Partial Fairness in Secure Two-Party Computation

S. Dov Gordon[1] and Jonathan Katz[1]*

Department of Computer Science, University of Maryland
{gordon,jkatz}@cs.umd.edu

Abstract. A seminal result of Cleve (STOC '86) is that *complete* fairness is impossible to achieve in two-party computation. In light of this, various techniques for obtaining *partial* fairness have been suggested in the literature. We propose a definition of partial fairness within the standard real-/ideal-world paradigm that addresses deficiencies of prior definitions. We also show broad feasibility results with respect to our definition: partial fairness is possible for any (randomized) functionality $f : X \times Y \to Z_1 \times Z_2$ at least one of whose domains or ranges is polynomial in size. Our protocols are always private, and when one of the domains has polynomial size our protocols also simultaneously achieve the usual notion of security with abort. In contrast to some prior work, we rely on standard assumptions only.

We also show that, as far as general feasibility is concerned, our results are *optimal* (with respect to our definition).

1 Introduction

In the setting of secure two-party computation, two parties run a protocol that enables each of them to learn a (possibly different) function of their inputs while preserving security properties such as privacy, correctness, input independence, etc. These requirements, and more, are traditionally 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 protocol is "secure" if for any real-world adversary $\mathcal{A}$ there is a corresponding ideal-world adversary $\mathcal{S}$ (corrupting the same party) such that an execution of the protocol in the real world with $\mathcal{A}$ is computationally indistinguishable from computing the function in the ideal world with $\mathcal{S}$.

One desirable security property is *fairness* which, intuitively, ensures that either *both parties* learn the output or else *neither party* does. In a "true" ideal world — this is the ideal world used in the multi-party setting when a majority of parties are honest — fairness is ensured since the trusted party evaluating the function provides output to both parties. Unfortunately, Cleve [10] shows 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 (see [18])

* Research supported by NSF CAREER award #0447075 and NSF-CCF #0830464.

weakens the ideal world to one in which fairness is not guaranteed at all. A protocol is said to be "secure-with-abort" if it can be simulated (as described above) with respect to this less-satisfying ideal world.

Various methods for achieving *partial* fairness have been suggested; we provide an extensive discussion in Section 1.1. With the exception of [17], however, all previous work has departed from the traditional real-/ideal- world paradigm in defining partial fairness. (Indeed, addressing this deficiency is explicitly mentioned as an open problem by Goldreich [18, Section 7.7.1.1].) Furthermore, many previously suggested approaches to partial fairness only apply in specific settings (e.g., fair exchange of signatures) or under certain assumptions on the parties' inputs and auxiliary information (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. Finally, much previous work on partial fairness requires strong cryptographic assumptions, e.g., regarding the precise amount of time needed to solve some problem (even using parallelism).

As noted earlier, the most desirable (but, in the two-party setting, unachievable) definition of security requires computational indistinguishability between the real world and a "true" ideal world where both parties receive output. The usual relaxation of security-with-abort [18] leaves unchanged the requirement of computational indistinguishability, but weakens the ideal world to one in which fairness is no longer guaranteed at all. Katz [23] 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 $\frac{1}{p} + \mathsf{negl}$, where p is some specified polynomial[1] (see Definition 1). We refer to a protocol satisfying this definition as being "$\frac{1}{p}$-secure". Cleve [10] and Moran et al. [27] show $\frac{1}{p}$-secure protocols for two-party coin tossing (where parties have no inputs), but we are not aware of any other results satisfying our definition. In particular, none of the prior approaches for achieving partial fairness yield protocols that are $\frac{1}{p}$-secure.

We propose the notion of $\frac{1}{p}$-security as a new way to approach the problem of partial fairness, and view this as an independent contribution. We also demonstrate protocols that achieve this definition for a broad class of functions. 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.) For arbitrary polynomial p, we show $\frac{1}{p}$-secure protocols 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). Our protocols are always *private*, and when either X_n or Y_n is polynomial-size we also achieve the usual notion of security-with-abort. (Relevant definitions are stan-

[1] This definition is similar in spirit to (but weaker than) the notion of ϵ-zero knowledge [13] and is analogous to the definition used in [19] for password-based key exchange (although there p is fixed by the size of the password dictionary). A similar idea, formalized differently and with different motivation, is also used in [1].

dard and appear in the full version of this paper.) We assume only the existence of enhanced trapdoor permutations or, more generally, oblivious transfer.

We also prove that our feasibility results are, in general, *optimal.* First, we demonstrate a deterministic, boolean function $f_n : X_n \times Y_n \to \{0,1\}$, where X_n and Y_n both have super-polynomial size, for which no protocol computing f_n can simultaneously achieve both security-with-abort and $\frac{1}{p}$-security (for $p > 4$). We also show 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_n cannot be $\frac{1}{p}$-securely computed for $p > 2$.

1.1 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 [10, 11, 27] and contract signing/exchanging secrets [5, 25, 14, 4, 12, 6], as well as for the case of general functionalities [29, 16, 3, 20, 15, 28, 17]. Prior work (with the exception of [17]; see below), however, does not consider a *simulation-based* definition within the standard real/ideal world paradigm as we do here. Moreover, to the best of our knowledge none of the previous approaches (with the exception of [10, 27], that deal only with coin tossing) can be proven $\frac{1}{p}$-secure. Beyond the theoretical advantages of achieving a simulation-based notion of security, our protocols offer several concrete benefits with respect to prior solutions; these are explained in what follows.

One approach that has been suggested for achieving partial fairness is to construct a protocol where, roughly speaking, at every round both parties can recover their output using a "similar" amount of work (except in early rounds, where one party can recover their output only by investing exponential work). This idea was used in [16, 12, 6, 28], and was formalized by Garay et al. [17] within the framework of universal composability [9]. An unsatisfying feature of this approach, no matter how it is implemented, is that the decision of whether an honest party should invest the necessary work and recover the output is not determined by the protocol, but is somehow decided "externally"; if the adversary knows how this decision is made, then it can abort at "exactly the right time" and violate fairness completely. In this approach there may also be no *a priori* polynomial bound on the honest party's running time. This 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, this technique appears to inherently require strong assumptions regarding the precise time required to solve some specific computational problem.

A second approach, used in, e.g., [25] for exchanging secrets and in [3, 20] for computation of general functions, gradually increases each party's confidence in their output by, roughly speaking, masking the correct output with "noise" that tends to 0 as the protocol progresses. Protocols of this sort are inapplicable when the adversary has auxiliary information about the output of the function, since in that case the adversary's "confidence" at any point in the protocol is impossible to estimate. More problematic is that an adversary can *bias* the output of the honest party beyond what is possible in the ideal world. As a simple illustration, consider a computation of the equality function where each party holds a value chosen uniformly from some domain D. In the ideal world, the probability that an adversary can cause the honest player to output 1 is exactly $1/|D|$. Using the approach of [3, 20], however, the adversary can cause the honest player to output 1 with probability essentially $1/2$ by aborting in the first round (when the true answer is masked by an almost uniform random bit). Besides indicating a weakness of previous protocols, this example also demonstrates the importance of defining partial fairness within the simulation paradigm.

Gordon et al. [22] recently showed that *complete* fairness is possible in the two-party setting for certain specific functions. Work continuing that direction is 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 [10] and so some relaxation must be considered (at least for some functions). Our feasibility results here apply to a much richer class of functions.

Other work has looked at achieving complete fairness with off-line trusted third parties (e.g., [7]) or in non-standard communication models (e.g., [24]). We work in the standard communication model, and without any trusted parties.

1.2 Overview of our Approach

We now give an informal description of our feasibility results (details are in Section 3). 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 omit the dependence of X, Y, and Z on n, and focus on the case where each party receives the same output.) As in [23, 22, 27], 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. This consists of the following steps:

1. A value $i^* \in \{1, \ldots, r\}$ is chosen according to some distribution (see below). This represents the iteration in which the parties will learn the "true output".
2. Values $a_1, \ldots, a_r$ and $b_1, \ldots, b_r$ are generated. For $i < i^*$, the $\{a_i\}$ (resp., $\{b_i\}$) are chosen (independently) 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 $a_1^{(2)}$, $b_1^{(2)}$, $\ldots$, $a_r^{(2)}, b_r^{(2)}$. (Shares are also authenticated with an information-theoretic MAC.)

After this stage, each party has a set of random shares that reveal nothing about the other party's input. This stage can thus be carried out by any 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_2 who reconstructs b_i. (Parties also verify validity of the MAC but we omit this here.) If a party (say, P_1) aborts 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, after reaching iteration r the parties output a_r and b_r, respectively.

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 [23, 27], we choose i^* uniformly from $\{1, \ldots, r\}$. (In [22] a geometric distribution was used. That would work here, but with slightly worse round complexity.) When X and Y (the domains of f) are polynomial size, we follow [22] 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 because fairness is only violated if P_1 aborts *exactly* in iteration i^*. (If P_1 aborts before iteration i^* then neither party learns the "correct" value $z = f(x, y)$, while if it aborts subsequently then both parties learn the correct value. An abort by P_2 in iteration i^* does not violate fairness, since by then P_1 has already learned the output.) We show that *even if* P_1 *knows the value of* z (which it may, depending on partial information P_1 has about y), it cannot determine with certainty when iteration i^* occurs. Specifically, we prove a general result (see Lemma 1) implying (roughly) that as long as $\Pr[a_i = z] \geq \alpha$ for all $i < i^*$, then P_1 cannot abort in iteration i^* except with probability at most $1/\alpha r$ (recall that r is the number of iterations in the second phase). Since $\Pr[a_i = f(x, y)] = \Pr_{\hat{y} \in Y}[f(x, \hat{y}) = f(x, y)] \geq \Pr_{\hat{y} \in Y}[\hat{y} = y] = 1/|Y|$ for any x, y, we conclude that setting $r = p \cdot |Y|$, so that $1/\alpha r = 1/p$, suffices to achieve $\frac{1}{p}$-security. We thus get a protocol with polynomially many rounds as long as Y is polynomial size.

The above does not work directly when Y has super-polynomial size. To fix this, we must ensure that for every possible $z \in Z$ (the range of f) we have that $\Pr[a_i = z]$ is noticeable. We do this by changing the distribution of a_i (for $i < i^*$) as follows: with probability $1 - 1/q$ choose a_i as above, but with probability $1/q$ choose a_i uniformly from Z. Now, for any f, x, y, we have $\Pr[a_i = f(x, y)] \geq \frac{1}{q} \cdot \Pr_{a_i \in Z}[a_i = f(x, y)] \geq 1/q|Z|$ and so setting $r = pq|Z|$ ensures that P_1 cannot abort in iteration i^* except with probability at most $1/p$.

Changing the distribution of a_i, however, introduces a new problem: if P_2 aborts prior to iteration i^*, 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 $\frac{1}{p}$-secure protocol with polynomially many rounds.

2 Definitions

Preliminaries. A function $\mu(\cdot)$ is *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 $\mathcal{D}_n$ may depend on n.

For a fixed function p, the 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 $a \in \mathcal{D}_n$

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

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.

Functionalities. A *functionality* $\mathcal{F} = \{f_n\}_{n\in\mathbb{N}}$ is a sequence of poly-time computable, randomized mappings $f_n : X_n \times Y_n \to Z_n^1 \times Z_n^2$, where X_n and Z_n^1 (resp., Y_n and Z_n^2) denote the input and output of the first (resp., second) party. 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 $f_n : X_n \times Y_n \to Z_n$. If $\mathcal{F}$ is deterministic, we sometimes call it a *function*. For notational convenience, we sometimes drop the explicit dependence on n.

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 correctness 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 $(f^1(x,y), f^2(x,y))$.

Security of protocols. We consider *active* adversaries, who may deviate from the protocol in an arbitrary manner, and static corruptions. We use the standard real/ideal paradigm [18] (based on [26, 2, 8]). Define $\textsc{ideal}_{\mathcal{F},\mathcal{A}(\mathsf{aux})}(x,y,n)$ as the random variable consisting of the output of the adversary $\mathcal{A}$ and the output of the honest party following a computation of $\mathcal{F}$ in the ideal model (where complete fairness is guaranteed), with security parameter n and parties holding initial inputs x and y, respectively, and auxiliary input aux. We also define $\textsc{real}_{\Pi,\mathcal{A}(\mathsf{aux})}(x,y,n)$ as the analogous random variable for the real-world execution of protocol Π.

Having defined the ideal and real models, we now state our new notion of security. Loosely speaking, our definition asserts that a secure protocol (in the real model) emulates the ideal model (in which a trusted party exists) *to within a difference of* $\frac{1}{p}$. 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*

$$\left\{\textsc{ideal}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x,y,n)\right\} \overset{1/p}{\approx} \left\{\textsc{real}_{\Pi,\mathcal{A}(\mathsf{aux})}(x,y,n)\right\} .$$

Although our definition of $\frac{1}{p}$-security allows privacy to be violated with probability $\frac{1}{p}$, in fact all our protocols are fully private. We remark further that $\frac{1}{p}$-security (even with privacy) and security-with-abort are incomparable.

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

We begin in Section 3.1 by stating a lemma that forms an essential piece of our analysis in the two sections that follow. In Section 3.2 we demonstrate a private and $\frac{1}{p}$-secure protocol for functionalities defined on polynomial-size domains. A slight modification of this protocol is also simultaneously 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 general case). 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 functionalities generating different outputs for each party.

3.1 A Useful Lemma

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 chooses $a_1, \ldots, a_r$ as follows:
   – For $i < i^*$, it chooses $a_i \leftarrow D_1$.
   – For $i \geq i^*$, it chooses $a_i \leftarrow D_2$.
2. The challenger and $\mathcal{A}$ 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 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^*.

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

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

Proof. Fix D_1, D_2 satisfying (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. So 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

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

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

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

This completes the proof.

3.2 $\frac{1}{p}$-Security for Functionalities with Polynomial-Size Domain

In this section, we describe a protocol that works for functionalities where at least one of the domains is polynomial-size. (We stress that the protocol works *directly* for randomized functionalities; the standard reduction from randomized to deterministic functionalities [18] would not apply here since, in general, it makes the domain too large.) Although a small modification of the protocol works even when the parties receive different outputs, for simplicity we assume here that the parties compute a single-output function. We return to the more general setting in the following section.

Theorem 1. *Let $\mathcal{F} = \{f_n : X_n \times Y_n \to Z_n\}$ be a (randomized) functionality where $|Y_n| = \mathsf{poly}(n)$. Assuming the existence of enhanced trapdoor permutations, for any polynomial p there is an $\mathcal{O}(p \cdot |Y_n|)$-round protocol computing $\mathcal{F}$ that is private and $\frac{1}{p}$-secure.*

$\mathsf{ShareGen}_r$

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 $a_i = f_n(x, \hat{y})$.
     - Choose $\hat{x} \leftarrow X_n$ and set $b_i = f_n(\hat{x}, y)$.
   - Compute $z = f_n(x, y)$. For $i = i^*$ to r, set $a_i = b_i = z$.
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. (I.e., $a_i^{(1)}$ 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.

Fig. 1. Functionality $\mathsf{ShareGen}_r$.

Proof. As described in Section 1.2, 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 of Π using a sub-protocol π 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 $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$. In the second stage of Π the parties exchange these shares in a sequence of r iterations as described in Figure 2.

We analyze our protocol in a hybrid model where there is a trusted party computing $\mathsf{ShareGen}_r$ according to the second ideal model where a malicious P_1 can abort the trusted party before it sends output to the honest party. We prove privacy and $\frac{1}{p}$-security of Π in this hybrid model; it follows as in [8] that if we use a sub-protocol for computing $\mathsf{ShareGen}_r$ that is secure-with-abort, then the real-world protocol Π is private and $\frac{1}{p}$-secure.

We first consider the case of a malicious P_1. Intuition for the following claim was given in Section 1.2. The formal statement and proof follow.

Claim 1. *Let Π^{hy} denote an execution of Π in a hybrid model with access to an ideal functionality computing $\mathsf{ShareGen}_r$ (with abort). For every non-uniform, polynomial-time adversary $\mathcal{A}$ corrupting P_1 and running Π^{hy}, there exists a non-uniform, polynomial-time adversary $\mathcal{S}$ corrupting P_1 and running in the ideal*

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 $a_0 = f_n(x, \hat{y})$. Similarly, P_2 chooses $\hat{x} \in X_n$ uniformly at random, and sets $b_0 = f_n(\hat{x}, y)$.
   (b) Parties P_1 and P_2 run a protocol π to 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.

Fig. 2. Generic protocol for computing a functionality f_n.

world with access to an ideal functionality computing $\mathcal{F}$ (with complete fairness), such that $\frac{1}{p}$-security and privacy hold.

Proof. We construct a simulator $\mathcal{S}$ given black-box access to $\mathcal{A}$. *For readability in what follows, we ignore the MAC-tags and keys.* 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 and write X, Y in place of X_n, Y_n.

1. $\mathcal{S}$ invokes $\mathcal{A}$ on the input[2] 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).

[2] We reserve x for the value input by $\mathcal{A}$ to the computation of $\mathsf{ShareGen}_r$.

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 $\mathcal{A}$. (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 $\mathcal{A}$ never aborted (and all r iterations are done), $\mathcal{S}$ outputs what $\mathcal{A}$ outputs and halts.

It is immediate that the view of $\mathcal{A}$ in the simulation above is distributed identically to its view in Π^{hy}; privacy follows. We now prove $\frac{1}{p}$-security.

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). Conditioned on this, 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, although the view of $\mathcal{A}$ is still 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, Lemma 1 implies that this event occurs with probability at most $1/p$. To see this, let D_1 denote the distribution of a_i for $i < i^*$ (i.e., this is the

distribution defined by the output of $f(x, \hat{y})$, for $\hat{y}$ chosen uniformly from Y), and let D_2 denote the distribution of a_{i^*} (i.e., the distribution defined by the output of $f(x, y)$). For any $z \in Z$ we have

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

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.

Next we consider the case of a malicious P_2. A proof of the following is almost identical to that of Claim 1; in fact, the proof is simpler and we can prove a stronger notion of security since P_1 always "gets the output first" in every iteration of Π. For these reasons, a proof is omitted.

Claim 2. *(Informal.) Let Π^{hy} denote an execution of Π in a hybrid model where the parties have access to an ideal functionality computing $\mathsf{ShareGen}_r$ (with abort). Then for any adversary corrupting P_2, protocol Π^{hy} securely computes $\mathcal{F}$ (which in particular implies privacy).*

The results of [8], along with the fact that a secure-with-abort protocol for $\mathsf{ShareGen}_r$ is implied by the existence of enhanced trapdoor permutations, complete the proof of Theorem 1.

Achieving security-with-abort. As written, the protocol is not secure-with-abort. However, the protocol can be modified easily so that it is (without affecting $\frac{1}{p}$-security): simply have $\mathsf{ShareGen}_r$ choose i^* uniformly from $\{2, \ldots, r+1\}$ and 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 since by that time P_1 has already received the correct output.

3.3 $\frac{1}{\mathbf{p}}$-Security for Functionalities with Polynomial-Size Range

The protocol from the previous section does not apply to functions on domains of super-polynomial size, since the round complexity is linear in the size of the smaller domain. Here we show how to extend the protocol to handle arbitrary domains if the range of the function (for at least one of the parties) is polynomial size. We now also explicitly take into account the case when parties obtain different outputs. Intuition for the changes we introduce is given in Section 1.2.

Theorem 2. *Let $\mathcal{F} = \{f_n : X_n \times Y_n \to Z_n^1 \times Z_n^2\}$ be a (randomized) functionality, where $|Z_n^1| = \mathsf{poly}(n)$. Assuming the existence of enhanced trapdoor permutations, for any polynomial p there is an $\mathcal{O}\left(p^2 \cdot |Z_n^1|\right)$-round protocol computing $\mathcal{F}$ that is private and $\frac{1}{p}$-secure.*

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

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 $\hat{x} \leftarrow X_n$ and set $b_i = f_n^2(\hat{x}, y)$.
     - 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 $a_i = f_n^1(x, \hat{y})$.
   - Compute $z_1 = f_n^1(x, y)$ and $z_2 = f_n^2(x, y)$ (if $f_n = (f_n^1, f_n^2)$ is randomized, these values are computed using the same random tape). For $i = i^*$ to r, set $a_i = z_1$ and $b_i = z_2$.
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.

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

Proof. Our protocol Π is, once again, composed of two stages. The second stage is identical to the second stage of the previous protocol (see Figure 2), except that the number of iterations r is now set to $r = p^2 \cdot |Z_n^1|$. The first stage generates shares using a sub-routine π computing a different functionality $\mathsf{ShareGen}'_{p,r}$, parameterized by both p and r and described in Figure 3.

We again analyze our protocol in a hybrid model, where there is now a trusted party computing $\mathsf{ShareGen}'_{p,r}$. (Once again, P_1 can abort the computation of $\mathsf{ShareGen}'_{p,r}$ in the ideal world.) We prove privacy and $\frac{1}{p}$-security of Π in this hybrid model, implying [8] that if the parties use a secure-with-abort protocol for computing $\mathsf{ShareGen}'_{p,r}$, then the real-world protocol Π is private and $\frac{1}{p}$-secure. We first consider the case of a malicious P_1.

Claim 3. *(Informal.) Let Π^{hy} denote an execution of Π in a hybrid model where the parties have access to an ideal functionality computing $\mathsf{ShareGen}'_{p,r}$ (with abort). Then for any adversary corrupting P_1, protocol Π^{hy} privately and $\frac{1}{p}$-securely computes $\mathcal{F}$.*

Proof. The simulator used to prove this claim 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 of the claim.

We next consider the case of a malicious P_2. Note that, in contrast to Claim 2, here we claim only $\frac{1}{p}$-security.

Claim 4. *(Informal.) Let Π^{hy} denote an execution of Π in a hybrid model where the parties have access to an ideal functionality computing $\mathsf{ShareGen}'_{p,r}$ (with abort). Then for any adversary corrupting P_2, protocol Π^{hy} privately and $\frac{1}{p}$-securely computes $\mathcal{F}$.*

Proof. A proof appears in the full version of this work, and is omitted here due to space constraints.

The results of [8], along with the fact that a secure-with-abort protocol for $\mathsf{ShareGen}'_{p,r}$ is implied by the existence of enhanced trapdoor permutations, complete the proof of Theorem 2.

4 Optimality of Our Results

We show that the results of the previous section are optimal as far as generic feasibility is concerned.

4.1 Impossibility of $\frac{1}{p}$-Security and Security-with-Abort Simultaneously

In Section 3.2 (cf. the remark at the end of that section) we showed a protocol achieving $\frac{1}{p}$-security and security-with-abort *simultaneously* for functionalities where at least one of the domains is polynomial-size. We show that if both domains are super-polynomial in size then, in general, it is impossible to achieve both these criteria at once.

Theorem 3. *Let $\mathcal{F} = \left\{\mathsf{EQ}_n : \{0,1\}^{\ell(n)} \times \{0,1\}^{\ell(n)} \to \{0,1\}\right\}$, where EQ_n denotes the equality function on strings and $\ell(n) = \omega(\log n)$. Let Π be any protocol computing $\mathcal{F}$. If Π is secure-with-abort, then Π does not $\frac{1}{p}$-securely compute $\mathcal{F}$ for any $p \geq 4 + \frac{1}{\mathsf{poly}(n)}$.*

Proof. Let Π be a protocol that computes $\mathcal{F}$ and is secure-with-abort. Assume without loss of generality that P_2 sends the first message in Π and that P_1 sends the last message. Say Π has $r = r(n)$ iterations for some polynomial r, where an iteration consists of a message sent by P_2 followed by a message sent by P_1. Let a_0 denote the value that P_1 outputs if P_2 sends nothing, and let a_i, for $1 \leq i \leq r$, 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 $1 \leq i \leq r$, 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, we have $a_i \in \{0,1\}$ 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 experiment 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 $\Pr_{\mathrm{eq}}[\cdot]$.

Lemma 2. $\Pr_{\mathrm{rand}}[a_0 = 1 \vee \cdots \vee a_r = 1]$ *and* $\Pr_{\mathrm{rand}}[b_0 = 1 \vee \cdots \vee b_r = 1]$ *are negligible.*

Proof. This follows from the fact that Π is secure-with-abort. If, say, it were the case that $\Pr_{\mathrm{rand}}[a_0 = 1 \vee \cdots \vee a_r = 1]$ is not negligible, then we could consider an adversarial P_2 that runs the protocol honestly but aborts at a random round. This would cause the honest P_1 to output 1 with non-negligible probability in the real world, whereas P_1 outputs 1 with only negligible probability in the ideal world (since the parties are given independent, random inputs).

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 having to deal with annoying technicalities.) Then

$$\Pr_{\mathrm{eq}}[a_0 = 1 \vee \cdots \vee a_r = 1] = \Pr_{\mathrm{eq}}[b_0 = 1 \vee \cdots \vee b_r = 1] = 1$$

since, in particular, $\Pr_{\mathrm{eq}}[a_r = 1] = \Pr_{\mathrm{eq}}[b_r = 1] = 1$. In a given execution, let i^* denote the lowest index for which $a_{i^*} = 1$, and let j^* denote the lowest index for which $b_{j^*} = 1$. Since

$$\Pr_{\mathrm{eq}}[i^* \leq j^*] + \Pr_{\mathrm{eq}}[i^* > j^*] = 1,$$

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

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* (i.e., in the first ideal model). We denote the probability of events in this experiment 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. It is not hard to show that

$$\Pr_3^{\mathsf{ideal}}[\mathsf{guess} \wedge \mathsf{out}_2 \neq 1] = \frac{1}{2}. \tag{2}$$

(Note that $\mathsf{out}_2 \in \{0, 1\}$ in the first ideal world.) Now take the following real-world adversary $\mathcal{A}$ corrupting P_1: upon receiving input x, adversary $\mathcal{A}$ runs Π honestly but computes a_i after receiving each iteration-i message from P_2. Then:

- If, at some point, $a_i = 1$ then $\mathcal{A}$ aborts the protocol (before sending the iteration-i message on behalf of P_1) and outputs the guess "$b = 1$".
- If $a_i = 0$ for all i, then $\mathcal{A}$ simply runs the protocol to the end (including the final message of the protocol) and outputs the guess "$b = 0$".

We have:

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

using Lemma 2 for the second inequality. Equations (2) and (3) show that Π cannot also be $\frac{1}{p}$-secure for any $p \geq 4 + \frac{1}{\mathsf{poly}(n)}$.

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

Our results show that $\frac{1}{p}$-security is achievable for any functionality $f : X_n \times Y_n \to Z_n^1 \times Z_n^2$ if at least one of X_n, Y_n, Z_n^1, Z_n^2 has polynomial size. Here, we demonstrate that this limitation is inherent.

Define a deterministic, single-output function $\mathcal{F} = \{\mathsf{Swap}_n\}$ with

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

as follows: Fix some $\ell(n) = \omega(\log n)$. Let $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ denote an information-theoretic, one-time MAC for messages of length $2 \cdot \ell(n)$ with key length $O(\ell(n))$ and tag length $\ell(n)$. Then

$$\begin{aligned}&\mathsf{Swap}_n\big((x_1, t_1, k_2), (x_2, t_2, k_1)\big)\\ &\stackrel{\text{def}}{=} \begin{cases} (x_1, x_2) & \text{if } \mathsf{Vrfy}_{k_1}(x_1, t_1) = \mathsf{Vrfy}_{k_2}(x_2, t_2) = 1 \\ \perp & \text{otherwise} \end{cases}.\end{aligned}$$

(Note that both parties receive the same output (x_1, x_2) in the first case.)

Theorem 4. *Function $\mathcal{F}$ cannot be $\frac{1}{p}$-securely computed for any $p \geq 2 + \frac{1}{\mathsf{poly}(n)}$.*

Proof. Consider an ideal-world computation of Swap where:

- x_1, x_2 are chosen uniformly at random from $\{0,1\}^{2\ell(n)}$.
- k_1, k'_1, k_2, k'_2 are output by $\mathsf{Gen}(1^n)$ (i.e., they are random MAC-keys).
- $t_1 = \mathsf{Mac}_{k_1}(x_1)$, $t'_1 = \mathsf{Mac}_{k'_1}(x_1)$, $t_2 = \mathsf{Mac}_{k_2}(x_2)$, and $t'_2 = \mathsf{Mac}_{k'_2}(x_2)$.
- P_1 is given input (x_1, t_1, k_2) and auxiliary information (k'_2, t'_2)
- P_2 is given input (x_2, t_2, k_1) and auxiliary information (k'_1, t'_1).

Define a *win* for P_1 as the event that P_1 outputs x_2 while P_2 fails to output x_1. (A win for P_2 is defined analogously.) It is easy to see that, e.g., a malicious P_1 cannot win in the ideal world, where complete fairness is guaranteed, except with negligible probability. This is because x_2 is a uniform $2\ell(n)$-bit value, while the only information P_1 has about x_2 initially is the $\ell(n)$-bit tag t'_2. Thus, the only way for P_1 to learn x_2 is to submit to the trusted party some input $(\hat{x}_1, \hat{t}_1, \hat{k}_2)$ for which $\mathsf{Vrfy}_{k_1}(\hat{x}_1, \hat{t}_1) = 1$; unless $\hat{x}_1 = x_1$, however, this condition holds with negligible probability.

In any real-world computation of Swap, however, there must be one party who "gets its output first" with probability at least $1/2$, and can identify exactly when this occurs using its auxiliary information. More formally, say we have an r-iteration protocol Π computing Swap where P_2 sends the first message and P_1 sends the last message. Let a_i, for $i = 0, \ldots, r$, denote the second component of the value P_1 would output if P_2 aborts the protocol after sending its iteration-i message, and 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 where $\mathsf{Vrfy}_{k'_2}(a_i, t'_2) = 1$; then output a_i and abort.

An adversary P_2^* can be defined analogously. Note that if, e.g., $\mathsf{Vrfy}_{k'_2}(a_i, t'_2) = 1$ then $a_i = x_2$ except with negligible probability; this follows from the information-theoretic security of the MAC along with the fact that the execution of Π is independent of k'_2, t'_2.

Let i denote the first round in which $\mathsf{Vrfy}_{k'_2}(a_i, t'_2) = 1$, and let j denote the first round in which $\mathsf{Vrfy}_{k'_1}(b_j, t'_1) = 1$. Assuming for simplicity that Π has perfect correctness, we have

$$\Pr[i \leq j] + \Pr[j > i] = 1.$$

Further, since $\left|\Pr[P_1^* \text{ wins}] - \Pr[i \leq j]\right|$ and $\left|\Pr[P_2^* \text{ wins}] - \Pr[i > j]\right|$ are both negligible, we see that either P_1^* or P_2^* wins in the real world with probability at least $1/2 - \mathsf{negl}(n)$. Since an adversary wins in the ideal world with negligible probability, this rules out $\frac{1}{p}$-security for $p > 2$.

Theorem 4 does not contradict the results of [12], or any previous work on fair exchange of signatures. One reason is that prior work on fair exchange typically assumes that each party has no auxiliary information about the other party's secret, whereas our definition (as is standard for definitions of secure computation) accounts for this possibility.[3] Also, in some previous work on fair exchange the running time of the honest party is not bounded by a fixed polynomial, whereas in our setting we require this to be the case.

5 Conclusions and Open Questions

Our work offers a clean definition of partial fairness within the standard real/ideal world paradigm, and settles the question of the general feasibility of achieving this notion in the two-party setting. Several compelling questions remain:

- An easy modification of our second impossibility result (cf. Theorem 4) rules out our definition of partial fairness for the interesting special case of exchanging digital signatures. What is the appropriate (simulation-based?) notion of partial fairness for that setting?
- We can show a function $\mathcal{F} = \{f_n : X_n \times Y_n \to Z_n\}$ for which any protocol computing $\mathcal{F}$ with $\frac{1}{p}$-security requires $\min\{p, |X_n|, |Y_n|\}$ rounds. This leaves a gap as compared to Theorem 1.
- The question of partial fairness in the multi-party setting (with dishonest majority) is wide open. We are not aware of any results in this direction except for the case of coin tossing [10, 27], or functions where complete fairness is possible [21].

References

1. Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *4th Theory of Cryptography Conference — TCC 2007*, volume 4392 of *LNCS*, pages 137–156. Springer, 2007.

[3] Our proof of Theorem 4 exploits this. We can prove an analogue of Theorem 4, based on the assumption that one-way functions exist, that rules out partial fairness even if the adversary has no auxiliary information.

2. D. Beaver. Foundations of secure interactive computing. In *Advances in Cryptology — Crypto '91*, volume 576 of *LNCS*, pages 377–391. Springer, 1992.
3. D. Beaver and S. Goldwasser. Multiparty computation with faulty majority. In *30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 468–473. IEEE, 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 Transactions on Computer Systems*, 1:175–193, 1984.
6. D. Boneh and M. Naor. Timed commitments. In *Advances in Cryptology — Crypto 2000*, volume 1880 of *LNCS*, pages 236–254. Springer, 2000.
7. C. Cachin and J. Camenisch. Optimistic fair secure computation. In *Advances in Cryptology — Crypto 2000*, volume 1880 of *LNCS*, pages 93–111. Springer, 2000.
8. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
9. R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 136–145. IEEE, 2001.
10. R. Cleve. Limits on the security of coin flips when half the processors are faulty. In *18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 364–369. ACM Press, 1986.
11. R. Cleve. Controlled gradual disclosure schemes for random bits and their applications. In *Advances in Cryptology — Crypto '89*, volume 435 of *LNCS*, pages 573–588. Springer, 1990.
12. I. Damgård. Practical and provably secure release of a secret and exchange of signatures. *Journal of Cryptology*, 8(4):201–222, 1995.
13. C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. *Journal of the ACM*, 51(6):851–898, 2004.
14. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Comm. ACM*, 28(6):637–647, 1985.
15. M. Franklin. *Complexity and Security of Distributed Protocols*. PhD thesis, Columbia University, 1993.
16. Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure faut-tolerant protocols and the public-key model. In *Advances in Cryptology — Crypto '87*, volume 293 of *LNCS*, pages 135–155. Springer, 1988.
17. J. A. Garay, P. D. MacKenzie, M. Prabhakaran, and K. Yang. Resource fairness and composability of cryptographic protocols. In *3rd Theory of Cryptography Conference — TCC 2006*, volume 3876 of *LNCS*, pages 404–428. Springer, 2006.
18. O. Goldreich. *Foundations of Cryptography, vol. 2: Basic Applications*. Cambridge University Press, Cambridge, UK, 2004.
19. O. Goldreich and Y. Lindell. Session-key generation using human passwords only. *Journal of Cryptology*, 19(3):241–340, 2006.
20. S. Goldwasser and L. A. Levin. Fair computation of general functions in presence of immoral majority. In *Advances in Cryptology — Crypto '90*, volume 537 of *LNCS*, pages 77–93. Springer, 1991.
21. S. Gordon and J. Katz. Complete fairness in multi-party computation without an honest majority. In *6th Theory of Cryptography Conference — TCC 2009*, volume 5444 of *LNCS*, pages 19–35. Springer, 2009.
22. S. D. Gordon, C. Hazay, J. Katz, and Y. Lindell. Complete fairness in secure two-party computation. In *40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 413–422. ACM Press, 2008.

23. J. Katz. On achieving the "best of both worlds" in secure multiparty computation. In *39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–20. ACM Press, 2007.
24. M. Lepinski, S. Micali, C. Peikert, and A. Shelat. Completely fair SFE and coalition-safe cheap talk. In *23rd ACM Symposium Annual on Principles of Distributed Computing*, pages 1–10. ACM Press, 2004.
25. M. Luby, S. Micali, and C. Rackoff. How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin. In *24th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 23–30. IEEE, 1983.
26. S. Micali and P. Rogaway. Secure computation. In *Advances in Cryptology — Crypto '91*, volume 576 of *LNCS*, pages 392–404. Springer, 1992.
27. T. Moran, M. Naor, and G. Segev. An optimally fair coin toss. In *6th Theory of Cryptography Conference — TCC 2009*, volume 5444 of *LNCS*, pages 1–18. Springer, 2009.
28. B. Pinkas. Fair secure two-party computation. In *Advances in Cryptology — Eurocrypt 2003*, volume 2656 of *LNCS*, pages 87–105. Springer, 2003.
29. A. C.-C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 162–167. IEEE, 1986.