J. Cryptol. (2012) 25: 14–40
DOI: 10.1007/s00145-010-9079-5

Journal of
CRYPTOLOGY

Partial Fairness in Secure Two-Party Computation*

S. Dov Gordon and Jonathan Katz
Dept. of Computer Science, University of Maryland, College Park, USA
gordon@cs.umd.edu; jkatz@cs.umd.edu

Communicated by Oded Goldreich

Received 10 March 2010
Online publication 15 September 2010

Abstract. A protocol for secure computation is *fair* if either both parties learn the output or else neither party does. A seminal result of Cleve (STOC '86) is that, in general, *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. 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 achieve the usual notion of security with abort. We work in the standard communication model (in particular, we do not assume simultaneous channels) and, in contrast to some prior work, rely only on standard cryptographic assumptions (e.g., enhanced trapdoor permutations).

We also show that, as far as general feasibility is concerned, our results are *optimal*. Specifically, there exist functions with super-polynomial domains and ranges for which it is impossible to achieve our definition.

Key words. Secure computation, Fairness.

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 $\mathcal{S}$.

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

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—and 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 [11] shows that complete fairness is impossible to achieve, in general, in the two-party setting. (Specifically, Cleve rules out complete fairness for the coin flipping functionality. We remark that Cleve's result does not rule out complete fairness for *every* functionality; see further discussion at the end of Sect. 1.1.) For this reason, the usual treatment of secure two-party computation (see [19]) *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. (Formal definitions of all these notions are given in Appendix A.1.)

Various methods for achieving *partial* fairness have been suggested; we provide an extensive discussion in Sect. 1.1. With the exception of [18], 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 [19, Sect. 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 discussed, the most desirable (but, in the two-party setting, typically 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 [19] leaves unchanged the requirement of computational indistinguishability, but weakens the ideal world so that fairness is no longer guaranteed at all. Motivated by [25], we suggest 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 [11] and Moran et al. [29] 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. We also demonstrate protocols that achieve this definition for a broad class of functionalities. 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

[1] Katz [25] proposed a related notion of "$\frac{1}{p}$-security-with-abort"; that definition, however, continues to compare the real world with the relaxed ideal world and so again guarantees no fairness at all. A similar relaxation, formalized differently and with different motivation (and again giving no fairness), is also used in [1]. Our definition of $\frac{1}{p}$-security is also similar in spirit to (but weaker than) the notion of ϵ-zero knowledge [14] and is analogous to some definitions of password-based key exchange [20] (although there p is fixed by the size of the password dictionary).

output $z^1 \in Z_n^1$ (resp., $z^2 \in Z_n^2$). (Throughout this paper, n denotes the security parameter. We also impose some technical assumptions on X_n, Y_n, Z_n^1, Z_n^2 that are described in Sect. 2.) 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 in Appendix A.1.) 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 [11,12,29] and contract signing/exchanging secrets [5–7,13,15,27], as well as for the case of general functionalities [3,16–18,21,30,31]. Prior work (with the exception of [18]), however, does not consider a *simulation-based* definition within the standard real/ideal-world paradigm as we do here.[2] Moreover, to the best of our knowledge none of the previous approaches (with the exception of [11,29], 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. (See also [22] for an exhaustive discussion.)

One approach that has been suggested [6,15] 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 computational effort. Simplifying matters for the purposes of this discussion, imagine a protocol that runs for n iterations with the guarantee that after $i \in \{1, \ldots, n\}$ iterations each party can recover its output in time 2^{n-i}. Thus, if a party aborts after some iteration $i > 1$ we know that the other party can recover its output using at most twice the effort as the first party. (If a party aborts in round $i = 1$ then the second party may have no information about its output by the first party can only recover its output using exponential work.) This idea was used for general secure computation in [7,13,17,30], and was formalized by Garay et al. [18] within the framework of universal composability [10]. We note that in this approach there is no *a priori* polynomial bound on the honest party's running time.

An unsatisfying feature of this approach, in all the works cited, is that the decision of whether an honest party should invest the necessary computational effort and recover the output is not specified as part of the protocol, but is somehow decided "externally". Unfortunately, if the adversary knows how this decision is made, then the adversary can abort at "exactly the right time" and violate fairness completely. (Specifically, and

[2] Damgård [13] defines a simulation-based notion of partial fairness, but not within the standard framework.

continuing the example above, if the adversary knows that the honest party is not willing to invest time $T = 2^t$ to recover its output, then the adversary can abort in round $i = n - t - 1$ and recover its own output in time $2T$.) For similar reasons, this approach seems problematic in defending against an adversary who runs in polynomial time, but has more computational power than honest parties are able to invest. (In the example earlier, it could be that $T = \mathsf{poly}(k)$ but still the honest party is unwilling to invest time T to recover its output.) Finally, this technique appears inherently to require strong assumptions regarding the precise time required to solve some specific computational problem.

A second approach, suggested for contract signing/exchanging secrets [5,27] as well as for computation of general functions [3,21] (see also [19]), gradually increases both parties' "confidence" in the output. By way of example (and simplifying slightly for the purposes of this discussion), consider the case where both parties receive the same boolean output $z = f(x, y)$ and imagine a protocol running for $\mathsf{poly}(n)$ iterations where, in each iteration i, the parties learn $b_i = z \oplus r_i$ with the $\{r_i\}$ independent and $\Pr[r_i = 0] = \frac{1}{2} + 1/\mathsf{poly}(n)$. (This is basically the protocol of [3].) When the protocol ends, or when one party aborts, the honest party outputs the majority of the $\{b_i\}$ seen thus far. Protocols of this sort do not appear to apply (at least not without significant changes) to multi-bit outputs or when parties are to receive *different* outputs. More problematic is the fact that an adversary can significantly *bias* the output of the honest party. For example, in the protocol just described, an adversary who always aborts after the first iteration causes the honest party to output an almost-random bit. We may contrast this with the standard simulation-based definition where such bias should, in general, be impossible.

Complete Fairness Gordon et al. [24] 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 [11] 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 using off-line trusted third parties (e.g., [8]) or in non-standard communication models (e.g., [26]). We work in the standard communication model, and without relying on any trusted parties.

1.2. *Overview of Our Approach*

We now give an informal description of our feasibility results (details are in Sect. 3). Let x denote the input of P_1, let y denote the input of P_2, and let $f : X \times Y \to Z$ 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 [24,25,29], 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. (Here, r is a parameter of the protocol that is known in the first stage.) The stages have the following form:

First Stage

1. A value $i^* \in \{1, \ldots, r\}$ is chosen according to some distribution (see below). This represents the iteration in which both parties will learn the "true output" $f(x, y)$.
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); see below for further discussion. 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 (since even a party who aborts after receiving his output in this stage does not learn anything about $f(x, y)$).

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 [25,29], we choose i^* uniformly from $\{1, \ldots, r\}$. (In [24] 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 [24] 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 violated only 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 after iteration i^* 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 2) 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. (A small change to the protocol ensures that it is also secure-with-abort; see Sect. 3.2 for details.) We thus get a protocol with polynomially many rounds as long as Y is polynomial size. The same argument, with the parties' roles interchanged, yields a protocol when X is polynomial size.

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

The above does not work directly when Y has super-polynomial size, because then it may be the case that $\Pr[a_i = f(x, y)] = 1/|Y|$ is negligible (in which case, using the argument in the previous paragraph, we would have to set the number of rounds to be super-polynomial). 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 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 $\mathcal{O}(1/q)$. Taking $q = p$ (along with $r = pq|Z|$) thus gives a $\frac{1}{p}$-secure protocol with polynomially many rounds.

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

1.3. *Overview of Our Impossibility Results*

We also show that the feasibility results described in the previous section are tight, at least with regard to general functionalities. First, we show

Theorem. *There is a deterministic function $\mathcal{F} = \{f_n : \{0,1\}^{\omega(\log n)} \times \{0,1\}^{\omega(\log n)} \to \{0,1\}\}$ for which no protocol computing $\mathcal{F}$ can be simultaneously secure-with-abort and $1/5$-secure.*

This demonstrates that the restriction of the first result from above—namely, that one of the domains of $\mathcal{F}$ must be polynomial size—is inherent as far as general feasibility is concerned.

Our proof of this impossibility result takes $\mathcal{F}$ to be the equality function on $\ell(n)$-bit strings. Roughly, we show that any $1/5$-secure protocol for this function *must* enable one of the parties to (slightly) bias the output of the other party (the definition of $\frac{1}{p}$-security allows bias $\frac{1}{p}$). The presence of such bias means the protocol cannot be secure-with-abort.

We also show

Theorem. *There is a deterministic function $\mathcal{F} = \{f_n : \{0,1\}^{\omega(\log n)} \times \{0,1\}^{\omega(\log n)} \to \{0,1\}^{\omega(\log n)}\}$ for which no protocol computing $\mathcal{F}$ can be $1/3$-secure.*

This demonstrates that our feasibility results for partial fairness from above—namely, that either one of the domains or one of the ranges of $\mathcal{F}$ must be polynomial size—is inherent as far as general feasibility is concerned.

Our proof of this impossibility result takes $\mathcal{F}$ to be an "authenticated exchange" function, where parties P_1 and P_2 want to exchange signed secrets x_1 and x_2 (along with their signatures).[3] It is fairly intuitive that the strategy in which an adversary aborts as soon as he is able to derive *some* secret with a valid signature will violate partial fairness, and our proof formalizes this intuition.

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 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 $a \in \mathcal{D}_n$

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

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.

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. Our results assume that $X_n = \{0,1\}^{\ell(n)}$ for some (known) length parameter $\ell(n)$ (and similarly for Y_n, Z_n^1, and Z_n^2); however, this can be relaxed to require only that (1) membership in X_n, Y_n is efficiently decidable and (2) the uniform distribution over X_n, Y_n, Z_n^1, Z_n^2 is efficiently sampleable.

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

[3] Thus, in this paragraph we are assuming the existence of one-way functions. We actually prove an unconditional result by relying on information-theoretic MACs instead.

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 assume the standard communication model in which the parties alternate sending messages to each other (in particular, we do not assume simultaneous channels). We consider static corruptions by *active* adversaries, who may deviate from the protocol in an arbitrary manner. We define security via the standard real/ideal paradigm [19] (based on [2,9,28]), where 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 world that is secure by definition. For completeness, we include in Appendix A.1 descriptions of the standard ideal- and real-world models. There, we define $\text{IDEAL}_{\mathcal{F},\mathcal{A}(\mathsf{aux})}(x, y, n)$ 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 $\text{REAL}_{\Pi,\mathcal{A}(\mathsf{aux})}(x, y, n)$ 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

$$\left\{\text{IDEAL}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x, y, n)\right\}_{(x,y)\in X\times Y,\, \mathsf{aux}\in\{0,1\}^*,\, n\in\mathbb{N}}$$

$$\overset{1/p}{\approx} \left\{\text{REAL}_{\Pi,\mathcal{A}(\mathsf{aux})}(x, y, n)\right\}_{(x,y)\in X\times Y,\, \mathsf{aux}\in\{0,1\}^*,\, n\in\mathbb{N}}.$$

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 (see Definition 15 in Appendix A.1). Instead, the "defect" in our protocols (that occurs with probability at most $\frac{1}{p}$) always manifests itself with regard to the output of the honest party. For protocols that are $\frac{1}{p}$-secure and also secure-with-abort, this means that the adversary can cause the honest party to output $\perp$ with probability at most $\frac{1}{p}$. For protocols that are $\frac{1}{p}$-secure but not secure with abort, the adversary may be able to cause the honest party to output a value of the adversary's choice with probability at most $\frac{1}{p}$.

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

We begin in Sect. 3.1 by stating a lemma that forms an essential piece of our analysis in the two sections that follow. In Sect. 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 Sect. 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 $r \geq 1$, both known to $\mathcal{A}$. 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 above game.

Lemma 2. *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 (since $\mathsf{Win}(\alpha, r) \leq 1 \leq 1/\alpha$). 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

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

where the first inequality is due to (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

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

This completes the proof. □

3.2. $\frac{1}{p}$-*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 [19] 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 3. *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.*

Proof. As described in Sect. 1.2, our protocol Π consists of two stages. Let p be a polynomial, and set $r = p \cdot |Y_n|$. We implement the first stage of Π using a sub-protocol π for computing a randomized functionality $\mathsf{ShareGen}_r$ (parameterized by a polynomial r) that is defined in Fig. 1. This functionality returns shares to each party, authenticated using an information-theoretically secure r-time MAC $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$. (See Appendix A.2 for a definition.) In the second stage of Π the parties exchange these shares in a sequence of r iterations as described in Fig. 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 [9] that if we use a sub-protocol for comput-

$\mathsf{ShareGen}_r$

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 $1 \le i \le r$, 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 $1 \le i \le r$, 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 $\mathsf{ShareGen}_r$.

ing $\mathsf{ShareGen}_r$ 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 Sect. 1.2. The formal statement and proof follow.

Claim 4. *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 world with access to an ideal functionality computing $\mathcal{F}$ (with complete fairness), such that $\frac{1}{p}$-security holds, i.e.,*

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

and privacy holds, i.e.,

$$\left\{\mathrm{OUT}^{\mathcal{S}}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x, y, n)\right\}_{x \in X_n, y \in Y_n, \mathsf{aux} \in \{0,1\}^*} \equiv \left\{\mathrm{VIEW}^{\mathcal{A}}_{\Pi^{\mathsf{hy}},\mathcal{A}(\mathsf{aux})}(x, y, n)\right\}_{x \in X_n, y \in Y_n, \mathsf{aux} \in \{0,1\}^*}.$$

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.

Proof. We construct a simulator $\mathcal{S}$ that is given black-box access to $\mathcal{A}$. *For readability in what follows, we ignore the MAC-tags and keys.* That is, we do not mention the fact that $\mathcal{S}$ computes MAC-tags for messages it sends 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 and write X, Y in place of X_n, Y_n.

1. $\mathcal{S}$ invokes $\mathcal{A}$ on the input[4] 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).

[4] 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 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 $\mathcal{S}$ 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 2 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

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

Taking $\alpha = 1/|Y|$ and applying Lemma 2, 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 4; in fact, the proof is simpler and we can prove a stronger notion of security (namely, where complete fairness holds) since P_1 always "gets the output first" in every iteration of Π. (This is due to the fact that our protocol specifies that P_2 sends its iteration-i share to P_1 before P_1 sends its iteration-i share to P_2.)

Claim 5 (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 $\mathcal{F}$.*

The results of [9], 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 3. □

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): change $\mathsf{ShareGen}_r$ (cf. Fig. 1) so that i^* is chosen uniformly from $\{2, \ldots, r+1\}$ and so that $b_{i^*-1} = \perp$. This change has no effect in the case of a malicious P_2. For the case of a malicious P_1, Claim 4 (showing $\frac{1}{p}$-security) still holds with an almost identical proof. (The only difference occurs in case 3, when P_1 aborts in round $i = i^*$. Now the honest party in the hybrid world outputs $\perp$ rather than $f(\hat{x}, y)$, but this still happens with probability at most $\frac{1}{p}$.) The protocol is also secure-with-abort, as we now show:

Claim 6. *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 world with access to an ideal functionality computing $\mathcal{F}$ (with abort), such that*

$$\big\{\text{IDEAL}^{\perp}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x, y, n)\big\}_{x \in X_n, y \in Y_n, \mathsf{aux} \in \{0,1\}^*} \equiv \big\{\text{HYBRID}_{\Pi^{\mathsf{hy}},\mathcal{A}(\mathsf{aux})}(x, y, n)\big\}_{x \in X_n, y \in Y_n, \mathsf{aux} \in \{0,1\}^*}.$$

Proof. The proof is largely the same as the proof of Claim 4. The simulator $\mathcal{S}$ is almost identical; the main difference is in step 7 (i.e., the behavior of the simulator when $i = i^*$):

1. $\mathcal{S}$ invokes $\mathcal{A}$ on the input x', the auxiliary input, and the security parameter n. The simulator also chooses $\hat{x} \in X$ uniformly at random.
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 $a_1^{(1)}, \ldots, a_{r+1}^{(1)}$ and $b_1^{(1)}, \ldots, b_{r+1}^{(1)}$. 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 $\hat{x}$ (followed by continue) to the trusted party computing f, outputs whatever $\mathcal{A}$ outputs, and halts. Otherwise (i.e., if $\mathcal{A}$ sends continue), $\mathcal{S}$ proceeds as below.
5. Choose i^* uniformly from $\{2, \ldots, r+1\}$.
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 $\hat{x}$ (followed by continue) to the trusted party, outputs whatever $\mathcal{A}$ outputs, and halts.
7. For $i = i^*$:
   (a) $\mathcal{S}$ 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 $\mathcal{S}$ sends abort to the trusted party computing f, outputs whatever $\mathcal{A}$ outputs, and halts. Otherwise, $\mathcal{S}$ sends continue to the trusted party computing f and proceeds.
8. For $i = i^* + 1$ to r:
   a) $\mathcal{S}$ sets $a_i^{(2)} = a_i^{(1)} \oplus z$ and gives $a_i^{(2)}$ to $\mathcal{A}$.
   b) If $\mathcal{A}$ aborts, then $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs, and halts. If $\mathcal{A}$ does not abort, then $\mathcal{S}$ proceeds.
9. If $\mathcal{A}$ never aborted (and all r iterations are done), $\mathcal{S}$ outputs what $\mathcal{A}$ outputs and halts.

The analysis is also largely the same as in the proof of Claim 4. In fact, the only difference in the proofs occurs when $i = i^*$. Now, the output of the honest party is *identical* in the hybrid and ideal worlds: in each case, the honest party outputs $\perp$. (Recall that the "ideal world" here is one where the adversary is allowed to abort the computation of f, whereas in the proof of Claim 4 the adversary was not given that power in the ideal world.) This completes the proof. □

3.3. $\frac{1}{p}$-*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. The difficulty in the proof—which leads to an actual attack for certain functions $\mathcal{F}$—lies in the application of Lemma 2. To see the problem, let D_1 denote the distribution of the

$\{a_i\}$ values for $i < i^*$, and let D_2 be the point distribution on $a_{i^*} = f(x, y)$ (assuming f is deterministic). The *only* thing we can claim is

$$\Pr\big[a_i = f(x, y)\big] = \Pr_{\hat{y} \in Y}\big[f(x, \hat{y}) = f(x, y)\big] \geq \Pr_{\hat{y} \in Y}[\hat{y} = y]$$
$$= \frac{1}{|Y|} \cdot \Pr_{a_{i^*} \leftarrow D_2}\big[a_{i^*} = f(x, y)\big],$$

and it is easy to come up with specific examples where the inequality is tight. To apply Lemma 2 we would thus need to take $\alpha = 1/|Y|$, and then meaningful security is obtained only if r (the number of rounds) is at least linear in $|Y|$. If $|Y|$ is super-polynomial, this gives an inefficient protocol.

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 Sect. 1.2.

Theorem 7. *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 $\mathcal{O}(p^2 \cdot |Z_n^1|)$-round protocol computing $\mathcal{F}$ that is private and $\frac{1}{p}$-secure.*

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 Fig. 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 subroutine π computing a different functionality $\mathsf{ShareGen}'_{p,r}$, parameterized by both p and r and described in Fig. 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 [9] 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 8. *Let Π^{hy} denote an execution of Π in a hybrid model with access to an ideal functionality computing $\mathsf{ShareGen}'_{p,r}$ (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 world with access to an ideal functionality computing $\mathcal{F}$ (with complete fairness), such that $\frac{1}{p}$-security holds, i.e.,*

$$\big\{\mathrm{IDEAL}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x, y, n)\big\}_{x \in X_n, y \in Y_n, \mathsf{aux} \in \{0,1\}^*}$$
$$\overset{1/p}{\approx} \big\{\mathrm{HYBRID}_{\Pi^{\mathsf{hy}},\mathcal{A}(\mathsf{aux})}(x, y, n)\big\}_{x \in X_n, y \in Y_n, \mathsf{aux} \in \{0,1\}^*},$$

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

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 $1 \le i \le r$, 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 $1 \le i \le r$, 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 $\mathsf{ShareGen}'_{p,r}$.

and privacy holds, i.e.,

$$\left\{\mathrm{OUT}^{\mathcal{S}}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x, y, n)\right\}_{x \in X_n, y \in Y_n, \mathsf{aux} \in \{0,1\}^*} \equiv \left\{\mathrm{VIEW}^{\mathcal{A}}_{\Pi^{\mathsf{hy}},\mathcal{A}(\mathsf{aux})}(x, y, n)\right\}_{x \in X_n, y \in Y_n, \mathsf{aux} \in \{0,1\}^*}.$$

Proof. The simulator used to prove this claim is essentially the same as the simulator used in the proof of Claim 4, 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 2, 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 $\Pr_{a \leftarrow D_1}[a = z] \ge \frac{1}{p} \cdot \frac{1}{|Z_n^1|}$. Regardless of f^1 and y, it therefore holds for all $z \in Z_n^1$ that

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

Setting $\alpha = 1/p \cdot |Z_n^1|$ and applying Lemma 2, 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 5, here we claim only $\frac{1}{p}$-security (and privacy).

Claim 9. *Let Π^{hy} denote an execution of Π in a hybrid model with access to an ideal functionality computing* $\mathsf{ShareGen}'_{p,r}$ *(with abort). For every non-uniform, polynomial-time adversary $\mathcal{A}$ corrupting P_2 and running Π^{hy}, 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 $\frac{1}{p}$-security holds, i.e.,*

$$\left\{\text{IDEAL}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x,y,n)\right\}_{x\in X_n, y\in Y_n, \mathsf{aux}\in\{0,1\}^*} \overset{1/p}{\approx} \left\{\text{HYBRID}_{\Pi^{\mathsf{hy}},\mathcal{A}(\mathsf{aux})}(x,y,n)\right\}_{x\in X_n, y\in Y_n, \mathsf{aux}\in\{0,1\}^*},$$

and privacy holds, i.e.,

$$\left\{\text{OUT}^{\mathcal{S}}_{\mathcal{F},\mathcal{S}(\mathsf{aux})}(x,y,n)\right\}_{x\in X_n, y\in Y_n, \mathsf{aux}\in\{0,1\}^*} \equiv \left\{\text{VIEW}^{\mathcal{A}}_{\Pi^{\mathsf{hy}},\mathcal{A}(\mathsf{aux})}(x,y,n)\right\}_{x\in X_n, y\in Y_n, \mathsf{aux}\in\{0,1\}^*}.$$

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 $\mathcal{S}$ 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.

Privacy is immediate, and so we focus on $\frac{1}{p}$-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 $\mathsf{ShareGen}'_{p,r}$, 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.) 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. □

The results of [9], 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 7. □

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 Sect. 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 10. *Let $\mathcal{F} = \{\mathsf{EQ}_n : \{0,1\}^{\ell(n)} \times \{0,1\}^{\ell(n)} \to \{0,1\}\}$, 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 $\Pr_{\text{rand}}[\cdot]$. 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 $\Pr_{\text{eq}}[\cdot]$.

Claim 11. $\Pr_{\text{rand}}[a_0 = 1 \vee \cdots \vee a_r = 1]$ *and* $\Pr_{\text{rand}}[b_0 = 1 \vee \cdots \vee b_r = 1]$ *are negligible.*

Proof. This follows from the fact that Π is secure-with-abort. If, say, it were the case that $\Pr_{\text{rand}}[a_0 = 1 \vee \cdots \vee a_r = 1]$ 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. If perfect correctness is not assumed then all the calculations below change by at most a negligible additive factor.) Then

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

since, in particular, $\Pr_{\text{eq}}[a_r = 1] = \Pr_{\text{eq}}[b_r = 1] = 1$. 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

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

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

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

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

(Note that $\mathsf{out}_2 \in \{0, 1\}$ in the first ideal world.) Indeed, we can condition on whether P_1 submits x or some value $x' \neq x$ to the trusted party computing EQ; note that P_1's decision on what input to send is independent of b. Then:

- If P_1 submits x then $\Pr[\mathsf{out}_2 = 1] = \frac{1}{2}$, and so (2) holds.
- If P_1 submits $x' \neq x$, its best strategy is to guess '$b = 1$' when the output from the trusted party is 0 since the conditional probability of $b = 1$ in this case is greater than the conditional probability of $b = 0$. (It does not matter what P_1 does when the output from the trusted party is 1, since then the event of interest cannot happen anyway.) Thus P_1 guesses correctly exactly when $b = 1$, and (2) holds.

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

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

using Claim 11 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 $\mathcal{O}(\ell(n))$ and tag length $\ell(n)$. Then

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

(Both parties receive the same output (x_1, x_2) in the first case. Note that the functionality ignores the final two inputs of the parties; those values will be used in the proof, below.)

Theorem 12. *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 $(x_1, t_1, k_2, k_2', t_2')$
- P_2 is given input $(x_2, t_2, k_1, k_1', t_1')$.

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 the final two components of its input. 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 $\mathsf{Vrfy}_{k_1'}(b_j, t_1') = 1$. Assume for simplicity that Π has perfect correctness. (Once

again, this assumption is for simplicity only: if perfect correctness is not assumed then all the calculations below change by at most a negligible additive factor.) We have

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

Further, since $|\Pr[P_1^* \text{ wins}] - \Pr[i \leq j]|$ and $|\Pr[P_2^* \text{ wins}] - \Pr[i > j]|$ 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 $p > 2$. □

We stress that Theorem 12 does not contradict any previous work on fair exchange, since those works consider different models (e.g., exchange of unauthenticated values) and/or definitions of partial fairness than we do here. It is instructive to see why this is the case, e.g., for protocols constructed using the first approach sketched in Sect. 1.1. In such a protocol it might be the case, say, that after $i \in \{1, \ldots, n\}$ rounds each party can recover its output (namely, a_i for P_1 or b_i for P_2) in time 2^{n-i}. But this means that there is no *fixed* polynomial bound in which a_i (or b_i) can be computed for all i. On the other hand, the proof of Theorem 12 relies on the fact that a_i can be computed by an adversary with a fixed (polynomial) upper bound on its running time.

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:

- Our impossibility results rule out our notion of partial fairness, in general, for functions on large domains/ranges. Can we characterize functions on large domains/ranges for which partially fair protocols are possible?
- Can the round complexity of achieving $\frac{1}{p}$-security for general functionalities be improved?
- 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 [4,11,29], or functions where complete fairness is possible [23].

Acknowledgements

The second author thanks Yehuda Lindell for many enlightening discussions on this topic, one of which prompted the inclusion of Sect. 4.1. We thank the authors of [29] for sharing an early copy of their manuscript with us, and the referees for their helpful suggestions.

Appendix A. Additional Definitions

A.1. *Secure Two-Party Computation*

Execution in the Real Model We first 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 some auxiliary input aux and then starts running the protocol, sending all messages on behalf of the corrupted 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 $\text{VIEW}^{\mathcal{A}}_{\Pi,\mathcal{A}(\mathsf{aux})}(x, y, n)$ be the random variable denoting the entire view of the adversary following an execution of Π as described above, and let $\text{OUT}^{\mathsf{hon}}_{\Pi,\mathcal{A}(\mathsf{aux})}(x, y, n)$ be the random variable denoting the output of the honest party. Set

$$\text{REAL}_{\Pi,\mathcal{A}(\mathsf{aux})}(x, y, n) \stackrel{\text{def}}{=} \left(\text{VIEW}^{\mathcal{A}}_{\Pi,\mathcal{A}(\mathsf{aux})}(x, y, n),\ \text{OUT}^{\mathsf{hon}}_{\Pi,\mathcal{A}(\mathsf{aux})}(x, y, n)\right).$$

Execution in the First Ideal Model Here we describe the first ideal model, where complete fairness is guaranteed. 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 $\mathcal{F} = \{(f_n^1, f_n^2)\}$ 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'). (We assume that if $x' \notin X_n$ the trusted party sets x' to some default element in X_n, and likewise if $y' \notin Y_n$.)

Trusted party sends outputs: 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. (Observe that this ensures complete fairness.)

Outputs: The honest party outputs whatever it was sent by the trusted party, and $\mathcal{A}$ outputs an arbitrary (probabilistic polynomial-time computable) function of its view.

We let $\text{OUT}^{\mathcal{A}}_{\mathcal{F},\mathcal{A}(\mathsf{aux})}(x, y, n)$ (resp., $\text{OUT}^{\mathsf{hon}}_{\mathcal{F},\mathcal{A}(\mathsf{aux})}(x, y, n)$) be the random variable denoting the output of $\mathcal{A}$ (resp., the honest party) following an execution in the ideal model as described above. Set

$$\text{IDEAL}_{\mathcal{F},\mathcal{A}(\mathsf{aux})}(x, y, n) \stackrel{\text{def}}{=} \left(\text{OUT}^{\mathcal{A}}_{\mathcal{F},\mathcal{A}(\mathsf{aux})}(x, y, n),\ \text{OUT}^{\mathsf{hon}}_{\mathcal{F},\mathcal{A}(\mathsf{aux})}(x, y, n)\right).$$

We can now define the strongest notion of security:

Definition 13. Let $\mathcal{F}$, Π be as above. Protocol Π is said to 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

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

(Recall that $X \stackrel{\text{c}}{\equiv} Y$ denotes that the two distribution ensembles X, Y are computationally indistinguishable, in the standard sense.)

Execution in the Second Ideal Model In this model, fairness is not guaranteed at all. Nevertheless, this is the usual model in which security of two-party protocols is proved [19, Sect. 7.2.3] since security in the sense of Definition 13 is, in general, unachievable in the two-party setting. The second ideal model is identical to the first ideal model, with the exception of the following step:

Trusted party sends outputs: The trusted party chooses r uniformly at random and computes $z_1 = f_n^1(x', y'; r)$ and $z_2 = f_n^2(x', y'; r)$. If $\mathcal{A}$ controls P_2, the trusted party sends z_1 to P_1 and sends z_2 to P_2.
If $\mathcal{A}$ controls P_1, then the trusted party sends z_1 to P_1, who then sends either continue or abort to the trusted party. In the first case, the trusted party then sends z_2 to P_2. Otherwise, the trusted party sends $\perp$ to P_2.

We stress that only a malicious P_1 is given the opportunity to abort the computation.

We let $\text{IDEAL}^{\perp}_{\mathcal{F},\mathcal{A}(\text{aux})}(x, y, n)$ be the random variable consisting of the output of the adversary and the output of the honest party following an execution in the second ideal model. Security in this weaker model is defined analogously to Definition 13:

Definition 14. Let $\mathcal{F}, \Pi$ be as above. Protocol Π is said to securely compute $\mathcal{F}$ with abort 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

$$\left\{\text{IDEAL}^{\perp}_{\mathcal{F},\mathcal{S}(\text{aux})}(x, y, n)\right\}_{(x,y)\in X\times Y,\text{aux}\in\{0,1\}^*} \stackrel{c}{\equiv} \left\{\text{REAL}_{\Pi,\mathcal{A}(\text{aux})}(x, y, n)\right\}_{(x,y)\in X\times Y,\text{aux}\in\{0,1\}^*}.$$

Finally, we define *privacy* which is weaker than both of the previous definitions. The notion of privacy captures the requirement that the adversary not learn any information about the honest party's input (other than what is implied by the output), but it does not ensure fairness or prevent an adversary from biasing the honest party's output improperly.

Definition 15. Let $\mathcal{F}, \Pi$ be as above. Protocol Π is said to privately 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

$$\left\{\text{OUT}^{\mathcal{S}}_{\mathcal{F},\mathcal{S}(\text{aux})}(x, y, n)\right\}_{(x,y)\in X\times Y,\ \text{aux}\in\{0,1\}^*} \stackrel{c}{\equiv} \left\{\text{VIEW}^{\mathcal{A}}_{\Pi,\mathcal{A}(\text{aux})}(x, y, n)\right\}_{(x,y)\in X\times Y,\ \text{aux}\in\{0,1\}^*}.$$

A.2. *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 (Gen, Mac, 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 16. 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:

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

References

[1] Y. Aumann, Y. Lindell, Security against covert adversaries: efficient protocols for realistic adversaries, in *Theory of Cryptography Conference—TCC 2007*. LNCS, vol. 4392 (Springer, Berlin, 2007), pp. 137–156

[2] D. Beaver, Foundations of secure interactive computing, in *Advances in Cryptology—Crypto '91*. LNCS, vol. 576 (Springer, Berlin, 1992), pp. 377–391

[3] D. Beaver, S. Goldwasser, Multiparty computation with faulty majority, in *30th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE Press, New York, 1989), pp. 468–473

[4] A. Beimel, E. Omri, I. Orlov, Protocols for multiparty coin toss with dishonest majority, in *Advances in Cryptology—Crypto 2010*. LNCS, vol. 6223 (Springer, Berlin, 2010), pp. 538–557

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

[6] M. Blum, How to exchange (secret) keys. *ACM Trans. Comput. Syst.* **1**, 175–193 (1984)

[7] D. Boneh, M. Naor, Timed commitments, in *Advances in Cryptology—Crypto 2000*. LNCS, vol. 1880 (Springer, Berlin, 2000), pp. 236–254

[8] C. Cachin, J. Camenisch, Optimistic fair secure computation, in *Advances in Cryptology —Crypto 2000*. LNCS, vol. 1880 (Springer, Berlin, 2000), pp. 93–111

[9] R. Canetti, Security and composition of multiparty cryptographic protocols. *J. Cryptol.* **13**(1), 143–202 (2000)

[10] R. Canetti, Universally composable security: a new paradigm for cryptographic protocols, in *42nd Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE Press, New York, 2001), pp. 136–145

[11] 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)* (Assoc. Comput. Mach., New York, 1986), pp. 364–369

[12] R. Cleve, Controlled gradual disclosure schemes for random bits and their applications, in *Advances in Cryptology—Crypto '89*. LNCS, vol. 435 (Springer, Berlin, 1990), pp. 573–588

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

[14] C. Dwork, M. Naor, A. Sahai, Concurrent zero-knowledge. *J. ACM* **51**(6), 851–898 (2004)
[15] S. Even, O. Goldreich, A. Lempel, A randomized protocol for signing contracts. *Commun. ACM* **28**(6), 637–647 (1985)
[16] M. Franklin, Complexity and security of distributed protocols. PhD thesis, Columbia University (1993)
[17] Z. Galil, S. Haber, M. Yung, Cryptographic computation: Secure faut-tolerant protocols and the public-key model, in *Advances in Cryptology—Crypto '87*. LNCS, vol. 293 (Springer, Berlin, 1988), pp. 135–155
[18] J.A. Garay, P.D. MacKenzie, M. Prabhakaran, K. Yang, Resource fairness and composability of cryptographic protocols, in *3rd Theory of Cryptography Conference—TCC 2006*. LNCS, vol. 3876 (Springer, Berlin, 2006), pp. 404–428
[19] O. Goldreich, *Basic Applications*. Foundations of Cryptography, vol. 2 (Cambridge University Press, Cambridge, 2004)
[20] O. Goldreich, Y. Lindell, Session-key generation using human passwords only. *J. Cryptol.* **19**(3), 241–340 (2006)
[21] S. Goldwasser, L.A. Levin, Fair computation of general functions in presence of immoral majority, in *Advances in Cryptology—Crypto '90*. LNCS, vol. 537 (Springer, Berlin, 1991), pp. 77–93
[22] S.D. Gordon, Fairness in secure computation. PhD thesis, University of Maryland (2010)
[23] S. Gordon, J. Katz, Complete fairness in multi-party computation without an honest majority, in *6th Theory of Cryptography Conference—TCC 2009*. LNCS, vol. 5444 (Springer, Berlin, 2009), pp. 19–35
[24] S.D. Gordon, C. Hazay, J. Katz, Y. Lindell, Complete fairness in secure two-party computation, in *40th Annual ACM Symposium on Theory of Computing (STOC)* (Assoc. Comput. Mach., New York, 2008), pp. 413–422
[25] J. Katz, On achieving the "best of both worlds" in secure multiparty computation, in *39th Annual ACM Symposium on Theory of Computing (STOC)* (Assoc. Comput. Mach., New York, 2007), pp. 11–20
[26] M. Lepinski, S. Micali, C. Peikert, A. Shelat, Completely fair SFE and coalition-safe cheap talk, in *23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)* (Assoc. Comput. Mach., New York, 2004), pp. 1–10
[27] M. Luby, S. Micali, 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)* (IEEE Press, New York, 1983), pp. 23–30
[28] S. Micali, P. Rogaway, Secure computation, in *Advances in Cryptology—Crypto '91*. LNCS, vol. 576 (Springer, Berlin, 1992), pp. 392–404
[29] T. Moran, M. Naor, G. Segev, An optimally fair coin toss, in *6th Theory of Cryptography Conference—TCC 2009*. LNCS, vol. 5444 (Springer, Berlin, 2009), pp. 1–18
[30] B. Pinkas, Fair secure two-party computation, in *Advances in Cryptology—Eurocrypt 2003*. LNCS, vol. 2656 (Springer, Berlin, 2003), pp. 87–105
[31] A.C.-C. Yao, How to generate and exchange secrets, in *27th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE Press, New York, 1986), pp. 162–167