Minimal Complete Primitives for Secure Multi-Party Computation

Matthias Fitzi[1], Juan A. Garay[2], Ueli Maurer[1], and Rafail Ostrovsky[3]

[1] Dept. of Computer Science, Swiss Federal Institute of Technology (ETH), CH-8092 Zürich, Switzerland. {fitzi,maurer}@inf.ethz.ch.
[2] Bell Labs – Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974, USA. garay@research.bell-labs.com.
[3] Telcordia Technologies Inc., 445 South Street, Morristown, New Jersey 07960-6438, USA. rafail@research.telcordia.com.

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 $t \geq n/3$. 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 the primitives are implemented by some other means, and the simpler the primitive the easier it is to realize it. We show that our primitives are complete and of minimal cardinality possible.

1 Introduction

In this paper, with respect to the strongest, active adversary, we initiate the study of minimal complete primitives for 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 less players is not complete.
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 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) 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 [Gol00], 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. In contrast, in this paper we are concerned with the minimal cardinality of complete primitives for multi-party computation.

1.2 Our results

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 corrupting that many players. With one exception, all these bounds are tight. 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. To our knowledge, this is the first time that the power of the cardinality of cryptographic primitives — and their minimality — is rigorously studied.

Primitive cardinality	Resiliency		Primitive	Number of instances
	Efficient reduction	Lower bound		
$k = 2$	$t < n/3$	$t < n/3$	SC_2	$2\binom{n}{2}$
$k = 3$	$t < n/2$	$t < n/2$	OC_3/CC_3	$3\binom{n}{3}$
$4 \le k \le n-1$	$t < n/2$	$t < n-2$	OC_3/CC_3	$3\binom{n}{3}$
$k = n$	$t < n$	$t < n$	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*

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.

The case $k = 3$ is of special interest. We introduce two primitives: *oblivious cast* [Bla96], a natural generalization of oblivious transfer to the three-party case, 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.

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 that 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 $t < n$. These results are summarized in Table 1.

Multicast and "convergecast," with a single sender and a single recipient, respectively, constitute two natural communication models. We also show that no primitive that conforms to these types — 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 types, Table 1 "collapses" to two equivalence classes: $k = 2$ and $3 \le k \le n$.

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

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.

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.[1] 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 $[k,k]$.[2]

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." 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.

Security model. Several formal definitions of secure function evaluation exist (e.g., [Bea92,Can00,Gol00,GL90,MR92]). The process is assumed to be synchronous, a fact that simplifies the task of reasoning about security. In [Can00]

[1] 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.

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

(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 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 indistinguishable, except for a negligible function of the security parameter, independently of the adversary's computational power.
The trusted party τ is assumed to be equivalent to a probabilistic Turing machine with a memory tape of fixed (limited) size. This implies that τ can perform any task a standard computer can but not more. On the other hand, τ is also equivalent to an arithmetic circuit (though of potentially large size) and hence can be modeled as a (stateless) circuit. Thus, the multi-party computation specification simply defines a sequence of circuit evaluations on the players' inputs. Note that this ideal computation model, and hence the set of problems computable with an SFE protocol, is as strong as the "standard" one (e.g., [BGW88,CCD88,RB89]).

Reducibility and completeness. A main theme in this paper is that of reductions "across" cardinalities. The notion of reduction generalizes to the case of an n-ary function (n-player protocol) invoking another k-ary function (primitive of cardinality k, resp.), with $k \leq n$, in a natural way [Gol00]:

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.*[3]

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:

[3] 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).

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. (Note that this allows for parallel composition.) 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 $2^{-\sigma}$.

3 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 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. We now prove that, for $t \geq \lceil n/3 \rceil$, no primitive of cardinality 2 can be complete (if $n > 2$). 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_2-channels" for any two-party primitive g_2. 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*: If the sender is correct, then all correct recipients decide on the sender's input bit ($v = b$).*

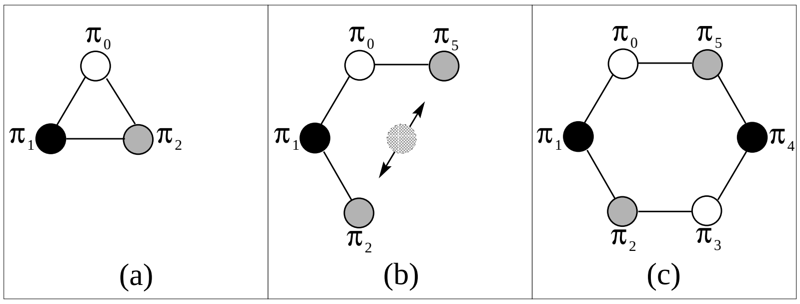

Fig. 1. *Rearrangement of processors in proof of Lemma 1*

We first consider the special case of $n = 3$ and $t \geq 1$, and then reduce the general case of $n \geq 3$ and $t \geq \lceil n/3 \rceil$ to this special case. The impossibility proof (for $n = 3$ and $t \geq 1$) is 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 authentic channels. In the new model, however, every pair of players can perform secure two-party computation. The idea in the proof is to assume that there exists an unconditionally secure broadcast protocol involving three players — interconnected by such a "g_2 channel", which then can be used to build a different system with contradictory behavior, hence proving that such a protocol cannot exist.

Lemma 1. *Let $n = 3$. For any two-player primitive connecting each pair of players, unconditionally secure broadcast is not possible if $t \geq 1$.*

Proof (sketch). Suppose, for the sake of contradiction, that there is a protocol that achieves broadcast for three players p_0, p_1, and p_2, with p_0 being the sender, even if one of the players is actively corrupted.
Let π_0, π_1, π_2 denote the players' corresponding processors with their local programs and, for each $i \in \{0, 1, 2\}$ let π_{i+3} be an identical copy of processor π_i; and let the (set of) given two-party primitive(s) between two processors π_i and π_j be called the *channel* between π_i and π_j.
Instead of connecting the original processors as required for the broadcast setting, we build a network involving all six processors (i.e., the original ones together with their copies) by arranging them in a circle, i.e., each processor π_i ($i \in \{0, \ldots, 5\}$) is connected (exactly) by one channel with $\pi_{(i-1) \bmod 6}$ and one with $\pi_{(i+1) \bmod 6}$.
We now prove that for every pair of adjacent processors π_i and $\pi_{(i+1) \bmod 6}$ in the new system and without the presence of an adversary, their common view is indistinguishable from their view as two processors $\pi_{i \bmod 3}$ and $\pi_{(i+1) \bmod 3}$ in the original system with respect to an adversary that corrupts the remaining proces-

sor $\pi_{(i+2) \bmod 3}$ in an admissible way.[4] Refer to Figure 1. The original system is depicted in Figure 1-(a). Let the processors π_0 and π_1 be correct. An admissible adversary strategy is to "split" π_2 and to make it behave independently with respect to π_0 and π_1 (Figure 1-(b)). Finally, by arranging the six processors in a circle as described above and shown in Figure 1-(c), this particular adversary strategy is simulated with respect to every pair π_i and $\pi_{(i+1) \bmod 6}$.

The new system involves two processors of the type corresponding to the sender, namely, π_0 and π_3, and these are the only processors that enter an input. Let now π_0 and π_3 be initialized with different inputs, i.e., let's assume that π_0 has input $v_0 \in \{0,1\}$ and that π_3 has input $v_3 = 1 - v_0$.[5]

We now show that there are at least two pairs of adjacent processors in the new system, i.e., one third among all six such 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 with respect to every pair on, wlog, the value v_0. Then the validity condition is violated with respect to both pairs involving processor π_3 since $v_3 \neq v_0$. On the other hand, suppose that the agreement condition is violated with respect to 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 $v_3 = 1 - v_0$, there must be some pair of adjacent processors $(\alpha, \beta) = (\pi_i, \pi_{(i+1) \bmod 6})$ that fails with a probability of at least $\frac{1}{3}$. 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 $(\alpha_0, \beta_0) = (\pi_i, \pi_{(i+1) \bmod 3})$ in the original system for one of the cases where the sender inputs either $v_0 = 0$ or $v_0 = 1$. Let Pr_0 be the probability that the sender selects input 0 in the original system. Then, in the original system, the adversary can force the pair (α_0, β_0) to be inconsistent with a probability of at least $\frac{1}{3}\min(\mathrm{Pr}_0, 1 - \mathrm{Pr}_0)$, which is non-negligible, since Pr_0 and $1 - \mathrm{Pr}_0$ are non-negligible by assumption. □

Theorem 1. *Let $n \geq 3$. For any primitive g_2, unconditionally secure broadcast is not possible if $t \geq \lceil n/3 \rceil$.*

Proof. Assume that there is an unconditionally secure broadcast protocol Π for $n \geq 3$ players and some $t \geq \lceil n/3 \rceil$ with arbitrarily small error probability $\varepsilon > 0$. Then we can let three players p_0, p_1, and p_2 each simulate up to $\lceil n/3 \rceil$ of the players in Π, with the sender in Π being simulated by p_0. Thus, this protocol among players $\{p_0, p_1, p_2\}$ achieves broadcast (with sender p_0) secure against one corrupted player because he simulates at most $\lceil n/3 \rceil$ of the players in Π, which is tolerated by assumption. Since this contradicts Lemma 1, the theorem follows. □

[4] I.e., for every pair of original processors, the rearrangement simultaneously simulates some particular adversary strategy by corrupting the third processor.

[5] We assume that any input value from $\{0,1\}$ will be selected by the sender with some non-negligible probability. Otherwise, the broadcast problem could be trivially solved for any $t < n$.

4 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 $t \geq \lceil n/2 \rceil$.
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.

4.1 $\mathbf{g_3[1,*]}$ 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 2. *Let $n \geq 3$. Then authentic channel reduces to oblivious cast for $t < n/2$.*

Proof (sketch). 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 $\mathrm{Pr}_{err}^{auth} = 2^{-\sigma}$. $\square$

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 2) 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.[6]

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

Proof. From Lemma 2 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].

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

[6] 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.

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 4. *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 $\Pr_{err} \le \max(\Pr_{err_1}, \Pr_{err_2}) = 2^{-\sigma}$. □

Lemma 4 together with the reduction of broadcast to weak 2-cast in [FM00] (which does not require pairwise channels) immediately yield

Lemma 5. *Broadcast among* $n \ge 3$ *players reduces to oblivious cast for* $t < n/2$.

Lemmas 3 and 5 and the constructions of [RB89,CDD+99] yield

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

4.2 $\mathrm{g}_3[*, 1]$ 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 (in several ways) — 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* $\mathcal{D}$*,* $|\mathcal{D}| \ge 3$*, 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 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, 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), using the corresponding key as input to a converge cast with p_1 as the recipient to communicate the desired bit.

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. $p_0, p_2 \xrightarrow{?} p_1$: $(s_0^{(x_0)}, *)$ (element-wise) (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`

The proof of the following lemma follows from elementary probability, independently of p_2's strategy:

Lemma 6. *Consider protocol* `Secure-Channel-Impl-2`*. If p_0 and p_1 are correct, then for every k, $k \in \{1, \ldots, \ell\}$,*

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

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

Lemma 7. *Let $n \geq 3$. Then secure channels 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 the protocol can fail is that either $x_0 = 0$ and $\mathcal{H}(s_1', s_1^{(0)}) \geq \frac{7}{12}\ell$ (probability 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:

- Pr_0: By Lemma 6(1), $s_1'[k] = s_1^{(0)}[k]$ holds with probability $\frac{1}{2}$. Hence, Pr_0 is the probability that out of ℓ trials with expected value $\frac{1}{2}$, at most $\frac{5}{12}l$ do match. We get $\mathrm{Pr}_0 \leq e^{-\frac{\ell}{4}(\frac{1}{6})^2} = e^{-\frac{\ell}{144}}$.

- Pr_1: By Lemma 6(2), $s'_1[k] = s_1^{(0)}[k]$ holds with probability $\frac{1}{3}$. Hence, Pr_1 is the probability that out of ℓ trials with expected value $\frac{1}{3}$, at least $1-\frac{7}{12} = \frac{5}{12}\ell$ do match. We get $\mathrm{Pr}_1 \leq e^{-\frac{\ell}{9}(\frac{1}{4})^2} = e^{-\frac{\ell}{144}}$.

Thus, the overall error probability is $\mathrm{Pr}_{err}^{auth} \leq \max(\mathrm{Pr}_0, \mathrm{Pr}_1) = e^{-\frac{\ell}{144}}$. □

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. Roughly, he protocol can be described 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 "$p_i \stackrel{!}{\longrightarrow} p_j$" denote the secure channel from p_i to p_j (by means of protocol `Secure-Channels-Impl-2`).

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

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. $p_1, p_2 \stackrel{?}{\longrightarrow} p_0$: $(s_1^{(0)}, s_2^{(0)})$; $(s_1^{(1)}, s_2^{(1)})$ (element-wise) (p_0 receives $s_0^{(0)}; s_0^{(1)}$)
3. $p_0 \stackrel{!}{\longrightarrow} p_i$ $(i = 1, 2)$: $x_0 \in \{0, 1\}$ (p_i receives $x_i \in \{0, 1\}$)
4. $p_0 \stackrel{!}{\longrightarrow} p_1$: $s_0^{(x_0)}$ (p_1 receives s'_1)
5. p_1: $\texttt{if}$ $\mathcal{H}(s'_1, s_1^{(x_1)}) < \frac{4}{5}\ell$ $\texttt{then}$ $y_1 = x_1$, $\texttt{else}$ $y_1 = \perp$ $\texttt{fi}$
6. $p_1 \stackrel{!}{\longrightarrow} p_2$: y_1; s'_1 (p_2 receives y_2; s'_2)
7. p_2: $\texttt{if}$ $(y_2 = \perp) \vee \left(\mathcal{H}(s'_2, s_2^{(y_2)}) > \frac{5}{4}\ell\right)$ $\texttt{then}$ $y_2 = x_2$ $\texttt{fi}$

Lemma 8. *Let $n \geq 3$. Then weak 2-cast reduces to converge cast for $t < n/2$.*

The proof appears in Section A.

As before, Lemmas 7 and 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.

4.3 Impossibility of $\mathbf{t \geq \lceil n/2 \rceil}$ for $\mathbf{g_3}$

We now show that no primitive of cardinality 3 can be complete with respect to half resiliency. We do so by generalizing the impossibility proof in [FM00] for broadcast with $t \geq \lceil n/2 \rceil$ using primitive 2-Cast, to arbitrary primitives of cardinality 3.

Theorem 4. *Let $n \geq 4$. For any primitive g_3, unconditionally secure broadcast is not possible if $t \geq \lceil n/2 \rceil$.*

Proof (sketch). Impossibility for $n = 4$ and $t = 2$: Suppose that there are four processors (with local programs) that achieve broadcast for $t = 2$. Again, we build a new system with the four processors and one copy of each, arranged in a circle. Analogously to Lemma 1, the 3-player primitives can be reconnected such that the view of any two adjacent processors is indistinguishable from their view in the original system (i.e., they can be reconnected in the same way as in the proof in [FM00]), and by assigning different inputs to the sender and its copy we get the same kind of contradiction.
Impossibility for $n > 4$ and $t \geq \lceil n/2 \rceil$: Suppose now that there are n processors and we want to achieve broadcast with $t \geq \lceil n/2 \rceil$. The processors are partitioned into four sets and each set is duplicated. Instead of reconnecting single processors, the connections between different sets are reconnected so that the common view of all the processors in two adjacent sets is indistinguishable from their view in the original system, and we get a contradiction along the lines of [FM00]. □

5 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 4). We then introduce a new primitive of type $[n, n]$, the *universal black box* (UBB_n), which allows for arbitrary resiliency ($t < n$). 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. Finally, we show that full cardinality is necessary to achieve arbitrary resiliency. We start with the impossibility results for restricted primitives.

5.1 $\mathbf{g_n[1,*]}$ and $\mathbf{g_n[*,1]}$ primitives

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

Proof (sketch). Assume that a particular primitive $g_n[1,*]$ 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 securely compute the OR function. Since there is only one input to g_n (to be given either by p or by q), there must be a first invocation of the primitive that reveals some input information to the other player. This is a contradiction to [BGW88], where it is shown that no player may reveal any information about his input to the other player unless he knows that the other player's input is 0. □

Theorem 6. *There is no $g_n[*,1]$ primitive complete for $t \geq \lceil n/2 \rceil$.*

Proof (sketch). The proof is again by contradiction. 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. $\square$

5.2 $\mathbf{g_n[n, n]}$ 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 simplicity, we first assume that exactly one function is to be computed on the inputs of all players, and that exactly one player, p_0, is to learn the result of the computation. The more general cases (multiple functions, multiple/different outputs) can be obtained by simple extensions to this case.

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

- the *function input*, wherein 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 per computation.

Theorem 7. *The universal black box is a complete primitive for $t < n$.*

Proof (sketch). We show that privacy and correctness hold for arbitrary t.

Privacy: Trivially, no $p_i \neq p_0$ learns anything. On the other hand, p_0's output can give information about player p_i's argument input only if p_i entered the same function input as p_0 (which means that p_i had "agreed" on exactly this

computation). Hence, p_0 would get the same information about p_i's argument as in an ideal process involving a trusted party. If p_0 is corrupted and inputs a wrong function input, no argument from a correct player will be used for this computation.
Correctness: The function to be computed is selected by p_0. Hence, if he's correct, the UBB_n does compute the desired function. Corrupted players that input a different function only achieve that their input be replaced by a default value — a strategy that is also (easily) achievable in an ideal process by selecting the default argument. □

Corollary 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.*

The single-output version of a UBB_n can be generalized to a multi-output UBB_n by the following modification. The function input specifies n functions to be computed on the inputs — one function per player. The function f_i 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 are considered by the UBB_n who agree on the same n functions $f_1, \ldots, f_n$ to be computed with respect to the n players, i.e., whose function inputs match with the function input of p_i.

Finally, we prove that full cardinality is necessary in order to achieve arbitrary resiliency (proof in Section A):

Theorem 8. *For $k < n$, there is no primitive g_k complete for $t < n$.*

Moreover, there is strong evidence that even a primitive of the most powerful category for cardinalities $k < n$, i.e., a $g_{n-1}[n-1, n-1]$, cannot be complete for $t \geq \lceil n/2 \rceil$, but we have no formal proof for it.

Conjecture 1. For $k < n$, there is no primitive g_k complete for $t \geq \lceil n/2 \rceil$.

6 Summary and Open Problems

Originally (Section 2), 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 1 and from the constructions for the proofs of Theorems 2, 3, and 7 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 2. *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 $\mathrm{SC}_2[1,1]$ and no g_2 can do any better; and, for $t < n/2$, there are complete primitives $\mathrm{OC}_3[1,2]$ and $\mathrm{CC}_3[2,1]$ and no g_k, $k \leq n$, can do any better. For the case of type $[k,k]$ it remains to prove Conjecture 1, 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 $k=n$ $(t<n)$.

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 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 Info-hiding*, 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.

[Chv79] V. Chvátal. The tail of the hypergeometric distribution. *Discrete Mathematics*, 25:285–287, 1979.

[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.

[Gol00] 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 Proofs

We repeat the statements here for convenience.

Lemma 8. *Let $n \geq 3$. Then weak 2-cast reduces to converge cast for $t < n/2$.*

In the proof we will be using the following two lemmas. Their validity follows from elementary probability.

Lemma 9. *The probability that in protocol* `Weak-2-Cast-Impl-2` p_0 *receives at least $\frac{9}{8}\ell$ elements of $s_1^{(x_0)}$ and at most $\frac{7}{8}\ell$ elements of $s_2^{(x_0)}$, or vice versa, is* $\Pr_1 \leq 2e^{-\frac{\ell}{128}}$.

Lemma 10. *Given is a set of pairs* $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ *with elements* $x_i, y_i \in \{0, 1, 2\}$. *If at least all elements* x_i *or all elements* y_i *are selected uniformly at random from* $\{0, 1, 2\}$, *then the following holds:*

(1) *If* $|S| \geq \frac{9}{8}\ell$, *then the probability that there are* $\frac{13}{40}\ell$ *or less indices* i *such that* $x_i = y_i$ *is* $\Pr_2 \leq e^{-\frac{\ell}{300}}$.

(2) *If* $|S| \leq \frac{7}{8}\ell$, *then the probability that there are* $\frac{7}{20}\ell$ *or more indices* i *such that* $x_i = y_i$ *is* $\Pr_3 \leq e^{-\frac{\ell}{258}}$.

Proof of Lemma 8. 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. Assume that p_0 receives at least $\frac{7}{8}\ell$ elements from each of the players' key during step 2 (by Lemma 9, this happens with probability at least $1 - \Pr_1$). 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 $\perp$ ($\mathcal{H}(s'_1, s_1^{(x_1)}) \geq \frac{4}{5}\ell$); i.e., that at most $\frac{6}{5}\ell$ elements of s'_1 match with p_1's key. We assume that p_0 receives at least $k \geq \frac{7}{8}\ell$ elements of p_1's key during step 2, $k = \frac{7}{8}\ell$ in the worst case. Hence, of all other $2\ell - \frac{7}{8}\ell = \frac{9}{8}\ell$ elements (i.e., the elements of s'_1 originating from p_2), at most $\frac{6}{5}\ell - \frac{7}{8}\ell = \frac{13}{40}\ell$ are identical to the element that p_1 chose for the same invocation of "$p_1, p_2 \xrightarrow{?} p_0$". By Lemmas 9 and 10, this happens with probability $\Pr_4 \leq \Pr_1 + \Pr_2 \leq 3e^{\frac{\ell}{300}}$.

• 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 $\mathcal{H}(s'_2, s_2^{(y_2)}) \leq \frac{5}{4}\ell$ before step 7; i.e., that at least $\frac{3}{4}\ell$ elements of s'_2 match p_2's key. But since p_1 does not learn anything about the elements in $s_2^{(y_2)}$, he must guess $\frac{3}{8}$ or more of those 2ℓ elements correctly — otherwise p_2 would decide on x_0. This probability can be estimated by Chernoff bounds (2ℓ trials with expected value $\frac{1}{3}$) and we get $\Pr_5 \leq e^{-\frac{\ell}{288}}$. (In fact, this holds independently of the assumption at the beginning of the proof.)

• p_0 *corrupted.* Since p_1 and p_2 are correct, the following equalities hold before step 7: $s'_1 = s'_2 \stackrel{\text{def}}{=} s'$ and $x_1 = y_1 = y_2 \stackrel{\text{def}}{=} y$. In order to achieve that p_1 and p_2 decide on different bits p_0 must select $x_2 = 1 - y$ and achieve that $\mathcal{H}(s', s_1^{(y)}) < \frac{4}{5}\ell$ and $\mathcal{H}(s', s_2^{(y)}) > \frac{5}{4}\ell$; i.e., p_0 must prepare s' such that at least $\frac{6}{5}\ell$ elements still match p_1's key but at most $\frac{3}{4}\ell$ elements match p_2's key. Given that the CC_3 statistics are good (which happens with probability $1 - \Pr_1$; see above) $s_0^{(y)}$ contains at most $\frac{9}{8}\ell$ elements originating from player p_1 and at least $\frac{7}{8}\ell$ elements originating from player p_2. In the sequel we assume that these quantities exactly hold, which constitutes the best case for p_0 to succeed (maximal number of matches with p_1's key and minimal number for p_2's).

Suppose now that s' is selected such that $h = \mathcal{H}(s', s_0^{(y)}) > \frac{3}{4}\ell$. We show that then $\mathcal{H}(s', s_1^{(y)}) < \frac{4}{5}\ell$ cannot be achieved almost certainly. By Lemma 10, with probability $1 - \Pr_3$, there are at most $\frac{7}{20}\ell$ elements in s' that match $s_1^{(y)}$ at positions where the corresponding element of $s_0^{(y)}$ was actually received from

p_2. Hence at least $x > (2-\frac{4}{5})\ell - \frac{7}{20}\ell = \frac{17}{20}\ell$ elements of s' must match $s_1^{(y)}$ at positions where the corresponding element was actually received from p_1; i.e., of the $h > \frac{3}{4}\ell$ differences between s' and $s_0^{(y)}$, at most $y = \frac{9}{8}\ell - x < \frac{11}{40}\ell$ may be made up at positions where the corresponding element was received from p_1. The probability of this event can be estimated by Hoeffding bounds [Chv79] (Hypergeometric distribution; $N = 2\ell$ elements, $n = \frac{3}{4}\ell$ trials, $K = \frac{9}{8}\ell$ "good" elements, less than $k = \frac{11}{40}\ell$ hits), and we get $\Pr_a < e^{-\frac{\ell}{18}}$.

Suppose now that s' is selected such that $h = \mathcal{H}(s', s_0^{(y)}) \leq \frac{3}{4}\ell$. We show that then $\mathcal{H}(s', s_2^{(y)}) > \frac{5}{4}\ell$ cannot be achieved almost certainly. By Lemma 10, with probability $1 - \Pr_2$, there are at least $\frac{13}{40}\ell$ elements in s' that match $s_2^{(y)}$ at positions where the corresponding element of $s_0^{(y)}$ was actually received from p_1. Hence at most $x < (2-\frac{5}{4})\ell - \frac{13}{40}\ell = \frac{17}{40}\ell$ elements of s' may match $s_2^{(y)}$ at positions where the corresponding element was actually received from p_2; in other words, of the $h \leq \frac{3}{4}\ell$ differences between s' and $s_0^{(y)}$, at least $y = \frac{7}{8}\ell - x > \frac{9}{20}\ell$ must be made up at positions where the corresponding element was received from p_2. The probability of this event can be estimated by the Hoeffding bound (hypergeometric distribution; $N = 2\ell$ elements, $n = \frac{3}{4}\ell$ trials, $K = \frac{7}{8}\ell$ "good" elements, more than $k = \frac{9}{20}\ell$ hits), giving $\Pr_a < e^{-\frac{\ell}{26}}$.

Hence, in the case where p_0 is corrupted, we get an error probability of at most $\Pr_6 \leq \Pr_1 + \max(\Pr_a + \Pr_3, \Pr_b + \Pr_2) \leq 4e^{\frac{\ell}{300}}$.

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 $\Pr_{SC} \leq e^{-\frac{\ell}{300}}$. Thus, the overall error probability of the weak 2-cast construction is at most

$$\Pr_{err} \leq \Pr_{SC} + \max(\Pr_4, \Pr_5, \Pr_6) \leq 5e^{-\frac{\ell}{300}} .$$

For security parameter σ, we let $\ell \geq 300(\sigma + \lceil \ln 5 \rceil)$, and hence $\Pr_{err} \leq e^{-\sigma}$. $\square$

Theorem 8. *For $k < n$, there is no primitive g_k complete for $t < n$.*

Proof (sketch). Consider a UBB of cardinality $k = n-1$. Since it is complete for any computation among $n-1$ players, it can securely simulate the functionality of any k-player primitive with $k < n$; i.e., the existence of any k-player primitive ($k < n$) complete for $t < n$ would imply that the UBB_{n-1} is also complete. 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 black box 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 $t = n-2$, i.e., in the presence of exactly two correct players.[7] Let $\pi_0, \ldots, \pi_{n-1}$ 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

[7] 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.

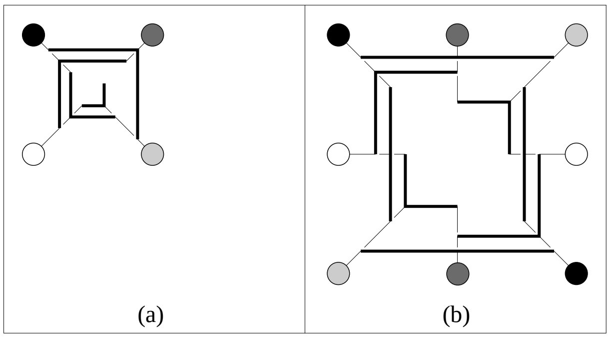

Fig. 2. *Reconnection of processors in the proof of Theorem 8: special case $n = 4$.*

of processor π_i. For every processor π_k, $k \in \{0, \ldots, 2n-1\}$, let the number ($k \bmod n$) be called the *type* of p_k. Similarly to the proof of Lemma 1, we now build a new system involving all $2n$ processors but, instead of reconnecting them with pairwise channels, the instances of BB_{n-1} have to be reconnected in such a way that, again, for each pair of adjacent processors, π_i and $\pi_{(i+1) \bmod 2n}$, their view in the new system is indistinguishable from their view in the original system, for some particular strategy of an adversary corrupting all the remaining $n-2$ processors.

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 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 BB_{n-1} with a processor of type $\pi_{(i\pm1) \bmod n}$ (i.e., an adjacent type in the original system), then it shares it with the concrete processor $\pi_{(i\pm1) \bmod 2n}$ (i.e., its adjacent processor of this type in the new system).

This can be achieved by applying the following rule for every processor π_i, $i \in \{0, \ldots, 2n-1\}$. For each δ, $\delta = 1, \ldots, n-1$, there is a BB_{n-1} that originally connects $\pi_{i \bmod n}$ with all other processors but $\pi_{(i+\delta) \bmod n}$. For every such δ, exactly one BB_{n-1} now connects π_i with processors $\pi_{(i+1) \bmod 2n}, \ldots, \pi_{(i+\delta-1) \bmod 2n}$ and $\pi_{(i-1) \bmod 2n}, \ldots, \pi_{(i-n+\delta+1) \bmod 2n}$. This principle is depicted in Figure 2 for the special case of $n = 4$ and BB_3.

Now the proof proceeds as the proof of Lemma 1 by assigning different input values to both sender processors, and concluding that the broadcast conditions are not satisfied with respect to at least 2 of the $2n$ pairs of adjacent processors. This contradicts the assumption that broadcast is possible with an arbitrarily small error probability. $\square$