(f(x)) \in f^{-1}(f(x))] \leq 2^{-\delta n},$$

for all $\mathcal{A}$ running in time $2^{\delta n}$. This implies that f is one-way (in the standard sense) when its input is chosen from $\{0,1\}^{\omega(\log n)}$.

Given such an f, we define the function

$$\mathsf{Swap} : \left(\{0,1\}^{\omega(\log n)}\right)^2 \times \left(\{0,1\}^{\omega(\log n)}\right)^2 \to \left(\{0,1\}^{\omega(\log n)}\right)^2$$

as follows: $\mathsf{Swap}((x_1, y_2), (x_2, y_1))$ outputs (x_1, x_2) iff $f(x_1) = y_1$ and $f(x_2) = y_2$, and outputs $\perp$ otherwise.

Consider an ideal-world computation of Swap, where x_1, x_2 are chosen uniformly at random, $y_1 = f(x_1)$ and $y_2 = f(x_2)$, and P_1 is given (x_1, y_2) while P_2 is given (x_2, y_1). We can easily observe that, e.g., an adversarial P_1 cannot simultaneously output an inverse of y_2 while causing P_2 to *fail* to output a valid inverse of y_1, except with negligible probability. In what follows, we refer to this event as a *win* for P_1.

In any real-world computation of Swap, however, there must be one party who "gets its output first" with probability at least 1/2. More formally, say we have an r-iteration protocol computing Swap where P_2 sends the first message and P_1 sends the last message, and let a_i, for $i = 0, \ldots, a_r$, denote the second component of the value P_1 would output if P_2 aborts the protocol after sending its iteration-i message. Similarly, let b_i denote the first component of the value that P_2 would output if P_1 aborts the protocol after sending its iteration-i message. Each value a_i and b_i can be computed in polynomial time after receiving the other party's iteration-i message. We can therefore define an adversary P_1^* that acts as follows:

Run the protocol honestly until the first round in which $f(a_i) = y_2$; then abort.

An adversary P_2^* can be defined analogously. Let i_1 be a random variable denoting the first round in which $f(a_i) = y_2$, and let i_2 denote the first round in which $f(b_i) = y_1$. Since

$$\Pr[i_1 \leq i_2] + \Pr[i_1 > i_2] = 1,$$

while $\Pr[P_1^* \text{ wins}] = \Pr[i_1 \leq i_2]$ and $\Pr[P_2^* \text{ wins}] = \Pr[i_1 > i_2]$, it is clear that either P_1^* or P_2^* wins with probability at least 1/2. Since an adversary wins in the ideal world with negligible probability, this rules out $\frac{1}{p}$-security if $\frac{1}{2} - \frac{1}{p}$ is noticeable.

The above does not contradict the result of [11], or any other work on partial fairness that aims to solve exactly this problem. The reason is that in previous work on partial fairness *the running time of the honest parties is not bounded by a fixed polynomial*, whereas in our setting we require this to be the case. We add further that in previous work on partial fairness (this is made explicit in [16]), the decision of how much "effort" honest parties should invest is *not* determined by the protocol, but instead is supposed to be determined is some other (unspecified) manner.

Acknowledgments

The second author would like to thank Yehuda Lindell for many enlightening discussions on this topic, one of which prompted the inclusion of Section 4.1. We thank the authors of [24] for sharing a copy of their manuscript with us.

References

[1] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *4th Theory of Cryptography Conference (TCC)*, volume 4392 of *Lecture Notes in Computer Science*, pages 137–156. Springer-Verlag, 2007.

[2] D. Beaver. Foundations of secure interactive computing. In *Advances in Cryptology — Crypto '91*, volume 576 of *Lecture Notes in Computer Science*, pages 377–391. Springer, 1992.

[3] D. Beaver and S. Goldwasser. Multiparty computation with faulty majority. In *Proc. 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 468–473, 1989.

[4] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. *IEEE Trans. Information Theory*, 36(1):40–46, 1990.

[5] M. Blum. How to exchange (secret) keys. *ACM Trans. Computer Systems*, 1(2):175–193, 1983.

[6] D. Boneh and M. Naor. Timed commitments. In *Advances in Cryptology — Crypto 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254. Springer, 2000.

[7] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[8] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145, 2001.

[9] R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *Proc. 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–369, 1986.

[10] R. Cleve. Controlled gradual disclosure schemes for random bits and their applications. In *Advances in Cryptology — Crypto '89*, volume 435 of *Lecture Notes in Computer Science*, pages 573–588. Springer, 1990.

[11] I. Damgård. Practical and provably secure release of a secret and exchange of signatures. *J. Cryptology*, 8(4):201–222, 1995.

[12] C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 409–418, 1998.

[13] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Comm. ACM*, 28(6):637–647, 1985.

[14] M. Franklin. *Complexity and Security of Distributed Protocols*. PhD thesis, Columbia University, 1993.

[15] Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In *Advances in Cryptology — Crypto '87*, volume 293 of *Lecture Notes in Computer Science*, pages 135–155. Springer, 1988.

[16] J. Garay, P. MacKenzie, M. Prabhakaran, and K. Yang. Resource fairness and composability of cryptographic protocols. In *3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 404–428. Springer-Verlag, 2006.

[17] O. Goldreich. *Foundations of Cryptography, Volume 2 – Basic Applications*. Cambridge University Press, 2004.

[18] O. Goldreich and Y. Lindell. Session-key generation using human passwords only. *J. Cryptology*, 19(3):241–340, 2006.

[19] S. Goldwasser and L. Levin. Fair computation and general functions in the presence of immoral majority. In *Advances in Cryptology — Crypto '90*, volume 537 of *Lecture Notes in Computer Science*. Springer, 1991.

[20] D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. In *Proc. 40th Annual ACM Symposium on Theory of Computing (STOC)*, 2008.

[21] J. Katz. On achieving the "best of both worlds" in secure multiparty computation. In *Proc. 48th Annual Symposium on Foundations of Computer Science (FOCS)*, 2007.

[22] M. Luby, S. Micali, and C. Rackoff. How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin. In *Proc. 24th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 23–30, 1983.

[23] S. Micali and P. Rogaway. Secure computation. In *Advances in Cryptology — Crypto '91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer, 1992.

[24] T. Moran, M. Naor, and G. Segev. An optimally fair coin toss: Cleve's bound is tight. Manuscript available from the authors.

[25] B. Pinkas. Fair secure two-party computation. In *Advances in Cryptology — Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 87–105. Springer, 2003.

[26] A. Yao. Protocols for secure computation. In *Proc. 27th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 162–167, 1986.