Minimal Complete Primitives for Secure Multi-Party Computation*

Matthias Fitzi[†] Juan A. Garay[‡] Ueli Maurer[†] Rafail Ostrovsky[¶]

Abstract

The study of minimal cryptographic primitives needed to implement secure computation among two or more players is a fundamental question in cryptography. The issue of complete primitives for the case of two players has been thoroughly studied. However, in the multi-party setting, when there are $n > 2$ players and t of them are corrupted, the question of what are the simplest complete primitives remained open for $t \geq n/3$. (A primitive is called *complete* if any computation can be carried out by the players having access only to the primitive and local computation.)

In this paper we consider this question, and introduce complete primitives *of minimal cardinality* for secure multi-party computation. The cardinality issue (number of players accessing the primitive) is essential in settings where primitives are implemented by some other means, and the simpler the primitive the easier it is to realize. We show that our primitives are complete and of minimal cardinality possible for most cases.

1 Introduction

In (general) secure multi-party computation, n parties (players) attempt to compute multi-party functionalities in the presence of an adversary who may corrupt up to t of them. The study of secure multi-party computation is important both from a theoretical and a practical point of view. It not only establishes what is possible to compute and what is not in multi-party environments, but may sometimes yield techniques and components which may be applied to application-specific systems [Gol00a].

Traditionally, the models for secure multi-party computation have assumed that certain communication "resources" (primitives) are available to the players. Well-known instances include pairwise private channels [BGW88, CCD88], a broadcast channel [RB89], and, more recently, a "mini-cast" primitive, which is available to every triple of players [FM00].

In this paper, and with respect to the strongest, active adversary, we initiate the study of minimal complete primitives for secure multi-party computation from the point of view of the **cardinality** of the primitive — i.e., the number of players accessing it. A primitive is called *complete* if any computation can be carried out by the players having access (only) to the primitive and local computation. A primitive is called *minimal* if any primitive involving fewer players is not complete.

*A preliminary version of this paper appeared in *Advances in Cryptology – CRYPTO '01*, LNCS (2139), Springer-Verlag, pp. 80-100, Santa Barbara, CA, August 2001.

[†]Department of Computer Science, Swiss Federal Institute of Technology (ETH), CH-8092 Zurich, Switzerland. E-mail: {fitzi,maurer}@inf.ethz.ch

[‡]Bell Labs – Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974, USA. garay@research.bell-labs.com.

[¶]Telcordia Technologies Inc., 445 South Street, Morristown, New Jersey 07960-6438, USA. rafail@research.telcordia.com.

For n players, t of which might be corrupted, the question is well understood for $t < n/3$. In this paper we consider this question for $t \geq n/3$. We show that in fact there are three interesting "regions" for t: $t < n/3$, $n/3 \leq t < n/2$, and $n/2 \leq t < n$, and present, for each region, minimal complete primitives for t-resilient unconditional multi-party computation.

1.1 Motivation

There are three main motivations for this work. First, determining the minimal necessary primitives for secure multi-party computation, for various adversary models, is of fundamental theoretical interest. Second, minimizing the number of players involved in a primitive is important because the lower the primitive's cardinality the easier it might be to realize it by exploiting some physical phenomenon (e.g., quantum entanglement). For instance, broadcast among three players [FM00] appears to be a more realistic primitive than Internet-scale broadcast. And third, investigating information-theoretic reductions of multi-party computation to a certain primitive is of interest, even if the implementation of this primitive is based on cryptographic assumptions, for the following reasons:

- The (cryptographic) security proof of a protocol for a complex task is typically easier when the analysis can be broken up into a global information-theoretic argument and a "local" cryptographic argument.
- Information-theoretic multi-party protocols are typically more efficient (though less resilient) than their cryptographic counterparts, and hence the described design approach may lead to the most efficient multi-party protocols.

1.2 Prior and related work

Secure multi-party computation. Secure multi-party computation (MPC) has been actively studied since the statement of the problem by Yao in [Yao82]. For the standard model with secure pairwise channels between the players, the first general solution of the problem was given by Goldreich, Micali, and Wigderson [GMW87] with respect to computational security. Ben-Or, Goldwasser, and Wigderson [BGW88] and Chaum, Crépeau, and Damgård [CCD88] constructed the first general protocols with unconditional security. Additionally, it was proven in [BGW88] that unconditionally secure MPC was possible if and only if less than half (one third) of the players are corrupted passively (actively).

For the model where, in addition to the pairwise secure channels, a global broadcast channel is available, Rabin and Ben-Or [RB89] constructed a protocol that tolerates (less than) one half of the players being actively corrupted. Their solution is not perfect, as it carries a small probability of error. However, it was later shown by Dolev, Dwork, Waarts and Yung [DDWY93] that this is unavoidable for the case $t \geq \lceil n/3 \rceil$ (and the assumed communication primitives), as there exist problems with no error-free solutions in this setting. Fitzi and Maurer [FM00] recently proved that, instead of global broadcast, broadcast among three players is sufficient in order to achieve unconditionally secure MPC for $t < n/2$.

Complete primitives. Another line of research deals with the completeness of primitives available to the players. Kilian [Kil88] proved that oblivious transfer (OT) [Rab81] is complete for two-party computation in the presence of an active adversary. A complete characterization of complete functions for two-party computation, for both active and passive adversaries, was given in [Kil00] based on [Kil91] and results by Beimel, Micali, and Malkin [BMM99]. These results are stated with respect to *asymmetric* multi-party computation in the sense that the result of the function is provided to one single (predefined)

Primitive cardinality	Resiliency		Primitive	Number of instances
	Efficient reduction	Impossibility		
$k = 2$	$t < \frac{n}{3}$	$t \geq \frac{n}{3}$	SC_2	$2\binom{n}{2}$
$k = 3$	$t < \frac{n}{2}$	$t \geq \frac{n}{2}$	OC_3/CC_3	$3\binom{n}{3}$
$4 \leq k \leq n-1$	$t < \frac{n}{2}$	$t \geq n-2$	OC_3/CC_3	$3\binom{n}{3}$
$k = n$	$t < n$	—	UBB_n	1

SC_2: Secure Channel, OC_3: Oblivious Cast, CC_3: Converge Cast, UBB_n: Universal Black Box. OC_3, CC_3 and UBB_n are primitives introduced in this paper.

Table 1: *Complete primitives of cardinality k*

player.

A first generalization of completeness results to the more general n-party case was made by Kilian, Kushilevitz, Micali, and Ostrovsky [KMO94, KKMO99], who characterized all complete boolean functions for multi-party computation secure against a passive adversary that corrupts any number of players.

With the noted exception of Goldreich's treatment of reductions in [Gol00b], previous work on complete primitives typically assumes that the cardinality of the primitive is the same as the number of players involved in the computation.[1] In contrast, in this paper we are concerned with the minimal cardinality of complete primitives for multi-party computation.

1.3 Contributions of this paper

In this paper, for any primitive cardinality k, $2 \leq k \leq n$, we give upper and lower bounds on t such that there is a complete primitive g_k for multi-party computation secure against an active adversary that may corrupt up to t of the players. In particular, for each resiliency "region" $t < \frac{n}{3}$, $t < \frac{n}{2}$, and $t < n$, we present minimal complete primitives for t-resilient unconditional multi-party computation. With one exception — $\frac{n}{2} \leq t \leq n-2$, where the upper bound is n and the lower bound is 3 — all these bounds are tight. To our knowledge, this is the first time that the power of the cardinality of cryptographic primitives — and their minimality — is rigorously studied.

When $k = 2$, it is well known that *secure pairwise channels* (or, more generally, OT) are enough (complete) for $t < n/3$, as it follows from [BGW88, CCD88] and [Kil88]. We show that, for $n > 2$, no primitive of cardinality 2 can go above this resiliency bound, including primitives that are complete for 2-party computation. Specifically, we show that there is a problem, namely, broadcast (aka Byzantine agreement) [PSL80], that cannot be solved in a model where players are connected by "g_2 channels" for any two-party primitive g_2.

The case $k = 3$ is of special interest. We introduce two primitives: *oblivious cast*, a natural generalization of oblivious transfer to the three-party case,[2] and *converge cast*, a primitive that is related to the anonymous channel of [Cha88], and show that they are complete for $t < n/2$. In light of the impossibility result for $k = 2$, these primitives are also minimal. We also show that no primitive of cardinality 3 can be complete for $t \geq n/2$.

[1]Implicitly, though, Kilian, Kushilevitz, Micali, and Ostrovsky [KMO94, KKMO99] transform calls to a cardinality n primitive into pairwise OT invocations. Fitzi and Maurer's work on achieving global broadcast (multi-party computation) based on 2-cast and authenticated (resp., secure) channels can also be viewed as a step towards minimizing the primitives' cardinality.

[2]Independently (but previously), a similar primivite called "Oblivious Multicast" was introduced by Blaze in [Bla96].

For the case $k = n$ we introduce a new primitive, which we call the *universal black box* (UBB), and show that it is complete for arbitrary resiliency ($t < n$). This primitive has interesting implications for computations involving a trusted third party (TTP), in that it enables *oblivious* TTPs, i.e., trusted parties which do not require any prior knowledge of the function to be computed by the players — even if a majority of the players is corrupted. The UBB is also minimal, since we also show that no primitive of cardinality $n - 1$ can be complete for $t < n$, by showing that broadcast tolerating that many faults cannot be achieved by any $(n - 1)$-cardinality primitive.

Table 1 summarizes the results mentioned so far.

Multicast and "convergecast," with a single sender and a single recipient, respectively, constitute two natural communication models. We also show that no primitive of this kind (i.e., a primitive with one input or one output) — even of full cardinality — can achieve more than $t < n/2$, and therefore be more powerful than our primitives of cardinality 3. In other words, with respect to these primitive types, Table 1 "collapses" to two equivalence classes: $k = 2$ and $3 \leq k \leq n$. For these impossibility proofs, we use reductions to two-party problems that are known to be unsolvable.

All the primitives we present allow for *efficient* secure multi-party computation.

1.4 Organization of the paper

We start in Section 2 with definitions of and notation for primitives of arbitrary cardinality, the security model we will be using, and the notion of reducibility and completeness for functionalities and primitives of different cardinality. Section 3 contains the impossibility results, i.e., the lower bounds on the required number of correct players for multi-party computation with primitives of cardinality $2 \leq k \leq n - 1$. In Section 4 we revisit the case of cardinality 2 with a tight bound of $t < n/3$. In Section 5 we introduce the two new primitives of cardinality 3, and show that they are complete for $t < n/2$, which is a tight bound as it follows from Section 3. Finally, in Section 6 we study the case of full cardinality. First, we show the limitation to $t < n/2$ of primitives that are single-input or single-output. Then, we introduce a primitive for arbitrary resiliency; this is also tight, as it follows from Section 3 that full cardinality is necessary in order to achieve arbitrary resiliency. We conclude in Section 7 with open problems and directions for further research. In some of our proofs we will make use of Chernoff bounds; we include a succint description in Section A for convenience.

2 Model and Definitions

In this paper we focus on *secure function evaluation* (SFE) [Yao82] by a set P of n players, where each player p_i has an input value x_i and obtains an output value $f_i(x_1, x_2, \cdots x_n)$, for a (probabilistic) function f_i. We are interested in unconditional security against an active adversary who may corrupt up to t of the players; i.e., the adversary may make the corrupted players deviate from the protocol in an arbitrarily malicious way, and no assumptions are made about his computational power.

In contrast to the treatment of two-party computation (e.g. [Kil91, Kil00] and [BMM99]), where only one predefined player receives the final result of the computation, our model allows every player to receive his own (in general, different) result — which corresponds to the general notion of multi-party computation in [Yao82, CCD88, BGW88]. Similarly, our definition of a primitive, as given in the next paragraph, also allows every involved player to provide an input and get an output, as opposed to just one player. Nonetheless, our constructions apply to the former model as well since for each of our complete multiple-output primitives there is also a single-output primitive that is complete with respect to single-output SFE.

2.1 Primitives of arbitrary cardinality

Our communication model is based on ideal *primitives* that can be accessed by k players, $2 \leq k \leq n$, implementing the secure computation of some k-ary, possibly probabilistic function; k is called the *cardinality* of the primitive. Besides this primitive, no other means of communication is assumed among the players.

We view primitives as "black boxes" in the sense that all implementation details are hidden from the players. Depending on the function being implemented, of the k players accessing the primitive one or more may secretly enter an input to it, and one or more may secretly receive the value(s) of the function.

We use $g_k[i,j]$ to denote the primitive implementing k-ary function g, in which $i \leq k$ players provide an input, and where $j \leq k$ players receive the output of the function.[3] We call $[i,j]$ the *type* of the primitive. We will drop the type when clear from the context. We focus on the following types: $[1,1]$, $[1,k]$, $[k,1]$, and $[k,k]$.[4]

Note that a primitive of a given cardinality can always be simulated (when applicable) by the same primitive with a larger cardinality by "cutting" some of the "wires," or connecting them to the same player. More formally, the following domination relation exists: Let $(k',i',j') \supseteq (k,i,j)$ (meaning $k' \geq k$, $i' \geq i$ and $j' \geq j$); then for every primitive $g_k[i,j]$ there exists a primitive $g'_{k'}[i',j']$ that is as powerful as $g_k[i,j]$.

We assume that every subset $S \subset P$ of k players shares $k!$ instances of the primitive — one for each permutation of the players; thus, we assume $\binom{n}{k}k!$ instances of the primitive in total. However, we will show that there is always a (minimal complete) primitive such that, overall, polynomially-many instances (specifically, less than n^3) of the primitive are sufficient.

2.2 Security model

Several formal definitions of secure function evaluation exist (e.g., [Bea92, Can00, Gol00b, GL90, MR92]). The process is assumed to be synchronous, a fact that simplifies the task of reasoning about security. In [Can00] (and in a nutshell), the computation to be performed among the n players is specified with respect to an incorruptible trusted party τ who interacts securely with the players. For the special case of secure function evaluation where a function on the players' inputs is to be computed and revealed, such a process can be defined by the players first secretly handing their inputs to τ, τ computing the output corresponding to the (possibly probabilistic) function, and then handing it back to the players. Such a protocol among $P \cup \{\tau\}$ is called an *ideal process.*

Of course, the goal of multi-party computation is to perform the same task without the need for a trusted party; thus, a multi-party computation protocol for evaluating a function is called *secure* if it emulates the ideal evaluation process of the function, i.e., if for every strategy of the adversary in the real protocol there is a corresponding adversary strategy that, with similar cost, achieves the same effect in the ideal process. In particular, this means that whenever the ideal process satisfies some consistency or privacy property

[3] A complete specification of the primitive should include additional aspects, such as which i (j) out of the k players provide an input (resp., receive an output), etc., but the simpler notation will be expressive enough for the primitives we will consider.

[4] In the case of $[1,1]$, we always ignore the "reflexive" case (same player providing input and receiving the output).

with respect to the players (e.g., privately computes some specific function on the players' inputs), then the secure protocol also satisfies them. This notion of security can then be refined further by distinguishing among the different types of similarities between the global outputs in both the ideal and real life computations. We are interested in unconditional security, which is obtained by requiring that these output distributions be (statistically) indistinguishable, except for a negligible function of the security parameter, independently of the adversary's computational power. See [Can00] for further details.

2.3 Reducibility and completeness

A main theme in this paper is that of reductions "across" cardinalities. The notion of reduction generalizes to the case of computation of an n-ary function (n-player protocol)[5] invoking another k-ary function (primitive of cardinality k, resp.), with $k \leq n$, in a natural way [Gol00b]:

Definition 1 [Reductions] An n-player protocol *unconditionally reduces* f_n to g_k for a given $t < n$, if it computes f_n unconditionally t-securely just by black-box calls to g_k and local computation. In such a case we say that f_n *unconditionally reduces* (for short, *reduces*) to g_k for that t.[6]

The notion of completeness also generalizes to the different cardinality setting in a natural way: if g_k is complete one can use g_k to perform secure n-party computation. More formally:

Definition 2 [Completeness] We say a primitive g_k is *unconditionally complete* (for short, *complete*) for a given $t < n$, if every n-ary function unconditionally reduces to g_k (for the same t).

Typically, the reduction step is applied more than once, by reducing a primitive already known to be complete to another, perhaps simpler primitive. For example, this is the case in the two-party case, where protocols are given that implement oblivious transfer using a different primitive (see, e.g., [Kil00]). This is also the approach we will follow in this paper, by showing how to implement, using our primitives, the "resources" that are known to be required for SFE.

Furthermore, all our reductions will be unconditionally secure in a way that the simulation can fail with some negligible probability, but, in the non-failure case, it *perfectly* provides the desired functionality; i.e., compared to an ideal implementation of the functionality, the reductions leak no additional information and provide perfect correctness. Hence, by estimating the overall error probability of the complete reduction from the given SFE problem to the complete primitive as the probability that at least one single implementation of a reduction step fails, we actually get an upper bound on the probability that the whole protocol does not provide perfect security. Since our reductions keep this probability negligibly small, we achieve unconditional security according to the definition above.

Finally, we note that all our reductions are *efficient*, i.e., polynomial in n and a security parameter σ such that the overall error probability is smaller than $2^{-\sigma}$.

[5](Local) Computation can, of course, be randomized, and the requirement is that the function's output is produced with high probability.

[6]Note that the definition of reduction also admits the opposite direction, i.e., from smaller cardinality to larger cardinality. Occasionally in our constructions we will also use this direction (for example, by implementing secure pairwise channels using a three-player primitive).

3 Impossibility Results

In this section we prove that no primitive of cardinality $k = 2$ can be complete for $t \geq \frac{n}{3}$ (if $n > 2$), no primitive of cardinality $k = 3$ can be complete for $t \geq \frac{n}{2}$ (if $n > 3$), and that, for general n, no primitive of cardinality $k = n-1$ can be complete for $t \geq n-2$. This is done by showing that there is a problem, namely broadcast (aka Byzantine agreement) [PSL80], that cannot be solved in a model where players are connected by "g_k channels," for any k-party primitive g_k. We first recall the definition of broadcast.

Definition 3 *Broadcast* is a primitive among n players, one sender and $n-1$ recipients. The sender sends an input bit $b \in \{0,1\}$ and the recipients get an output (decision) value $v \in \{0,1\}$ such that the following conditions hold:

Agreement: All correct recipients decide on the same value $v \in \{0,1\}$.

Validity: For all $b \in \{0,1\}$, if the sender is correct and has input b, then all correct recipients decide on the sender's input, $v = b$.

We first consider the case $k = n-1$ and $t \geq n-2$, and then generalize the proof to the case $k \geq 2$ and $t \geq \lceil \frac{k-1}{k+1}\, n \rceil$, thus covering all the impossibility results announced at the beginning of the section.[7] Both proofs are based on the impossibility proof in [FLM86], where it is shown that broadcast for $t \geq \lceil n/3 \rceil$ is not achievable in a model with pairwise authenticated channels. In the new model, however, every subset of $k \geq 2$ players can perform secure multi-party computation. The idea in the proof is to assume that there exists an unconditionally secure broadcast protocol involving all players — interconnected by such "g_k channels" — , which can then be used to build a different system with contradictory behavior, hence proving that such a protocol cannot exist.

Lemma 3.1 *For $k < n$, there is no primitive g_k complete for $t < n$.*

Proof First note that for every k-player primitive with $k < n$, there is an $(n-1)$-player primitive that provides exactly the same functionality by simply ignoring some of the players (cf. Section 2.1). Hence it is sufficient to show that there is no complete $(n-1)$-player primitive for $t < n$.

Suppose, for the sake of contradiction, that there is an $(n-1)$-player primitive BB_{n-1} such that broadcast among the n players $p_0, \ldots, p_{n-1}$ is reducible to BB_{n-1} for $t < n$ — and hence also for $t = n-2$.[8] For simplicity, let $\overline{\mathrm{BB}}_{n-1}$ denote the primitive among $n-1$ players that captures (implements) the functionality of all possible instances of BB_{n-1} that may be shared among $n-1$ players — in general up to $(n-1)!$ instances, since we allow one instance for each permutation of the players. We now assume that each set of $n-1$ players shares exactly one instance of $\overline{\mathrm{BB}}_{n-1}$.

Let $\Pi = \{\pi_0, \ldots, \pi_{n-1}\}$ denote the players' processors with their local programs and, for each $i \in \{0, \ldots, n-1\}$, let π_{i+n} be an identical copy of processor π_i. For every processor π_k, $k \in \{0, \ldots, 2n-1\}$, let the number $(k \mod n)$ be called the *type* of p_k.

Instead of connecting the n original processors as required for the broadcast setting, we build a network involving all $2n$ processors (i.e., the original ones together with their copies) by connecting them with instances of $\overline{\mathrm{BB}}_{n-1}$ such that for every pair of adjacent processors π_i and $\pi_{(i+1) \bmod 2n}$ in the new system and without the presence of an adversary, their common view is indistinguishable from their view as two adjacent processors $\pi_{i \bmod n}$

[7] The preliminary version of this paper had three separate impossibility theorems. The "unified" proof in this generality was proposed by one of the anonymous referees.

[8] Since, by definition, broadcast is trivial if strictly less than two players are correct, this is the non-trivial case that involves the least number of correct players.

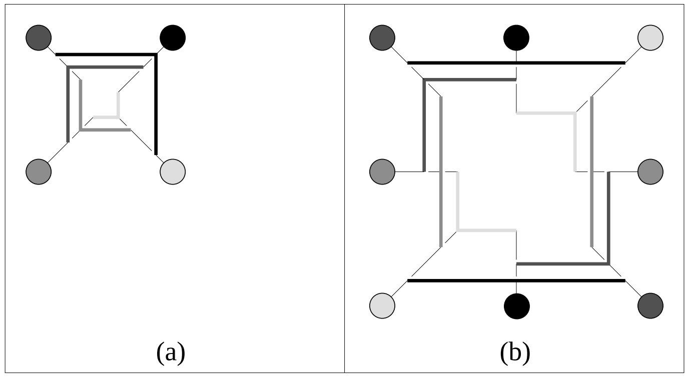

Figure 1: *Reconnection of processors in the proof of Lemma 3.1: special case $n = 4$.*

and $\pi_{(i+1) \bmod n}$ in the original system with an adversary that corrupts the processors in $\Pi \setminus \{\pi_{i \bmod n}, \pi_{(i+1) \bmod n}\}$ in an admissible way.

In order to guarantee that the view of every processor pair π_i and $\pi_{(i+1) \bmod 2n}$ is consistent with their view in the original system, the following two conditions must be satisfied:

1. For every processor π_i, $i \in \{0, \ldots, 2n-1\}$, and for every selection of $n-2$ processors of types different from $(i \bmod n)$, π_i shares exactly one $\overline{\mathrm{BB}}_{n-1}$ with these processors (as it does in the original system).

2. If processor π_i, $i \in \{0, \ldots, 2n-1\}$, shares an instance of $\overline{\mathrm{BB}}_{n-1}$ with a processor of type $\pi_{(i \pm 1) \bmod n}$ (i.e., an adjacent type in the original system), then it shares it with the concrete processor $\pi_{(i \pm 1) \bmod 2n}$ (i.e., its adjacent processor of this type in the new system).

This can be achieved by applying the following rule: For each $\overline{\mathrm{BB}}_{n-1}$ that originally connects all processors except π_i, $i \in \{0, \ldots, n-1\}$, there are now two $\overline{\mathrm{BB}}_{n-1}$'s, one connecting the processors $\{\pi_{i+1}, \ldots, \pi_{i+n-1}\}$ and one connecting the processors $\{\pi_{(i+1+n) \bmod 2n}, \ldots, \pi_{(i+n-1+n) \bmod 2n}\}$. This principle is depicted in Figure 1 for the special case of $n = 4$ and BB_3.

The new system involves two processors of the type corresponding to the sender, namely, π_0 and π_n, and these are the only processors that enter an input. Let now π_0 and π_n be initialized with different inputs, i.e., let's assume that π_0 has input $v_0 \in \{0, 1\}$ and that π_n has input $v_n = 1 - v_0$.

We now show that there are at least two pairs of adjacent processors in the new system (i.e., a fraction $\frac{1}{n}$ of all pairs), for which the broadcast conditions are not satisfied despite being completely consistent with two correct processors in the original system.

First, suppose that Agreement holds for every pair of processors on, without loss of generality, the value v_0. Then the Validity condition is violated in both pairs involving processor π_n, since $v_n \neq v_0$. On the other hand, suppose that the Agreement condition is violated in at least one pair. Then there must exist at least two such pairs because the processors are arranged in a circle. Hence, on inputs $v_0 \in \{0, 1\}$ and $v_n = 1 - v_0$, there must be some pair of adjacent processors $(\alpha, \beta) = (\pi_i, \pi_{(i+1) \bmod 2n})$ that fails to achieve broadcast with probability at least $\frac{1}{n}$. Otherwise, strictly less than two pairs would fail per such invocation of the new system. The view of pair (α, β) is consistent with the view of the pair $(\alpha_0, \beta_0) = (\pi_i, \pi_{(i+1) \bmod n})$ in the original system for one of the cases where the sender inputs either $v_0 = 0$ or $v_0 = 1$. Hence, in the original system, for some input value

$v_0 \in \{0,1\}$, the adversary can force the pair (α_0, β_0) to decide on different values with a probability of at least $\frac{1}{n}$. This contradicts the assumption that broadcast is possible with an arbitrarily small error probability. ■

Theorem 1 *Let $k \geq 2$, $n > k$ and t be integers. If $t \geq \lceil \frac{k-1}{k+1}\, n \rceil$, then there is no primitive g_k complete for t.*

Proof As in the proof of Lemma 3.1, let $\overline{\mathrm{BB}}_k$ capture the whole functionality of all possible instances of BB_k that may be shared by the same k players.

Suppose, for the sake of contradiction, that for some $k \geq 2$ and $n > k$ there is a broadcast protocol $\mathcal{P}$ for $t \geq \lceil \frac{k-1}{k+1} n \rceil$. Let $\Pi = \{\pi_0, \ldots, \pi_{n-1}\}$ be the set of the players' corresponding processors with their local programs and let $\Pi_0 \dot\cup \Pi_1 \dot\cup \ldots \dot\cup \Pi_k = \Pi$ be a partition of Π into $k+1$ sets of cardinalities $|\Pi_i| \in \{\lfloor \frac{n}{k+1} \rfloor, \lceil \frac{n}{k+1} \rceil\}$. In other words, $\ell = n \bmod (k+1)$ sets of cardinality $\lceil \frac{n}{k+1} \rceil$ and $k+1-\ell$ sets of cardinality $\lfloor \frac{n}{k+1} \rfloor$, with the additional property that if $\ell \geq \frac{k+1}{2}$ then, of any two adjacent sets Π_i and $\Pi_{(i+1) \bmod (k+1)}$, at least one is of cardinality $\lceil \frac{n}{k+1} \rceil$. Furthermore, for each $i \in \{0, \ldots, n-1\}$, let π_{i+n} be an identical copy of processor π_i. Finally, for each $i \in \{0, \ldots, k\}$, let $\Pi_{i+n} = \{\pi_{k+n} \,|\, \pi_k \in \Pi_i\}$ form an identical copy of set Π_i.

We now proceed as in the proof of Lemma 3.1. Instead of connecting the original processors as required for the broadcast setting, we build a network involving all $2n$ processors by connecting them with instances of $\overline{\mathrm{BB}}_k$ such that for every set $\Pi_i \cup \Pi_{(i+1) \bmod (2k+2)}$ of processors in the new system, and without the presence of an adversary, their common view is indistinguishable from their view as processors in $\Pi_{i \bmod (k+1)} \cup \Pi_{(i+1) \bmod (k+1)}$ in the original system with an adversary that corrupts the processors in $\Pi \setminus (\Pi_{i \bmod (k+1)} \cup \Pi_{(i+1) \bmod (k+1)})$ in an admissible way.

Note that for every $\overline{\mathrm{BB}}_k$ in the original system there is a set Π_i such that no processor in Π_i is connected to it. This is because there are $k+1$ different sets Π_i. The $\overline{\mathrm{BB}}_k$'s are now reconnected in the following way: For each $\overline{\mathrm{BB}}_k$ that originally connects a set S of k processors in $\Pi \setminus \Pi_i$ ($i \in \{0, \ldots, k\}$), there are now two $\overline{\mathrm{BB}}_k$, one connecting processors $\{\pi_j \in \Pi_{i+1} \cup \cdots \cup \Pi_{i+n-1} \,|\, \pi_{j \bmod (k+1)} \in S\}$ and one connecting processors $\{\pi_j \in \Pi_{i+1+n} \cup \cdots \cup \Pi_{i+n-1+n} \,|\, \pi_{j \bmod (k+1)} \in S\}$.

We now show that every set $\Pi_i \cup \Pi_{(i+1) \bmod (2k+2)}$ of processors contains at least $n-t$ processors, implying that protocol $\mathcal{P}$ satisfies the broadcast conditions with respect to all processors in the union of two such adjacent processor sets. We have two cases:

$$\begin{aligned}
\ell < \tfrac{k+1}{2}: \quad |\Pi_i \cup \Pi_{(i+1) \bmod (2k+2)}| &\geq 2\left\lfloor \tfrac{n}{k+1} \right\rfloor = \left\lfloor \tfrac{2n}{k+1} \right\rfloor = n + \left\lfloor \tfrac{2n-(k+1)n}{k+1} \right\rfloor \\
&= n - \left\lfloor \tfrac{k-1}{k+1} n \right\rfloor \geq n - \left\lceil \tfrac{k-1}{k+1} n \right\rceil \geq n - t \text{ ; and} \\
\ell \geq \tfrac{k+1}{2}: \quad |\Pi_i \cup \Pi_{(i+1) \bmod (2k+2)}| &\geq \left\lfloor \tfrac{n}{k+1} \right\rfloor + \left\lceil \tfrac{n}{k+1} \right\rceil \geq \left\lfloor \tfrac{2n}{k+1} \right\rfloor \geq n - t \,.
\end{aligned}$$

The proof now proceeds in exactly the same way as the proof of Lemma 3.1. By assigning different input values to both senders, π_0 and π_n, we obtain a non-negligible error probability of at least $\frac{1}{k+1}$, which contradicts the assumption of broadcast being possible with an arbitrarily small error probability. ■

The following corollary makes the relevant cases explicit:

Corollary 3.2

- *Let $n \geq 3$. Then there is no primitive g_2 complete for $t \geq \lceil n/3 \rceil$.*
- *Let $n \geq 4$. Then there is no primitive g_3 complete for $t \geq \lceil n/2 \rceil$.*
- *Let $n > k$. Then there is no primitive g_k complete for $t \geq n-2$.*

Moreover, there is strong evidence that no primitive of cardinality $k < n$ can be complete even for $t \geq \lceil n/2 \rceil$, so we put forward the following

Conjecture 3.3 *For $k < n$, there is no primitive g_k complete for $t \geq \lceil n/2 \rceil$.*

4 Primitives of Cardinality 2

It is well known that secure channels (SC_2) are sufficient for unconditional SFE [BGW88, CCD88] with $t < n/3$. That is, in our parlance:

Proposition 4.1 *For any n, there is a primitive of cardinality 2, the secure channel, that is complete for $t < n/3$.*

Since we are assuming that every permutation of the players share a primitive, the type of a secure channel is $[1,1]$; hence, for $t < n/3$, the complete primitive is of the weakest type. For $n > 2$, no primitive of cardinality 2 can be complete for $t \geq \lceil n/3 \rceil$ (Corollary 3.2), therefore this bound is tight.

5 Primitives of Cardinality 3

Evidently, a primitive $g_3[1,1]$ is equivalent to $g_2[1,1]$ since in g_3 one of the players neither provides an input nor receives an output. Hence, in this section we consider primitives (of cardinality 3) of type different from $[1,1]$. In fact, it turns out that either two inputs (and single output) or two outputs (and single input) is sufficient. For each type we introduce a primitive and show it to be complete for $t < n/2$. Moreover, we show that no primitive of cardinality 3 can be complete for $t \geq \lceil n/2 \rceil$.

The approach to show that a primitive is complete is as follows. It follows from [RB89, CDD+99] that pairwise secure channels and a global broadcast channel are sufficient for SFE secure against $t < n/2$ active corruptions. Hence, it is sufficient to show that the primitives introduced in this section imply both, unconditionally secure pairwise channels and global broadcast.

5.1 $\mathbf{g_3[1,*]}$ primitives: Oblivious cast

Definition 4 *Oblivious cast (OC_3)* is a primitive among three players: a sender s who sends a bit $b \in \{0,1\}$ and two recipients r_0 and r_1, such that the following conditions are satisfied:

(1) The bit b is received by exactly one of the recipients, r_0 or r_1, each with probability $\frac{1}{2}$.

(2) While both recipients learn who got the bit, the other recipient gets no information about b. In case there are other players (apart from s, r_0 and r_1), they get no information about b.

Implementing secure channels using oblivious cast. Secure pairwise channels can be achieved by the simulation of authentic channels and the implementation of a pairwise key-agreement protocol between every pair of players p_i and p_j. Players p_i and p_j can then use the key (e.g., a one-time pad) to encrypt the messages to be sent over the authentic channel.

Lemma 5.1 *Let $n \geq 3$. Then authentic channel reduces to oblivious cast for $t < n/2$.*

Proof An authentic channel between players p_i and p_j can be achieved from oblivious cast among p_i, p_j, and some arbitrary third player $p_k \in P \setminus \{p_i, p_j\}$, by p_i (or p_j) oblivious-casting his bit (or whole message) σ times. Finally, p_j decides on the first bit he has received in those oblivious casts.

Since it is sufficient to achieve authentic channels only between pairs of correct players we can assume that the sender is correct. The invocation of this channel fails if p_j does not receive any of the bits being sent by oblivious cast, and this happens with a probability of at most $\Pr_{err}^{auth} = 2^{-\sigma}$. ■

In order to generate a one-time pad (OTP) s_{ij} of one bit between two players p_i and p_j, we can let p_i generate some m random bits $b_1, \ldots, b_m$ and oblivious-cast them to p_j and some arbitrary third player p_k, where m is chosen such that, with overwhelming probability, p_j receives at least one of those random bits (every bit b_x is received by p_j with probability $\frac{1}{2}$). Finally, p_j uses his authentic channel to p_i (Lemma 5.1) to send to p_i the index $x \in \{1, \ldots, m\}$ of the first bit b_x that p_j received. Since p_k gets no information about the bit, bit b_x can be used as an OTP-bit between p_i and p_j. In order to get an OTP of length $\ell > 1$ this process can be repeated ℓ times.[9]

In order to guarantee that the transmission of a bit through the secure channel thus obtained fails with an error probability of at most $\Pr_{err} = 2^{-\sigma}$, we can parameterize m and the security parameter for the invocations of the authentic channel, σ_{auth}, as follows:

- $\Pr_{err}^{oc} \leq 2^{-\sigma-1}$ — the probability that none of the m bits transmitted by oblivious cast is received by player p_j.
- $\Pr_{err}^{auth} \leq 2^{-\sigma-1}$ — the probability that at least one of the invocations of the authentic channel fails.

So we can choose $m = \sigma + 1$. The number of invocations of the authentic channel is $\ell = \lceil \log m \rceil + 1$ ($\lceil \log m \rceil$ for the transmission of index x plus one for the final transmission of the encrypted bit). Hence, σ_{auth} can be chosen as $\sigma_{auth} = \sigma + \lceil \log \ell \rceil + 1$.

Lemma 5.2 *Let $n \geq 3$. Then secure channel reduces to oblivious cast for $t < n/2$.*

Proof From Lemma 5.1 and the discussion above it follows that the secure channel construction has an error probability of $\Pr_{err} \leq \Pr_{err}^{auth} + \Pr_{err}^{oc} \leq 2^{-\sigma}$. ■

Implementing broadcast using oblivious cast. It is shown in [FM00] that a three-party primitive called *weak 2-cast*, defined below, yields global broadcast secure against $t < n/2$ active corruptions. Thus, it is sufficient to show that, using oblivious cast, an implementation of weak 2-cast in any set $S \subset P$, $|S| = 3$, and for any selection of a sender among those players, is possible. We first recall the definition of weak 2-cast from [FM00].

[9] A more efficient way to generate an OTP of length ℓ is to choose a larger m and have p_j send to p_i the indices of the first ℓ bits he received. For simplicity we restrict ourselves to the less efficient but simpler method.

Definition 5 *Weak* 2*-cast* is a primitive among three players: one sender and two recipients. The sender sends an input bit $b \in \{0,1\}$ and both recipients get an output (decision) value $v \in \{0,1,\perp\}$ such that the following conditions hold:

(1) If both recipients are correct and decide on different values, then one of them decides on $\perp$.

(2) If the sender is correct, then all correct recipients decide on his input bit.

The idea behind the implementation of weak 2-cast using oblivious cast is to have the sender repeatedly oblivious-cast his bit a given number of times. Hence, a recipient who receives two different bits reliably detects that the sender is faulty and may safely decide on $\perp$. On the other hand, in order to make the two recipients decide on different bits, a corrupted sender must oblivious-cast 0's and 1's in such a way that each recipient gets one value, but not the other one. However, since the sender cannot influence which of the recipients gets a bit, he can enforce this situation only with exponentially small probability. We now describe the implementation in more detail.

Protocol `Weak-2-Cast-Impl-1`$(s, \{r_0, r_1\}, \sigma)$**:**

1. Sender s oblivious-casts his bit $(\sigma+1)$ times to the recipients.
2. Recipients r_i $(i \in \{0,1\})$ decide $v_i = \begin{cases} 0 & \text{if 0 received at least once, and no 1's;} \\ 1 & \text{if 1 received at least once, and no 0's;} \\ \perp & \text{otherwise.} \end{cases}$

Lemma 5.3 *Protocol* `Weak-2-Cast-Impl-1` *achieves weak* 2*-cast with an error probability of at most* $2^{-\sigma}$*, by only using oblivious cast and local computation.*

Proof If the sender is correct, the protocol can only fail if one of the recipients does not receive any bit from the sender, because the sender always transmits the same bit. This happens with probability $\Pr_{err_1} = 2^{-\sigma}$.

If the sender is incorrect, the protocol may fail only if he manages to make one of the recipients receive all 0's and make the other one receive all 1's. In order to achieve this, after having transmitted the first bit, the sender must correctly guess in advance the recipient of every subsequent bit. This happens with probability $\Pr_{err_2} = 2^{-\sigma}$.

Hence, the error probability is $\Pr_{err} \leq \max(\Pr_{err_1}, \Pr_{err_2}) = 2^{-\sigma}$. ■

Lemmas 5.1 and 5.3 together with the reduction of broadcast to weak 2-cast and authentic channels in [FM00] immediately yield

Lemma 5.4 *Broadcast among* $n \geq 3$ *players reduces to oblivious cast for* $t < n/2$*.*

Lemmas 5.2 and 5.4 and the constructions of [RB89, CDD+99] yield

Theorem 2 *Let* $n \geq 3$*. Then there is a single-input two-output primitive of cardinality* 3*, oblivious cast, that is complete for* $t < n/2$*.*

5.2 $\mathbf{g_3}[*, 1]$ primitives: Converge cast

We now show that a cardinality-3 primitive with two inputs and a single output — i.e., the converse of oblivious cast — is also complete for $t < n/2$. Specifically, we introduce *converge cast*, a primitive related to the "anonymous channel" of [Cha88], defined as follows:

Definition 6 *Converge cast (CC_3)* is a primitive among three players: two senders s_0 and s_1 and one recipient r. The senders send a value x_i, $i \in \{0,1\}$, from a finite domain $\mathcal{D}$, $|\mathcal{D}| \geq 3$, such that the following conditions hold:

(1) The recipient r receives either x_0 or x_1, each with probability $\frac{1}{2}$.

(2) Neither sender learns the other sender's input value, and none of the players learns which of the senders was successful. In case there are other players (apart from s_0, s_1 and r), they get no information about the input values or the successful sender's identity.

As in the previous section, we show how to implement secure channels and broadcast (weak 2-cast). We use "$p_i, p_j \xrightarrow{?} p_k : (x_i, x_j)$" to denote an invocation of converge cast with senders p_i and p_j sending values x_i and x_j, respectively, and recipient p_k. Furthermore, for two sequences s_a and s_b of elements in $\{0, 1, 2\}$ of the same length, we use $\mathcal{H}(s_a, s_b)$ to denote the Hamming distance (difference) between the sequences.

Implementing secure channels using converge cast. We now present a protocol to implement a secure channel from p_0 to p_1 for the transmission of one bit x_0. The idea is as follows: first, p_1 and some other player, say, p_2, each choose two random keys of an adequate length, one for 0 and for 1, and converge-cast them to p_0. p_0 stores the two received keys (note that each received key may contain elements from both senders), and uses the corresponding key as input to a converge cast with p_1 as the recipient to communicate the desired bit. We now describe the protocol in more detail. In what follows, we assume that ℓ is a large-enough positive integer.

Protocol `Secure-Channel-Impl-2`(p_0, p_1, ℓ)**:**

1. Player p_i, $i = 1, 2$, computes random keys $s_i^{(0)}$ and $s_i^{(1)}$ of length ℓ over $\{0, 1, 2\}$
2. $p_1, p_2 \xrightarrow{?} p_0$: $(s_1^{(0)}, s_2^{(0)})$; $(s_1^{(1)}, s_2^{(1)})$ (element-wise) (p_0 receives $s_0^{(0)}; s_0^{(1)}$)
3. $p_0, p_2 \xrightarrow{?} p_1$: $(\sigma_0 = s_0^{(x_0)}, \sigma_2 = 0^\ell)$ (element-wise; p_2 always sends 0) (p_1 receives s_1')
4. p_1: `if` $\mathcal{H}(s_1', s_1^{(0)}) < \frac{7}{12}\ell$ `then` $y_1 = 0$, `else` $y_1 = 1$ `fi`

Lemma 5.5 *Consider protocol* `Secure-Channel-Impl-2`. *For $s \in \{0, 1, 2\}^\ell$ and $k \in \{1, \ldots, \ell\}$ let $s[k]$ denote the projection of s to its k-th dimension, i.e., $s = (s[1], \ldots, s[\ell])$. If p_0 and p_1 are correct, then for every $k \in \{1, \ldots, \ell\}$,*

(1) $s_1'[k] = s_1^{(x_0)}[k]$ *with probability $\frac{1}{2}$, and*

(2) $s_1'[k] = s_1^{(1-x_0)}[k]$ *with probability $\frac{1}{3}$.*

Proof We have to show that this holds independently of p_2's strategy.

(1) We distinguish two cases:

(a) p_2 enters the same input during steps 2 and 3, $x_2[k] := s_2^{(x_0)}[k] = \sigma_2[k]$. Since $s_1^{(x_0)}$ is random (and secret), we have $x_2[k] = s_1^{(x_0)}[k]$ with probability $P_1 = \frac{1}{3}$ and $x_2[k] \neq s_1^{(x_0)}[k]$ with probability $P_2 = \frac{2}{3}$. Hence the overall probability that $s_1'[k] = s_1^{(x_0)}[k]$ is $P = P_1 \cdot 1 + P_2 \cdot \frac{1}{4} = \frac{1}{2}$.

(b) p_2 enters two different inputs during steps 2 and 3, i.e., $s_2^{(x_0)}[k] \neq \sigma_2[k]$. Since $s_1^{(x_0)}$ is random (and secret), we have $s_1^{(x_0)}[k] \neq s_2^{(x_0)}[k] \wedge s_1^{(x_0)}[k] \neq \sigma_2[k]$ with probability $P_1 = \frac{1}{3}$ (in which case $s_1'[k] = s_1^{(x_0)}[k]$ holds with probability $\frac{1}{4}$); $s_1^{(x_0)}[k] = s_2^{(x_0)}[k] \wedge s_1^{(x_0)}[k] \neq \sigma_2[k]$ with probability $P_2 = \frac{1}{3}$ (in which case $s_1'[k] = s_1^{(x_0)}[k]$ holds with probability $\frac{1}{2}$); and $s_1^{(x_0)}[k] \neq s_2^{(x_0)}[k] \wedge s_1^{(x_0)}[k] = \sigma_2[k]$ with probability $P_3 = \frac{1}{3}$ (in which case $s_1'[k] = s_1^{(x_0)}[k]$ holds with probability $\frac{3}{4}$). Hence the overall probability that $s_1'[k] = s_1^{(x_0)}[k]$ is $P = P_1 \cdot \frac{1}{4} + P_2 \cdot \frac{1}{2} + P_3 \cdot \frac{3}{4} = \frac{1}{2}$.

(2) Since $s_1^{(1-x_0)}$ is chosen randomly and p_2 does not get any information about it, the probability that $s_1'[k] = s_1^{(1-x_0)}$ is $P = \frac{1}{3}$.

■

Lemma 5.6 *Let $n \geq 3$. Then secure channel reduces to converge cast for $t < n/2$.*

Proof Consider protocol `Secure-Channels-Impl-2`. First, it is easy to see that p_2 gets no information about bit x_0. We now show that the channel also provides authenticity. The only ways in which the protocol can fail are that either $x_0 = 0$ and $\mathcal{H}(s_1', s_1^{(0)}) \geq \frac{7}{12}\ell$ (call the probability of this event Pr_0), or that $x_0 = 1$ and $\mathcal{H}(s_1', s_1^{(0)}) < \frac{7}{12}\ell$ (probability Pr_1). These probabilities can be estimated by Chernoff bounds (see Appendix A):

- Pr_0: By Lemma 5.5(1), $s_1'[k] = s_1^{(0)}[k]$ with probability $\frac{1}{2}$. Hence, Pr_0 is the probability that out of $N = \ell$ trials with expected value $\mu = \frac{1}{2}$, at most $\frac{5}{12}\ell = \frac{\ell}{2} - \frac{\ell}{12}$ are successful. Applying the lower tail bound, we get $\text{Pr}_0 = \mathcal{C}_{\downarrow}(\frac{1}{2}, \ell, \frac{\ell}{12} - 1) \leq e^{-\frac{2\varepsilon^2}{\ell}} = e^{-\frac{\ell}{72}+\frac{1}{3}-\frac{2}{\ell}} \leq e^{-\frac{\ell}{72}+1}$.
- Pr_1: By Lemma 5.5(2), $s_1'[k] = s_1^{(0)}[k]$ holds with probability $\frac{1}{3}$. Hence, Pr_1 is the probability that out of $N = \ell$ trials with expected value $\mu = \frac{1}{3}$, more than $(1 - \frac{7}{12})\ell = \frac{5}{12}\ell = \frac{\ell}{3} + \frac{\ell}{12}$ are successful. Applying the upper tail bound we get $\text{Pr}_1 = \mathcal{C}_{\uparrow}(\frac{1}{3}, \ell, \frac{\ell}{12}) \leq e^{-\frac{\ell}{72}}$.

Thus, the overall error probability is $\text{Pr}_{err}^{auth} \leq \max(\text{Pr}_0, \text{Pr}_1) = e^{-\frac{\ell}{72}+1}$. ■

Implementing broadcast using converge cast. We now show how weak 2-cast of a bit x_0 from p_0 to p_1 and p_2 can be simulated using CC_3. The idea is as follows: First, p_1 and p_2 choose two random keys of an adequate length, one for 0 and for 1, and converge-cast them to p_0. p_0 stores the two received (mixed) keys. p_0 then sends his input bit to p_1 and p_2 using secure channels. Additionally, p_0 sends to p_1 the (received) key corresponding to his input bit. This key can then be used by p_1 to "prove" to p_2 which value he received from p_0. If things "look" consistent to p_2 (see protocol below), he adopts this value; otherwise, he outputs the value received directly from p_0. Let "$p_i \xrightarrow{!} p_j$" denote the secure channel from p_i to p_j (by means of protocol `Secure-Channels-Impl-2`). The protocol in more detail is as follows:

Protocol `Weak-2-Cast-Impl-2`(p_0, $\{p_1, p_2\}$, ℓ)**:**

1. Player p_i, $i = 1, 2$, computes random keys $s_i^{(0)}$ and $s_i^{(1)}$ of length 2ℓ over $\{0, 1, 2\}$
2. $p_1, p_2 \xrightarrow{?} p_0$: $(s_1^{(0)}, s_2^{(0)})$; $(s_1^{(1)}, s_2^{(1)})$ (element-wise) (p_0 receives $s_0^{(0)}; s_0^{(1)}$)
3. $p_0 \xrightarrow{!} p_i$ $(i = 1, 2)$: $x_0 \in \{0, 1\}$ (p_i receives $x_i \in \{0, 1\}$)
4. $p_0 \xrightarrow{!} p_1$: $s_0^{(x_0)}$ (p_1 receives s_1')
5. p_1: `if` $\mathcal{H}(s_1', s_1^{(x_1)}) \leq \frac{5}{6}\ell$ `then` $y_1 = x_1$, `else` $y_1 = \perp$ `fi`
6. $p_1 \xrightarrow{!} p_2$: y_1; s_1' (p_2 receives y_2; s_2')
7. p_2: `if` $(y_2 = \perp) \lor \left(\mathcal{H}(s_2', s_2^{(y_2)}) \geq \frac{7}{6}\ell\right)$ `then` $y_2 = x_2$ `fi`

Lemma 5.7 *Given a pair* $(x, y) \in \{0, 1, 2\}^2$ *such that* x *and* y *are the senders' inputs for an invocation of* CC_3*. If* x *is chosen uniformly at random from* $\{0, 1, 2\}$*, then the receiver's output is* x *with probability* $Pr_2 = \frac{2}{3}$*.*

Proof By the definition of converge cast, the probability that x is received is $\Pr_2 = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{3} = \frac{2}{3}$. ∎

Lemma 5.8 *Let* $n \geq 3$*. Then weak* 2*-cast reduces to converge cast for* $t < n/2$*.*

Proof Consider protocol `Weak-2-Cast-Impl-2`. Let us neglect the error probabilities of the secure channel invocations (protocol `Secure-Channels-Impl-2`) until the end of the proof. Since the conditions for weak 2-cast are trivially satisfied if more than one player is corrupted, we can distinguish three cases.

All players correct or at most p_2 **corrupted.** The only way the protocol can fail is if p_1 decides on $\perp$ ($\mathcal{H}(s'_1, s_1^{(x_1)}) > \frac{5}{6}\ell$); i.e., that s'_1 and p_1's key differ in more than $\frac{5}{6}\ell$ elements. By Lemma 5.7, since p_1 is correct, for each invocation of converge cast during step 2 of the protocol it holds that p_0 does not receive the value $s_1^{(*)}$ with probability $1 - \Pr_2 = \frac{1}{3}$. Since all invocations are independent, the failure probability $\Pr_3$ can be estimated by a Chernoff bound (cf. Appendix A) with parameters $\mu = \frac{1}{3}$, $N = 2\ell$, and $\varepsilon = \frac{5}{6}\ell - \mu N = \frac{5}{6}\ell - \frac{2}{3}\ell = \frac{\ell}{6}$:

$$\Pr_3 \leq \mathcal{C}_{\uparrow}(\frac{1}{3}, 2\ell, \frac{\ell}{6}) \leq e^{-\frac{2\varepsilon^2}{2\ell}} = e^{-\frac{\ell}{36}} .$$

p_1 **corrupted.** In order to achieve that p_2 decides on a wrong output it must hold that $y_2 = 1 - x_0$ and $\mathcal{H}(s'_2, s_2^{(y_2)}) < \frac{7}{6}\ell$ before step 7 of the protocol; i.e., that less than $\frac{7}{6}\ell$ elements of s'_2 do not match p_2's key. Since p_1 does not learn anything about the elements in $s_2^{(y_2)}$, he must guess more than $\frac{5}{6}\ell$ out of the 2ℓ elements correctly, as otherwise p_2 would decide on x_0. For each single element the probability of guessing correctly is $\frac{1}{3}$. Hence the probability of this event can again be estimated by a Chernoff bound with parameters $\mu = \frac{1}{3}$, $N = 2\ell$, and $\varepsilon = \frac{5}{6}\ell - \mu N = \frac{5}{6}\ell - \frac{2}{3}\ell = \frac{\ell}{6}$:

$$\Pr_4 \leq \mathcal{C}_{\uparrow}(\frac{1}{3}, 2\ell, \frac{\ell}{6}) \leq e^{-\frac{2\varepsilon^2}{2\ell}} = e^{-\frac{\ell}{36}} .$$

p_0 **corrupted.** In order to make the protocol fail, p_0 must select $x_1 \in \{0, 1\}$ and $x_2 = 1 - x_1$, and achieve that p_1 decides on $y_1 = x_1$. In this case, since p_1 and p_2 are correct, we get $s'_1 = s'_2 \stackrel{\text{def}}{=} s'$ and $x_1 = y_1 = y_2 \stackrel{\text{def}}{=} y$ after step 6. In particular, p_0 must achieve that $\mathcal{H}(s', s_1^{(y)}) \leq \frac{5}{6}\ell$ and $\mathcal{H}(s', s_2^{(y)}) \geq \frac{7}{6}\ell$, which implies that s' must be formed such that it matches $s_1^{(y)}$ in at least $\frac{2}{6}\ell = \frac{1}{3}\ell$ more elements than it matches $s_2^{(y)}$. For each single element $s'[k]$, no matter how p_0 chooses it, the probability that $s'[k] = s_1^{(y)}[k] \neq s_2^{(y)}[k]$ is the same as the probability that $s'[k] = s_2^{(y)}[k] \neq s_1^{(y)}[k]$, since converge cast is symmetric with respect to p_1 and p_2, who both choose their keys uniformly at random.

Let X_k be a random variable that takes on values in $\{0, \frac{1}{2}, 1\}$, with

$$X_k = \begin{cases} 0 & \text{if } s'[k] = s_1^{(y)}[k] \neq s_2^{(y)}[k]; \\ \frac{1}{2} & \text{if } s_1^{(y)}[k] = s_2^{(y)}[k]; \text{ and} \\ 1 & \text{if } s'[k] = s_2^{(y)}[k] \neq s_1^{(y)}[k]. \end{cases}$$

Then, independently of p_0's strategy, a failure of the protocol corresponds to the probability that for $N = 2\ell$ independent trials with respect to random variable X_k (with expected value $\frac{1}{2}$), it holds that

$$\sum_{k=1}^{N=2\ell} X_k \geq \mu N + \frac{1}{2}(\frac{7}{6}\ell - \frac{5}{6}\ell) = \mu N + \frac{1}{6}\ell \ .$$

This probability can be estimated by a Chernoff bound with parameters $\mu = \frac{1}{2}$, $N = 2\ell$, and $\varepsilon = \frac{\ell}{6} - 1$:

$$\Pr_5 \leq \mathcal{C}_{\uparrow}(\frac{1}{2}, 2\ell, \frac{\ell}{6} - 1) \leq e^{-\frac{2\varepsilon^2}{2\ell}} = e^{-\frac{\ell}{36}+\frac{1}{3}-\frac{36}{\ell}} \leq e^{-\frac{\ell}{36}+1} \ .$$

Since the error probability of protocol `Secure-Channels-Impl-2` can be made negligibly small, it can be parameterized such that the overall probability that at least one invocation fails satisfies $\Pr_{SC} \leq e^{-\frac{\ell}{36}}$. Thus, the overall error probability of the weak 2-cast construction based on converge cast is at most

$$\Pr_{err} \leq \Pr_{SC} + \max(\Pr_3, \Pr_4, \Pr_5) \leq e^{-\frac{\ell}{36}} + e^{-\frac{\ell}{36}+1} \leq 2e^{-\frac{\ell}{36}+1} < e^{-\frac{\ell}{36}+2} \ .$$

For security parameter σ, we let $\ell \geq 36(\sigma + 2)$, and hence $\Pr_{err} \leq e^{-\sigma}$. ∎

As before, Lemmas 5.6 and 5.8 and the constructions of [RB89, CDD$^+$99, FM00] yield

Theorem 3 *Let $n \geq 3$. Then there is a two-input, single-output primitive of cardinality 3, converge cast, that is complete for $t < n/2$.*

We note that allowing the inputs of converge cast to be from a larger domain (than $\{0, 1, 2\}$) considerably improves the efficiency of our reductions.

6 Primitives of Full Cardinality

In this section, we first show that even cardinality n does not help if the primitive is restricted, in the sense of having either a single input or a single output. Such a primitive is no more powerful than a primitive of cardinality 3 (Section 5). We then introduce a new primitive of type $[n, n]$, the *universal black box* (UBB_n), which allows for arbitrary resiliency ($t < n$); it follows from Section 3 that full cardinality is in fact necessary to achieve this. The UBB_n has an interesting application to computations involving a trusted third party: its functionality enables *oblivious* trusted third parties, that is, trusted parties which do not require any prior knowledge of the function to be computed by the players. We start with the impossibility results for restricted primitives.

6.1 $\mathbf{g_n[1, *]}$ and $\mathbf{g_n[*, 1]}$ primitives

Theorem 4 *There is no $g_n[1, *]$ primitive complete for $t \geq \lceil n/2 \rceil$.*

Proof The proof is by reduction to a two-party problem whose private computation is knwon to be impossible. Assume that there exists a primitive $g_n[1, *]$ that is complete for $t \geq \lceil n/2 \rceil$, and consider two players, p and q, who want to compute the logical OR of their input bits. We can have both players each simulate up to $\lceil n/2 \rceil$ of the players involved in the complete primitive g_n (in such a way that every original player is simulated either by

p or q) which allows them to privately compute the OR function. Since there is only one input to g_n (to be given by, say, player p), there must be a first invocation of the primitive in which p provides information about his input that allows for the computation of the function. This, however, is the scenario of Theorem 2 in [BGW88], where it is shown that this player may not reveal any information about his input to the other player unless he knows that the other player's input is 1. ∎

Theorem 5 *There is no $g_n[*, 1]$ primitive complete for $t \geq \lceil n/2 \rceil$.*

Proof The proof is again by reduction to another two-party problem known to have no solution. Suppose that there is a primitive of type $[*, 1]$ that is complete for $t \geq \lceil n/2 \rceil$. We can have two players p and q each simulate up to $\lceil n/2 \rceil$ of the players involved in this primitive which allows them to securely compute any function on their inputs. Thus, there is a two-player primitive with a single output that is complete with respect to any computation where both players learn the same result. This is a direct contradiction to the "one-sidedness" observation in [BMM99], that a protocol based on an asymmetric two-player primitive cannot guarantee that both players learn the result. ∎

6.2 $\mathbf{g_n[n, n]}$ primitives: The universal black box

We now introduce the *universal black box* (UBB_n), a complete primitive for $t < n$. At first, it might seem trivial to build a complete primitive for arbitrary t by just implementing the functionality of a trusted party. However, computations by trusted parties are generally based on the fact that the trusted party already knows the function to be computed. But since the primitive must be universally applicable, it cannot have any prior information about what is to be computed, i.e., what step of what computation is to be executed. Hence, the specification of the computation step to be performed by the black box must be entered by the players at every invocation of the black box. Although there seems to be no apparent solution to this problem since a dishonest majority might always overrule the honest players' specification, we now describe how the UBB_n effectively overcomes this problem.

For the description, we find it convenient to first assume that the function to be computed has a single output, and present the extension to the general case (multiple outputs, multiple functions) later. I.e., assume that the function to be computed receives inputs from all the players, but exactly one player, say, p_0, is to learn the result of the computation. (Note that this does not contradict Theorem 5.)

The main idea behind a UBB_n for this case is simple: It contains a universal circuit [Val76], and has two inputs per player:

- the *function input*, where the player specifies the function to be computed on all argument inputs, and
- the *argument input*, where the player inputs his argument to the function.

The UBB_n now computes the function specified by player p_0, but for every player that does not input the same function as p_0, it replaces his argument input by some fixed default value. Finally, the function is computed by evaluating the universal circuit on p_0's function and all argument inputs, and its output is sent to player p_0. Note that only one invocation of the UBB_n is required. Intuitively, privacy holds since no player other than p_0 receives any output and therefore learns anything, and p_0 gets no more information from the other players' arguments than he would from an ideal process involving a trusted party. Correctness follows since the function to be computed is selected by p_0 himself.

The single-output primitive above can now be turned into a UBB_n to compute multi-output functions by the following modification. Instead of one function, the function input specifies n functions to be computed on the inputs — one function per player. The function f_i, $1 \leq i \leq n$, to be computed and output to player p_i is determined by player p_i himself, and for the computation of f_i the argument inputs of only those players who agree on the *same* n functions $f_1, \ldots, f_n$ with p_i (i.e., players whose function input matches p_i's) are considered by the UBB_n. The following theorem formalizes and extends the argument above.

Theorem 6 *There is a primitive of cardinality n, the universal black box, that is complete for $t < n$.*

Proof We argue that privacy and correctness hold for arbitrary t.
Privacy: For $1 \leq i \leq n$, a player p_i's output can give information about other player p_j's argument input only if p_j entered the same function input as p_i, which means that p_j had "agreed" on exactly this computation. Hence, p_i gets the same information about p_j's argument as in an ideal process involving a trusted party. If p_i is corrupted and inputs a wrong function input, no argument from a correct player will be used by the UBB_n for this computation.
Correctness: The functions to be computed and output to the correct players are selected by the correct players themselves. Corrupted players that input a different set of functions only achieve that their inputs be replaced by a default value — a strategy that is also (easily) achievable in an ideal process involving a trusted party. ■

The above has the following implication for computations involving a trusted third party:

Corollary 6.1 [Oblivious TTPs] *Computations involving a trusted third party do not require the trusted party to have any prior knowledge of the task to be completed by the players.*

7 Summary and Open Problems

Originally (Section 2.1), we assumed that one primitive instance was available for every permutation of every k-tuple of players, i.e., $\binom{n}{k}k!$ instances. In contrast, it follows from Proposition 4.1 and from the constructions for the proofs of Theorems 2, 3, and 6 that there is always a minimal complete primitive such that at most $3\binom{n}{3}$ instances of the primitive are required for the computation of any function.

Corollary 7.1 *For each cardinality k, $2 \leq k \leq n$, and each primitive type, there is a complete primitive such that at most $3\binom{n}{3}$ instances are sufficient for unconditional SFE.*

In this paper we have put forward the concept of minimal cardinality of primitives that are complete for SFE. Since this is a new line of research, several questions remain open.

We completely characterized the cases of types $[1,1]$, $[1,k]$, and $[k,1]$, for all cardinalities $k \leq n$. In particular, for $t < n/3$ there is a complete primitive, $\text{SC}_2[1,1]$, and no g_2 can do any better; and, for $t < n/2$, there are two complete primitives, $\text{OC}_3[1,2]$ and $\text{CC}_3[2,1]$, and no g_k, $k \leq n$, can do any better. For the case of type $[k,k]$ it remains to prove Conjecture 3.3, that no $g_{n-1}[n-1,n-1]$ is complete for $t \geq \lceil n/2 \rceil$. This would partition the whole hierarchy into three equivalence classes of cardinalities $k = 2$ $(t < n/3)$, $2 < k < n$ $(t < n/2)$, and $k = n$ $(t < n)$.

In this paper we chose to present primitives that are *symmetric*, since they are conceptually simpler and perhaps more "natural." We note that asymmetric, more convoluted primitives are also possible, and they can be "tailored" in such a way that the analysis required by the reductions is less involved. This was indeed the case with a precursor of converge cast where the receiver would also provide an input. For another example of such primitive, consider the following $g_3[1,*]$ (as suggested by one of the referees): the sender p_0 has two input bits, s and b, and the output of p_1 is *always* b. If $s=0$ then the output of p_2 is $\perp$, and if $s=1$ the output of p_2 is b. This primitive simply simulates a secure channel and a weak 2-cast.

It would also be interesting to analyze the completeness of primitives as a function of the size of the input and output domains. For example, if the primitive CC_3 were restricted to one single input bit per player and one single output bit, it would not be complete for $t<n/2$. Also the completeness of the UBB_n for $t<n$ relies on the fact that inputs of large size are allowed.

Acknowledgements

We thank the anonymous referees for *CRYPTO '01* and for the *Journal of Cryptology* for their valuable comments. The work of Matthias Fitzi was partly done while visiting Bell Labs.

References

[Bea92] D. Beaver. Foundations of secure interactive computation. In *Advances in Cryptology — CRYPTO '91*, LNCS, pp. 377–391. Springer-Verlag, 1992.

[BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th ACM Symp. on the Theory of Computing*, pp. 1–10, 1988.

[Bla96] M. Blaze. Oblivious Key Escrow. In R. Anderson, editor, *Proc.First Infohiding*, LNCS, pp. 335–343, Cambridge, U.K., 1996. Springer-Verlag.

[BMM99] A. Beimel, T. Malkin, and S. Micali. The all-or-nothing nature of two-party secure computation. In *Advances in Cryptology — CRYPTO '99*, volume 1666 of *LNCS*, pp. 80–97. Springer-Verlag, 1999.

[Can00] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[CCD88] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proc. 20th ACM Symp. on the Theory of Computing*, pp. 11–19, 1988.

[CDD+99] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Advances in Cryptology — EUROCRYPT '99*, LNCS, 1999.

[Cha88] D. Chaum. The Dining Cryptographers Problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[DDWY93] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *Journal of the ACM*, 40(1):17–47, Jan. 1993.

[FLM86] M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1:26–39, 1986.

[FM00] M. Fitzi and U. Maurer. From partial consistency to global broadcast. In *32nd Annual Symp. on the Theory of Computing*, pp. 494–503, 2000.

[GL90] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *Advances in Cryptology — CRYPTO '90*, volume 537 of *LNCS*. Springer-Verlag, 1990.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. 19th ACM Symp. on the Theory of Computing*, pp. 218–229, 1987.

[Gol00a] O. Goldreich. Preface. *Journal of Cryptology,* Special issue on General Secure Multi-party Computation, 13(1):1–8, 2000.

[Gol00b] O. Goldreich. Secure multi-party computation, working draft, version 1.2, Mar. 2000.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proc.20th Annual ACM Symp. on the Theory of Computing*, pp. 20–31, 2–4 May 1988.

[Kil91] J. Kilian. A general completeness theorem for two-party games. In *Proc.23rd Annual ACM Symposium on the Theory of Computing*, pp. 553–560, New Orleans, Louisiana, 6–8 May 1991.

[Kil00] J. Kilian. More general completeness theorems for secure two-party computation. In *Proc.32nd Annual ACM Symp. on the Theory of Computing*, pp. 316–324, Portland, Oregon, 21–23 May 2000.

[KKMO99] J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in private computations. *SIAM Journal on Computing*, 29, 1999.

[KMO94] E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in multi-party private computations. In *Proc. 35th Annual IEEE Symp. on the Foundations of Computer Science*, pp. 478–491, Nov. 1994.

[MR92] S. Micali and P. Rogaway. Secure computation. In *Advances in Cryptology — CRYPTO '91*, volume 576 of *LNCS*, pp. 392–404. Springer-Verlag, 1992.

[PSL80] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, Apr. 1980.

[Rab81] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.

[RB89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. 21st ACM Symp. on the Theory of Computing*, pp. 73–85, 1989.

[Val76] L. G. Valiant. Universal circuits. In *ACM Symposium on Theory of Computing (STOC '76)*, pp. 196–203, New York, May 1976. ACM Press.

[Yao82] A. C. Yao. Protocols for secure computations. In *Proc. 23rd IEEE Symp. on the Foundations of Computer Science*, pp. 160–164. IEEE, 1982.

A Chernoff Bounds

In the analysis of our protocols we apply Chernoff bounds in order to estimate upper bounds on their error probabilities. We present a succint description of these bounds here for convenience.

Let $X_i \in [0,1]$ $(1 \leq i \leq N)$ be a sequence of independent random variables with expected value μ. The Chernoff bound gives an upper bound on the probability that the sum $\sum_{i=1}^{N} X_i$ deviates from the expected value μN by a given distance $\varepsilon > 0$:

$$\begin{aligned} \mathcal{C}_{\downarrow}(\mu, N, \varepsilon) &= \mathrm{Prob}(\textstyle\sum_{i=1}^{N} X_i < \mu N - \varepsilon) \leq e^{-\frac{2\varepsilon^2}{N}} \\ \mathcal{C}_{\uparrow}(\mu, N, \varepsilon) &= \mathrm{Prob}(\textstyle\sum_{i=1}^{N} X_i > \mu N + \varepsilon) \leq e^{-\frac{2\varepsilon^2}{N}} \ . \end{aligned} \tag{1}$$