

Secure Message Transmission with Small Public Discussion

Juan Garay¹, Clint Givens², and Rafail Ostrovsky^{3*}

¹ AT&T Labs – Research. garay@research.bell-labs.com

² Department of Mathematics, UCLA. cgivens@math.ucla.edu

³ Departments of Computer Science and Mathematics, UCLA. rafail@cs.ucla.edu

Abstract. In the problem of Secure Message Transmission in the public discussion model (SMT-PD), a Sender wants to send a message to a Receiver privately and reliably. Sender and Receiver are connected by n channels, up to $t < n$ of which may be maliciously controlled by a computationally unbounded adversary, as well as one public channel, which is reliable but not private.

The SMT-PD abstraction has been shown instrumental in achieving secure multi-party computation on sparse networks, where a subset of the nodes are able to realize a broadcast functionality, which plays the role of the public channel. However, the *implementation* of such public channel in point-to-point networks is highly costly and non-trivial, which makes minimizing the use of this resource an intrinsically compelling issue.

In this paper, we present the first SMT-PD protocol with *sublinear* (i.e., logarithmic in m , the message size) communication on the public channel. In addition, the protocol incurs a private communication complexity of $O(\frac{mn}{n-t})$, which, as we also show, is *optimal*. By contrast, the best known bounds in both public and private channels were linear. Furthermore, our protocol has an optimal round complexity of $(3, 2)$, meaning three rounds, two of which must invoke the public channel.

Finally, we ask the question whether some of the lower bounds on resource use for a single execution of SMT-PD can be beaten *on average* through amortization. In other words, if Sender and Receiver must send several messages back and forth (where later messages depend on earlier ones), can they do better than the naïve solution of repeating an SMT-PD protocol each time? We show that amortization can indeed drastically reduce the use of the public channel: it is possible to limit the total number of uses of the public channel to *two*, no matter how many messages are ultimately sent between two nodes. (Since two uses of the public channel are required to send any reliable communication whatsoever, this is best possible.)

1 Introduction

Dolev, Dwork, Waarts and Yung [DDWY93] introduced the model of *Secure Message Transmission* (SMT) in an effort to understand the connectivity requirements for secure

* Supported in part by IBM Faculty Award, Xerox Innovation Group Award, the Okawa Foundation Award, Intel, Teradata, NSF grants 0716835, 0716389, 0830803, 0916574, BSF grant 2008411 and U.C. MICRO grant.

communication in the information-theoretic setting. Generally speaking, an SMT protocol involves a sender, \mathcal{S} , who wishes to transmit a message M to a receiver, \mathcal{R} , using a number n of channels (“wires”), some of which are controlled by a malicious adversary \mathcal{A} . The goal is to send the message both *privately* and *reliably*. Since its introduction, SMT has been widely studied and optimized with respect to several different settings of parameters (for example, see [SA96,SNP04,ACH06,FFGV07,KS08]).

Garay and Ostrovsky [GO08] studied a model they called Secure Message Transmission by *Public Discussion* (SMT-PD) as an important building block for achieving secure multi-party computation [BGW88,CCD88] on sparse (i.e., not fully connected) networks. (An equivalent setup was studied earlier in a different context by Franklin and Wright [FW98].) In this model, in addition to the wires in the standard SMT formulation, called “common” or “private” wires from now on, \mathcal{S} and \mathcal{R} gain access to a *public* channel which the adversary can read but not alter. In this new setting, secure message transmission is achievable even if the adversary corrupts up to $t < n$ of the private wires—i.e., up to all but one.

The motivation for this abstraction comes from the feasibility in partially connected settings for a subset of the nodes in the network to realize a broadcast functionality despite the limited connectivity [DPPU86,Upf92,BG93]¹, which plays the role of the public channel. (The private wires would be the multiple paths between them.) As such, the *implementation* of the public channel in point-to-point networks is costly and highly non-trivial in terms of rounds of computation and communication, as already the sending of a single message to a node that is not directly connected is simulated by sending the message over multiple paths, not just blowing up the communication but also incurring a slowdown factor proportional to the diameter of the network, and this is a process that must be repeated many times—linear in the number of corruptions for deterministic, error-free broadcast protocols (e.g., [GM98]), or expected (but high) constant for randomized protocols [FM97,KK06].

A main goal of this work is to minimize the use of this expensive resource, both in terms of communication as well as in the number of times it must be used when sender and receiver must send many messages back and forth, as it is the case in secure multi-party computation. We first present an SMT-PD protocol with a logarithmic (in m , the message size) communication complexity on the public channel; the best known bound, due to Shi, Jiang, Safavi-Naini, and Tuhin [SJST09], was linear (see related work below). In addition, our protocol incurs a private communication complexity of $O(\frac{mn}{n-t})$, which, as we also show, is *optimal*, thus providing an affirmative answer to the question posed in [SJST09] of whether their $O(mn)$ private communication could be improved. Furthermore, our protocol has an optimal round complexity of $(3, 2)$, meaning 3 rounds, 2 of which must invoke the public channel [SJST09].

Regarding the number of times the public channel must be used when considering SMT-PD as a subroutine in a larger protocol, we ask the question whether some of the lower bounds on resource use for a single execution of SMT-PD can be beaten *on average* through amortization. In other words, if a sender and receiver must send several messages back and forth (where later messages depend on earlier ones), can they do better than the naïve solution of repeating an SMT-PD protocol each time, incurring

¹ Called “almost-everywhere” agreement, or broadcast, in this setting.

a cost of three rounds and two public channel transmissions per message? We show that amortization can in fact drastically reduce the use of the public channel: indeed, it is possible to limit the total number of uses of the public channel to *two*, no matter how many messages are ultimately sent between two nodes. (Since two uses of the public channel are required to send any reliable communication whatsoever, this is best possible.)

Prior work. The first variant of SMT considered in the literature is *perfectly secure message transmission* (PSMT), in which both privacy and reliability are perfect [DDWY93]. It is shown in the original paper that PSMT is possible if and only if $n \geq 2t + 1$. For such n , 2 rounds are necessary and sufficient for PSMT, while one-round PSMT is possible if and only if $n \geq 3t + 1$.

The communication complexity of PSMT depends on the number of rounds. For 1-round PSMT, Fitzi *et al.* [FFGV07] show that transmission rate $\geq \frac{n}{n-3t}$ is necessary and sufficient. (Recall that $n > 3t$ is required in this case.) For 2-round PSMT, Srinathan *et al.* [SNP04] show that a transmission rate $\geq \frac{n}{n-2t}$ is required²; this was extended in [SPR07], which showed that increasing the number of rounds does not help. Kurosawa and Suzuki [KS08] construct the first efficient (i.e., polynomial-time) 2-round PSMT protocol which matches this optimal transmission rate.

A number of relaxations of the perfectness requirements of PSMT are considered in the literature to achieve various tradeoffs (see for example [CPRS08] for a detailed discussion of variants of SMT). The most general version of SMT (or SMT-PD) is perhaps (ϵ, δ) -SMT. We call a protocol for SMT(-PD) an (ϵ, δ) -SMT(-PD) protocol provided that the adversary’s advantage in distinguishing any two messages is at most ϵ , and the receiver correctly outputs the message with probability $1 - \delta$. The lower bound $n \geq 2t + 1$ holds even in this general setting (at least for non-trivial protocols, such as those satisfying $\epsilon + \delta < 1/2$); hence the most interesting case for SMT-PD is the case when the public channel is required: $t < n \leq 2t$. As noted above, this requires round complexity $(3, 2)$ [SJST09]. Franklin and Wright [FW98] show that perfectly reliable ($\delta = 0$) SMT-PD protocols are impossible when $n \leq 2t$. On the other hand, perfect privacy ($\epsilon = 0$) is possible, and is achieved by previous SMT-PD constructions (see below).

The communication complexity lower bounds noted above all apply to PSMT; for more general SMT bounds, we are aware only of [KS07]. They consider the problem of *almost-secure message transmission*, which is only slightly less restrictive than PSMT. Namely, the problem requires perfect privacy, and that the Receiver *never* output an incorrect message, though he may output “failure” with probability δ . The authors show that in this model, there is a communication complexity lower bound of $n(m + \log(1/\delta))$ (up to an additive constant).

A number of protocols for SMT-PD appear in previous work. The first such comes in [FW98] as a consequence of the equivalence shown there between networks with multicast and those with simple lines and broadcast (i.e., the public discussion model). Their solution has optimal round complexity $(3, 2)$ ³; however, when $t < n < \lceil \frac{3t}{2} \rceil$

² The authors claim a matching upper bound as well, but this was shown to be flawed [ACH06].

³ The round complexity is not apparent from the text, for two reasons: (1) The protocol is described in terms of the multicast model, not SMT-PD directly; and (2) the authors consider

(including the worst case $t = n + 1$), their protocol has (pick your poison) either positive privacy error $\epsilon > 0$, or *exponential* communication complexity. Garay and Ostrovsky [GO08] first describe a (4,3)-round $(0, \delta)$ protocol which was subsequently improved to (3,2) rounds. The protocol has linear transmission rate (in terms of message size) on the public and private channels. Shi *et al.* [SJST09] give the first protocol with constant transmission rate on the public channel (for messages of sufficient, modest size) with linear transmission rate on the private channels as well; however, the *communication complexity* of their protocol is linear.

Our contributions. By contrast, we obtain the first round-optimal SMT-PD protocol with *sublinear* (logarithmic) communication complexity on the public channel. More specifically (and assuming for simplicity $\delta = O(1)$), our protocol has public channel communication complexity $O(n \log n \log m)$ for messages of sufficient size, as compared with $O(m)$ in the protocol of [SJST09]. (The message size required by either protocol—namely, $m/\log m = \Omega(n \log n)$ for ours, or $m = \Omega(n^2)$ for that of [SJST09]—ensures that $O(n \log n \log m)$ improves over $O(m)$ for relevant values of n, m .) The protocol also enjoys a private communication complexity of $O(\frac{nm}{n-t})$, which (just by itself) improves on previous constructions and, as we also show, is optimal. At a high level, the protocol has the same structure as previous 3-round SMT-PD protocols, with the following important differences: (1) our use of randomness extractors allows us to reduce the amount of transmitted randomness, which is reflected in the gain in private communication, and (2) typically in previous protocols the message is transmitted in the last round over the public channel, blinded by the private randomness thought not to have been tampered with; our improvement to public communication comes from the transmission of the (blinded) message on the *private* wires, provided that the sender *authenticates* the transmission making use of the public channel, which in turn requires smaller communication. Additionally, we achieve these improved communication bounds even for messages of smaller required size than Shi *et al.* [SJST09].⁴ Finally, the protocol achieves perfect privacy.

We arrive at this result through a series of transformations. First, we design a generic SMT-PD protocol with linear public communication and $O(\frac{nm}{n-t})$ private communication (note that this already improves on existing results); second, we consider instantiations of the generic protocol’s “black boxes” with different randomness extractors, each providing its own benefits (perfect privacy *vis-à-vis* smaller message size); and last, we obtain the final protocol by essentially running two perfect-privacy instantiations of the generic protocol in parallel, one for the message itself and a “smaller” version for the authentication key. These results are presented in Section 3.

As noted above, we also show (Section 4) an $\Omega(\frac{nm}{n-t})$ lower bound on private communication. The lower bound holds for SMT without public discussion as well. The bound itself is weaker than previous, but it holds for a more general class of SMT protocols. In particular, it is the first communication complexity lower bound to consider

synchronous “rounds” not in the abstract SMT-PD model, but in the more concrete setting of nodes relaying messages in the underlying network.

⁴ Specifically, [SJST09] require message size $m = \Omega(n^2(\log(1/\delta))^2)$, where we require only $m = \Omega(n(\log n + \log(1/\delta)) \log q)$, with $q \approx mn/(n-t)$.

non-perfect privacy, as well as the first to allow for the Receiver outputting an incorrect message.

Finally, we show in Section 5 how amortization can drastically reduce the use of the public channel, allowing sender and receiver to communicate *indefinitely* after using the public channel twice and a limited initial message. Our approach is to separate Sender and Receiver’s interaction following the first execution of SMT-PD into two modes: a *Normal Mode* and a *Fault-Recovery Mode*. At a high level, in the Normal Mode, secure communication is successful provided the adversary does not interfere; this is implemented by a one-round protocol satisfying a relaxed version of the problem that we call *Weak SMT-PD*. Fault-Recovery Mode is entered if corruption is detected.⁵

Preliminaries and definitions are given in Section 2. Due to space limitations, most of the proofs, as well as additional background material, are given in the full version of the paper [GGO09].

2 Model and Preliminaries

Definition 1. If X and Y are random variables over a discrete space S , the statistical distance between X and Y is defined to be

$$\Delta(X, Y) \stackrel{\text{def}}{=} \frac{1}{2} \sum_{s \in S} |\Pr[X = s] - \Pr[Y = s]|.$$

We say that X and Y are ϵ -close if $\Delta(X, Y) \leq \epsilon$.

The public discussion model. The *public discussion model* for secure message transmission [GO08] consists of a Sender \mathcal{S} and Receiver \mathcal{R} (PPTMs) connected by n communication channels, or *wires*, and one *public channel*. \mathcal{S} wishes to send a message $M_{\mathcal{S}}$ from message space \mathcal{M} to \mathcal{R} , and to this end \mathcal{S} and \mathcal{R} communicate with each other in synchronous rounds in which one player sends information across the wires and/or public channel. Communication on the public channel is reliable but public; the common wires may be corrupted and so are not necessarily reliable or private.

\mathcal{A} is a computationally unbounded adversary who seeks to disrupt the communication and/or gain information on the message. \mathcal{A} may *adaptively* corrupt up to $t < n$ of the common wires (potentially all but one!). Corrupted wires are actively controlled by \mathcal{A} : he can eavesdrop, block communication, or place forged messages on them. Further, we assume \mathcal{A} is *rushing*—in each round, he observes what is sent on the public channel and all corrupted wires before deciding what to place on corrupted wires, or whether to corrupt additional wires (which he then sees immediately).

An *execution* E of an SMT-PD protocol is determined by the random coins of \mathcal{S} , \mathcal{R} , and \mathcal{A} (which we denote $C_{\mathcal{S}}$, $C_{\mathcal{R}}$, $C_{\mathcal{A}}$ respectively), and the message $M_{\mathcal{S}} \in \mathcal{M}$. The *view of a player* $\mathcal{P} \in \{\mathcal{S}, \mathcal{R}, \mathcal{A}\}$ in an execution E , denoted $\text{View}_{\mathcal{P}}$, is a random variable consisting of \mathcal{P} ’s random coins and all messages received (or overheard) by \mathcal{P} .

⁵ Effectively, this is an instantiation in the SMT context of the “fast-track” approach (e.g., [Lam87,GRR98]), where if things are “smooth” then the algorithm or protocol performs very efficiently, reverting to a more punctilious mode otherwise.

(\mathcal{S} 's view also includes $M_{\mathcal{S}}$). Additionally, let $\text{View}_{\mathcal{P}}(M_0)$ denote the distribution on $\text{View}_{\mathcal{P}}$ induced by fixing $M_{\mathcal{S}} = M_0$. In each execution, \mathcal{R} outputs a received message $M_{\mathcal{R}}$, a function of $\text{View}_{\mathcal{R}}$.

We can now define an (ϵ, δ) -SMT-PD protocol (cf. [FW98,GO08,SJST09]):

Definition 2. A protocol Π in the model above, in which \mathcal{S} attempts to send a message $M_{\mathcal{S}}$ to \mathcal{R} , is (ϵ, δ) -secure (or simply, is an (ϵ, δ) -SMT-PD protocol) if it satisfies:

PRIVACY: For any two messages $M_0, M_1 \in \mathcal{M}$, $\text{View}_{\mathcal{A}}(M_0)$ and $\text{View}_{\mathcal{A}}(M_1)$ are ϵ -close.

RELIABILITY: For all $M_{\mathcal{S}} \in \mathcal{M}$ and all adversaries \mathcal{A} , \mathcal{R} should correctly receive the message with probability at least $1 - \delta$; i.e., $\Pr[M_{\mathcal{R}} = M_{\mathcal{S}}] \geq 1 - \delta$. (The probability is taken over all players' random coins.)

Error-correcting codes and consistency checks for codewords. For our purposes, the following definition of error-correcting codes is sufficient:

Definition 3. Given a finite alphabet Σ , an error-correcting code \mathcal{E} of minimum distance d is a pair of mappings $\text{Enc} : \Sigma^K \rightarrow \Sigma^N$, where $K < N$ and $\text{Dec} : \Sigma^N \rightarrow \Sigma^K$, such that (1) any two distinct elements x, y in the image of Enc (the codewords) have $\text{dist}(x, y) \geq d$ in the Hamming metric; (2) $\text{Dec}(\text{Enc}(x)) = x$ for all $x \in \mathbb{F}_q^K$.⁶ We say \mathcal{E} has rate K/N and relative minimum distance d/N .

We require a family of codes of increasing input length which is *asymptotically good*, that is, \mathcal{E} should have *constant* rate and *constant* relative minimum distance D . See, e.g., [MS83] for a standard reference.

Of particular interest for us are the well-known Reed-Solomon codes over F_q , obtained by oversampling polynomials in $\mathbb{F}_q[X]$. Given an input in \mathbb{F}_q^K , we interpret it as a polynomial f of degree $\leq K - 1$; to obtain a codeword from f , we simply evaluate it at N distinct points in \mathbb{F}_q , for any $N > K$. Indeed, any two such polynomials agree on at most $K - 1$ points, therefore the Reed-Solomon code has minimum distance $N - K + 1$.

Our protocols make use of a simple method to probabilistically detect when codewords sent on the private wires are altered by \mathcal{A} . Simply put, the sender of the codeword reveals a small subset of the codeword symbols. Formally, suppose \mathcal{S} sends a codeword $\mathcal{C} \in \Sigma^N$ to \mathcal{R} over one of the private wires, and \mathcal{R} receives the (possibly altered) codeword \mathcal{C}^* . (If \mathcal{R} receives a non-codeword, he immediately rejects it.) Then to perform the consistency check, \mathcal{S} chooses a random set $J = \{j_1, j_2, \dots, j_\ell\} \subset [N]$ and sends $(J, \mathcal{C}|_J)$ to \mathcal{R} , where $\mathcal{C}|_J$ represents the codeword \mathcal{C} restricted to the indices in J . If the revealed symbols match, then the consistency check succeeds; otherwise the check fails and \mathcal{R} rejects \mathcal{C}^* as tampered.

Suppose \mathcal{A} alters \mathcal{C} to a different codeword, $\mathcal{C}^* \neq \mathcal{C}$. Since \mathcal{C} and \mathcal{C}^* are distinct valid codewords, they differ in at least, say, $1/3$ of their symbols. Therefore, the probability that they agree on a randomly chosen index is $\leq 2/3$, and so

$$\Pr[\mathcal{R} \text{ accepts } \mathcal{C}^*] = \Pr[\mathcal{C}|_J = \mathcal{C}^*|_J] \leq (2/3)^\ell.$$

⁶ Note in particular that this allows us to test for membership in the image $\text{Enc}(\Sigma^K)$ by first decoding and then re-encoding.

Thus, with probability $\geq 1 - (2/3)^\ell$, \mathcal{R} will reject a tampered codeword. Of course, the validity of the check depends upon \mathcal{A} not knowing J at the time of potential corruption of \mathcal{C} .

Average min-entropy and average-case randomness extractors. Recall that the *min-entropy* of a distribution $X = (X_1, \dots, X_N)$ over $\{0, 1\}^N$ is defined as

$$H_\infty(X) = \min_x (-\log(\Pr[X = x])),$$

and gives a measure of the amount of randomness “contained” in a weakly random source. We say a distribution X is a k_{min} -source if $H_\infty(X) \geq k_{min}$.

A (*seeded*) $(N, M, k_{min}, \epsilon)$ -strong extractor is a (deterministic) function

$$\text{Ext} : \{0, 1\}^N \times \{0, 1\}^D \rightarrow \{0, 1\}^M$$

such that for any k_{min} -source X , the distribution $U_D \circ \text{Ext}(X, U_D)$ is ϵ -close to $U_D \circ U_M$ (where U_k represents the uniform distribution on $\{0, 1\}^k$). The input to the extractor is the N -bit k_{min} -source, X , together with a truly random seed s , which is uniformly distributed over $\{0, 1\}^D$. Its output is an M -bit string which is statistically close to uniform, *even conditioned on the seed s used to generate it*.

This notion of min-entropy, and of a general randomness extractor, may be an awkward fit when considering an adversary with side information Y as above. In these cases, a more appropriate measure may be found in the *average min-entropy* of X given Y , defined in [DORS08] by

$$\tilde{H}_\infty(X | Y) = -\log\left(\mathbb{E}_{y \leftarrow Y} \left[\max_x \Pr[X = x | Y = y] \right]\right).$$

Note that this definition is based on the *worst-case* probability for X , conditioned on the *average distribution* (as opposed to worst-case probability) of Y . The rationale is that Y is assumed to be outside of the adversary’s control; however, once Y is known, the adversary then predicts the *most likely* X , given that particular Y .

[DORS08] use average min-entropy to define an object closely related to extractors: A (*seeded*) *average-case* $(N, M, k_{min}, \epsilon)$ -strong extractor is a (deterministic) function

$$\text{Ext} : \{0, 1\}^N \times \{0, 1\}^D \rightarrow \{0, 1\}^M$$

such that the distribution of $(U_D \circ \text{Ext}(X, U_D), I)$ is ϵ -close to $(U_D \circ U_M, I)$, whenever (X, I) is a jointly distributed pair satisfying $\tilde{H}_\infty(X | I) \geq k_{min}$. The similarity to an ordinary extractor is clear. [DORS08] prove the following fact about average min-entropy:

Fact 4 *If Y has at most 2^ℓ possible values, then $\tilde{H}_\infty(X | (Y, Z)) \geq \tilde{H}_\infty(X | Z) - \ell$.*

Extracting randomness from \mathbb{F}_q . We will make use of a special-purpose *deterministic* (seedless) extractor Ext_q which operates at the level of field elements in \mathbb{F}_q as opposed to bits. Ext_q works not on general min-entropy sources, but on the restricted class of *symbol-fixing sources*, which are strings in \mathbb{F}_q^N such that some subset of K symbols is distributed independently and uniformly over \mathbb{F}_q , while the remaining $N - K$ symbols

are fixed. Given a sample from any such source, Ext_q outputs K field elements which are uniformly distributed over \mathbb{F}_q^K .

Ext_q works as follows: Given $\alpha \in \mathbb{F}_q^N$, construct $f \in \mathbb{F}_q[X]$ of degree $\leq N - 1$, such that $f(i) = \alpha_i$ for $i = 0, \dots, N - 1$. Then $\text{Ext}_q(\alpha) = (f(N), f(N + 1), \dots, f(N + K - 1))$. (Of course we require $N + K \leq q$.) This extractor has proven useful in previous SMT protocols as well (see, e.g., [ACH06,KS08]).

3 SMT-PD with Small Public Discussion

In this section we present our main positive results. First, we construct a basic (ϵ, δ) -SMT-PD protocol, Π_{Gen} (for “generic”), with optimal private communication and linear public communication. We then consider possible instantiations of Π_{Gen} ; using, in particular, Reed-Solomon codes and the extractor Ext_q , improves it to a 0-private protocol. Finally, we use Π_{Gen} (instantiated with Reed-Solomon codes) as a building block to construct our main protocol Π_{SPD} , which achieves *logarithmic* public communication while maintaining optimal private communication (and other desirable properties).

3.1 A generic protocol with optimal private communication

Protocol Π_{Gen} achieves essentially optimal communication complexity on the private wires of $O(\frac{mn}{n-t})$, where m is the length of the message, while maintaining linear communication complexity on the public channel. (See Section 4 for a precise statement of the lower bound.) This is the first SMT-PD protocol to achieve sublinear transmission rate on the private wires, and as such provides an affirmative answer to the question posed in [SJST09] of whether $O(n)$ private-wire transmission rate can be improved.

Π_{Gen} relies on two primitives as black boxes: an error-correcting code \mathcal{E} and an average-case strong extractor, Ext_A . The efficiency of the protocol depends on the interaction between the basic parameters of the protocol— ϵ , δ , m , n , and t —and the parameters of \mathcal{E} and Ext_A . After presenting the protocol and proving its security, we will examine its complexity in terms of these parameters.

At a high level, the protocol has the same structure as previous 3-round SMT-PD protocols: (1) in the first round, one of the parties (in our case \mathcal{R}) sends lots of randomness on each private wire; (2) using the public channel, \mathcal{R} then sends checks to verify the randomness sent in (1) was not tampered with; (3) \mathcal{S} discards any tampered wires, combines each remaining wire’s randomness to get a one-time pad R , and sends $C = M \oplus R$ on the public channel. However, our use of extractors allows us to reduce the amount of transmitted randomness, which is reflected in the gain in private communication.

We remark that one may modify Π_{Gen} to have interaction order $\mathcal{S}\text{-}\mathcal{R}\text{-}\mathcal{S}$, instead of $\mathcal{R}\text{-}\mathcal{R}\text{-}\mathcal{S}$ as we present it. One advantage of $\mathcal{R}\text{-}\mathcal{R}\text{-}\mathcal{S}$ is that when instantiated with deterministic extractors (see below), it does not require any random coins for \mathcal{S} (in contrast to $\mathcal{S}\text{-}\mathcal{R}\text{-}\mathcal{S}$, where both parties use randomness crucially).

Now we turn to the details of protocol Π_{Gen} . Let error-correcting code \mathcal{E} have encoding and decoding functions $\text{Enc} : \{0, 1\}^K \rightarrow \{0, 1\}^N$ and $\text{Dec} : \{0, 1\}^N \rightarrow \{0, 1\}^K$, respectively, and relative minimum distance D . (We will specify K below.)

While $N > K$ may be arbitrarily large for the purpose of correctness, we will want K/N and D both to be constant for our complexity analysis—that is, we want \mathcal{E} to be asymptotically good.

Second, let Ext_A be an average-case $(nK, m, k_{\min}, \epsilon/2)$ -strong extractor. Here K is, as above, the source length of the error-correcting code \mathcal{E} , and m and ϵ are the message-length and privacy parameters of Π_{Gen} . k_{\min} is the min-entropy threshold. Now clearly $m \leq k_{\min} \leq nK$. On the other hand, we require $k_{\min} = O(m)$ for our complexity claim to hold—that is, Ext_A should extract a constant fraction of the min-entropy. Further, the extractor’s seed length s should be $O(n + m)$.

Finally, let $b = \frac{1}{1-D}$, and then set $\ell = \lceil \log_b(t/\delta) \rceil$. Now with foresight, we set $K = \lceil k_{\min}/(n-t) \rceil + \ell$.⁷ Note that if $k_{\min} = O(m)$, then $K = O(m)/(n-t) + \ell$. The protocol, Π_{Gen} , is presented in Fig. 1.

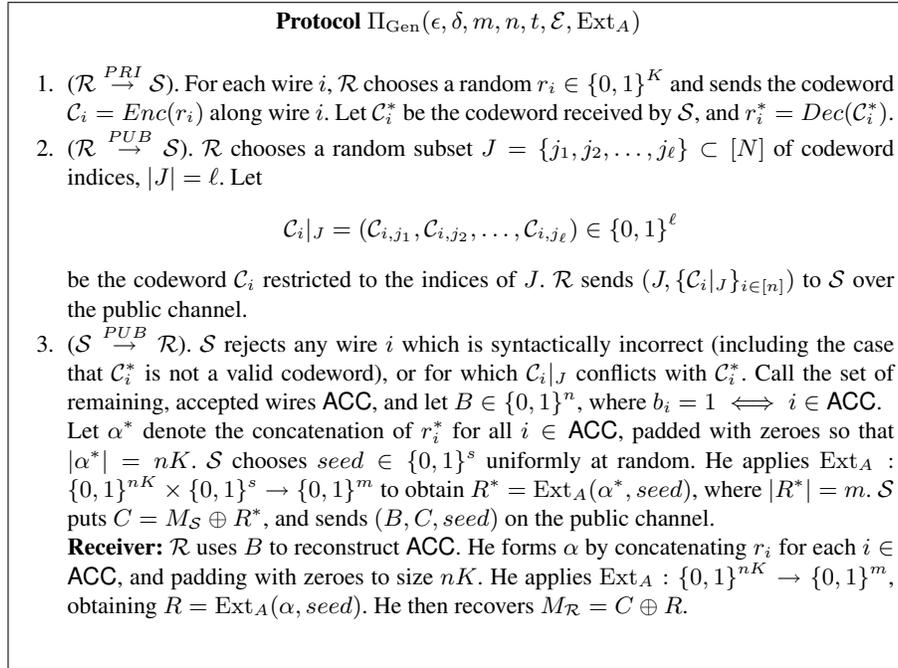


Fig. 1. A generic SMT-PD protocol with optimal communication complexity on the private wires and linear communication complexity on the public channel.

Theorem 5. Let $t < n$. Protocol Π_{Gen} is a $(3, 2)$ -round (ϵ, δ) -SMT-PD protocol with communication complexity $O(\frac{mn}{n-t})$ on the private wires provided that $m/(n-t) =$

⁷ As a sanity check, observe that $k_{\min} \leq nK = n(k_{\min}/(n-t) + \ell)$, so the extractor we define can exist.

$\Omega(\log(t/\delta))$, and communication complexity $\max(O(\log(t/\delta)(n + \log m)), O(m+n))$ on the public channel, provided only that $m = \Omega(\log(t/\delta))$.

Proof. Privacy. We first claim that if we omit C , then \mathcal{A} has essentially no information (up to ϵ) on \mathcal{S} 's output of the average-case extractor, $R^* = \text{Ext}_{\mathcal{A}}(\alpha^*, \text{seed})$. Formally:

Claim. The distribution $(U_s, R^*, \text{View}_{\mathcal{A}} \setminus C)$ is $\epsilon/2$ -close to $(U_s, U_m, \text{View}_{\mathcal{A}} \setminus C)$.

The remainder of the proof of ϵ -privacy is by contradiction: We show that, if there exists an adversary \mathcal{A} and messages M_0, M_1 such that $\Delta(\text{View}_{\mathcal{A}}(M_0), \text{View}_{\mathcal{A}}(M_1)) > \epsilon$, then there exists a distinguisher \mathcal{D} which can distinguish $(U_s, R^*, \text{View}_{\mathcal{A}} \setminus C)$ from $(U_s, U_m, \text{View}_{\mathcal{A}} \setminus C)$, in contradiction to the above claim.

So suppose such an \mathcal{A} , M_0, M_1 exist. Then there exists a distinguisher \mathcal{D}_0 which satisfies

$$|\Pr[D_0(\text{View}_{\mathcal{A}}(M_0)) = 1] - \Pr[D_0(\text{View}_{\mathcal{A}}(M_1)) = 1]| > \epsilon$$

In particular it follows that either

$$(1) \quad |\Pr[D_0(\text{View}_{\mathcal{A}}(M_0)) = 1] - \Pr[D_0(\text{View}_{\mathcal{A}}(M_{\S})) = 1]| > \epsilon/2$$

or

$$(2) \quad |\Pr[D_0(\text{View}_{\mathcal{A}}(M_{\S})) = 1] - \Pr[D_0(\text{View}_{\mathcal{A}}(M_1)) = 1]| > \epsilon/2.$$

Here $\text{View}_{\mathcal{A}}(M_{\S})$ denotes the random variable obtained by first sampling M_{\S} uniformly from $\{0, 1\}^m$, and then sampling from $\text{View}_{\mathcal{A}}$ conditioned on $M_{\mathcal{S}} = M_{\S}$. (If the probability distribution on \mathcal{M} is uniform, then the distribution of $\text{View}_{\mathcal{A}}(M_{\S})$ is identically that of $\text{View}_{\mathcal{A}}$, but we do not assume this here.)

Without loss of generality, we assume case (1) above holds. Now we describe \mathcal{D} , which uses \mathcal{D}_0 as a black box in order to distinguish $(U_s, R^*, \text{View}_{\mathcal{A}} \setminus C)$ and $(U_s, U_m, \text{View}_{\mathcal{A}} \setminus C)$. First, the challenger flips a coin. On heads, he samples $u \leftarrow (U_s, R^*, \text{View}_{\mathcal{A}} \setminus C)$, and on tails, $u \leftarrow (U_s, U_m, \text{View}_{\mathcal{A}} \setminus C)$. In either case he obtains $u = (u_s, u_{test}, u_{view})$ which he passes on to \mathcal{D} . \mathcal{D} forms $C_{\mathcal{D}} = M_0 \oplus u_{test}$, which plays the role of C in the protocol. He passes $u_{view} \cup C_{\mathcal{D}}$ to \mathcal{D}_0 , which returns a bit b representing its guess that $u_{view} \cup C_{\mathcal{D}}$ was sampled from $\text{View}_{\mathcal{A}}(M_b)$. If $b = 0$, then \mathcal{D} outputs a guess of “heads” (i.e., guesses u_{test} was sampled from R^*), otherwise \mathcal{D} guesses “tails” (u_{test} was sampled from U_m).

Now consider the success probability of \mathcal{D} when the challenger flips heads, so that $u_{test} \sim R^*$. In this case, $C_{\mathcal{D}} = M_0 \oplus R^*$ is obtained exactly as in Π_{Gen} , and therefore $u_{view} \cup C_{\mathcal{D}}$ is distributed identically with $\text{View}_{\mathcal{A}}(M_0)$. Thus $\Pr[\mathcal{D}(u) = 1 \mid \text{heads}] = \Pr[D_0(\text{View}_{\mathcal{A}}(M_0)) = 1]$. Alternatively, suppose the challenger flips tails, and u_{test} is uniform. Then $C_{\mathcal{D}} = M_0 \oplus u_{test}$ is uniform, which is also the distribution of C if we choose $M = M_{\mathcal{S}}$ uniformly at random. Thus $\Pr[\mathcal{D}(u) = 1 \mid \text{tails}] = \Pr[D_0(\text{View}_{\mathcal{A}}(M_{\S})) = 1]$. Putting these together, we discover

$$\begin{aligned} & |\Pr[\mathcal{D}(U_s, R^*, \text{View}_{\mathcal{A}} \setminus C) = 1] - \Pr[\mathcal{D}(U_s, U_m, \text{View}_{\mathcal{A}} \setminus C) = 1]| \\ &= |\Pr[D_0(\text{View}_{\mathcal{A}}(M_0)) = 1] - \Pr[D_0(\text{View}_{\mathcal{A}}(M_{\S})) = 1]| \\ &> \epsilon/2, \end{aligned}$$

which contradicts the above claim. This completes the verification of ϵ -privacy.

Reliability. Observe that $M_{\mathcal{R}} = C \oplus R$ and $M_S = C \oplus R^*$. Therefore,

$$\begin{aligned} \mathcal{R} \text{ fails to decode correctly } (M_{\mathcal{R}} \neq M_S) &\iff \text{Ext}(\alpha, \text{seed}) = R \neq R^* = \text{Ext}(\alpha^*, \text{seed}) \\ &\implies \alpha \neq \alpha^* \\ &\implies \exists i \in \text{ACC} \text{ s.t. } r_i \neq r_i^* \\ &\implies \exists i \in \text{ACC} \text{ s.t. } C_i \neq C_i^*. \end{aligned}$$

The latter event only happens if \mathcal{A} succeeds in altering C_i without S detecting it. By construction, our consistency check (Section 2) guarantees that this happens with probability at most $(1 - D)^\ell = \delta/t$ for a single wire, hence (taking a union bound over corrupt wires) probability at most δ overall. Consequently, $\Pr[M_{\mathcal{R}} = M_S] \geq 1 - \delta$.

Complexity. The private wires are used only in round 1, to send $\text{Enc}(r_i)$ on each wire. The total complexity is therefore $nN = O(nK)$ (for \mathcal{E} of constant rate). As noted above, our assumptions on \mathcal{E} and Ext_A imply that $K = O(m/(n-t) + \ell)$, and therefore the total private wire complexity is $O(mn/(n-t) + n\ell)$, which is $O(mn/(n-t))$ provided $m/(n-t) = \Omega(\ell)$.

The public channel is used in Rounds 2 and 3. In Round 2, \mathcal{R} transmits $J \subset [N]$ of size ℓ , and the restricted codewords $C_i|_J$, at total cost $\ell n + \ell \log N = \ell n + \ell(\log K + O(1)) = \ell n + O(\ell(\log(m/(n-t) + \ell)))$. Provided that $m = \Omega(\ell)$, this is $O(\ell(n + \log m))$.

In Round 3, S uses the public channel to send (B, C, seed) where B indicates accepted wires, C hides the message M_S , and seed is a seed for Ext_A . Thus the Round 3 public communication is $n + m + s$, which is $O(n + m)$ for any extractor with reasonable seed length. \square

3.2 Instantiating the generic protocol

Here we consider possible instantiations of Π_{Gen} . Since our main interest is in 0-private protocols, the most important instantiation will be that with Reed-Solomon codes and the extractor Ext_q of Section 2. Nevertheless, other choices of (explicit) extractor, such as Kamp and Zuckerman's deterministic symbol-fixing extractor [KZ06], are possible; refer to [GGO09] for more details.

Statistical error is a feature of all general-purpose randomness extractors. To get around it, we can exploit the fact that the sources arising from Π_{Gen} are not general min-entropy sources. Rather, conditioning on the adversary's view, each good wire carries independent, uniform randomness, and the corrupt wires carry fixed values. Thus the source we are interested in actually carries quite a great deal of structure. In particular, we may view it as a symbol-fixing source as described in Section 2, since we may group bits into symbols, and the adversary has no information on the symbols carried by good wires.

Consider an instantiation of Π_{Gen} using the extractor $\text{Ext}_q : \mathbb{F}_q^{kN} \rightarrow \mathbb{F}_q^r$ of Section 2, which is indeed errorless. (Here $r = \lceil m/\log q \rceil$ is the size of M_S in field elements.) Ext_q is, according to our notation, a $(kN, r, r, 0)$ extractor for sources over

\mathbb{F}_q : It extracts 100% of the randomness from its input with no statistical error. (It is also deterministic, hence trivially strong.) Since Ext_q operates at the level of field elements, Reed-Solomon codes are a natural choice for the error-correcting code \mathcal{E} of Π_{Gen} . We choose \mathcal{E} to be $\text{Ext}_q : \mathbb{F}_q^K \rightarrow \mathbb{F}_q^{2K}$, with relative minimum distance $1/2$.

We now describe two requirements imposed by this instantiation. First, the description of Π_{Gen} assumes an extractor which operates on bits rather than field elements. This presents no real problem, as all statements can be recast in a straightforward way to this new setting. However, as mentioned above, the move from $\{0, 1\}$ to \mathbb{F}_q does have the effect of adding a $\log q$ term to the message size required for optimal communication complexity (see statement of and complexity analysis for Theorem 6).

Second, we must specify the appropriate field size q in terms of the basic parameters m, n, t, δ . Recall $\ell = \lceil \log(t/\delta) \rceil$. We require (with foresight):

$$q \log q = \Omega(mn/(n-t)) \quad \text{and} \quad (q-2\ell) \log q > \frac{2m}{n-t}.$$

Thus $M_S \in \mathbb{F}_q^r$, where $r = \lceil m/\log q \rceil$.

For the proof of privacy, we require $\text{Ext}_q : \mathbb{F}_q^{nK} \rightarrow \mathbb{F}_q^r$ is in fact a perfect randomness extractor—so we need $q \geq nK + r$. Since $K = r/(n-t) + \ell$, we have (using $m = \Omega(n\ell)$):

$$\begin{aligned} nK + r &= n \cdot \left(\frac{r}{n-t} + \ell \right) + r = r \left(\frac{n}{n-t} + 1 \right) + n\ell \\ &= \frac{m}{\log q} \cdot \frac{n}{n-t} + O(m) = O\left(\frac{m}{\log q} \cdot \frac{n}{n-t} \right). \end{aligned}$$

Thus, for $q \geq nK + r$ it suffices that $q \log q = \Omega(mn/(n-t))$, which is our first assumption on q .

Now observe that in order for our codeword authentication to be valid, we need $q \geq 2K = 2r/(n-t) + 2\ell$. Thus we require:

$$\begin{aligned} q \geq 2r/(n-t) + 2\ell &\iff q \geq \frac{2m}{(\log q)(n-t)} + 2\ell \\ &\iff q \log q \geq \frac{2m}{n-t} + 2\ell \log q \\ &\iff (q-2\ell) \log q \geq \frac{2m}{n-t}, \end{aligned}$$

which gives our second condition on q .

3.3 A protocol with logarithmic public communication

In this section we present a protocol for SMT-PD which is the first to achieve logarithmic communication complexity (in m) on the public channel. The protocol is perfectly private, achieves the optimal communication complexity of $O(\frac{mn}{n-t})$ on the private wires, and has optimal round complexity of (3, 2).

In its Round 3 communication, Π_{Gen} incurs a cost of size m on the public channel, which we wish to reduce to $O(\log m)$. Our improvement comes from the insight that S

can send the third-round message (C , in the notation of Π_{Gen}) on the *common* wires, provided that \mathcal{S} *authenticates* the transmission (making use of the public channel).

\mathcal{S} could simply send C on every common wire and authenticate C publicly. The downside of this approach is that the private wire complexity would be $\Omega(mn)$ rather than $O(\frac{mn}{n-t})$ —no longer optimal. Our solution is to take C and encode it *once again* using Reed-Solomon into shares C_1, \dots, C_n , each of size $\approx \frac{m}{n-t}$, such that any $n-t$ correct C_i 's will reconstruct C . \mathcal{S} then sends C_i on wire i , and authenticates each C_i publicly.

This authentication uses a short secret key, s^* , of size $\ell(n + \log(\frac{cm}{n-t}))$ (which is the cost of authenticating n messages of size $cm/(n-t)$, using the consistency check of Section 2; c is an absolute constant defined below). Thus, \mathcal{S} and \mathcal{R} will run two processes in parallel: a “small” strand, in which \mathcal{S} privately sends the short key to \mathcal{R} ; and a “big” strand, in which \mathcal{S} sends $M_{\mathcal{S}}$ to \mathcal{R} , making use of the shared key in the third round. The small protocol sends the short key using any reasonably efficient SMT-PD protocol; for ease of exposition, we use Π_{Gen} , instantiated with Reed-Solomon codes. We also use Π_{Gen} with Reed-Solomon codes for the big strand of the protocol in order to achieve perfect privacy and optimal private wire complexity.

We now describe the protocol in detail. Many of the parameters are the same as in (the Reed-Solomon instantiation of) Π_{Gen} : We set $\ell = \lceil \log(t/\delta) \rceil$, and fix a prime q such that

$$q \log q = \Omega(mn/(n-t)) \quad \text{and} \quad (q-2\ell) \log q \geq \frac{2m}{n-t}.$$

The message space is $\mathcal{M} = \mathbb{F}_q^r$, that is, an m -bit message is considered as a sequence of $r = \lceil m/\log q \rceil$ field elements in \mathbb{F}_q . (However, we also assume, for the purpose of the Round 3 authentication, that the field elements are actually *represented* as bit-strings of length $r \log q$.) Set $K = \lceil r/(n-t) \rceil + \ell$ and $N = 2K$.

In addition to the above parameters, we will also define their small-strand counterparts, which we notate using variables with hats. Set $\hat{m} = \ell(n + \log(cK \log q))$ —as noted above, this is the size of the shared secret which will be used to authenticate the C_i 's. Here the constant $c > 1$ is the expansion factor of an efficiently computable, constant-rate error-correcting code \mathcal{E}' of relative minimum distance (say) $1/3$. (We caution that \mathcal{E}' plays a different role in Π_{SPD} than \mathcal{E} did in Π_{Gen} , hence the different name.) We will use Enc and Dec to denote the encoding and decoding functions of \mathcal{E}' ; we use Enc_{RS} and Dec_{RS} for the encoding and decoding functions of the Reed-Solomon code which functions as \mathcal{E} for Π_{SPD} .

Fix \hat{q} to be a prime such that

$$\hat{q} \log \hat{q} = \Omega(\frac{\hat{m}n}{n-t}) \quad \text{and} \quad (\hat{q}-2\ell) \log \hat{q} > \frac{2\hat{m}}{n-t},$$

Set $\hat{r} = \lceil \hat{m}/\log \hat{q} \rceil$, $\hat{K} = \lceil \hat{r}/(n-t) \rceil + \ell$, and $\hat{N} = 2\hat{K}$. Finally, set $\ell_{3/2} = \log_{3/2}(t/\delta)$.

The protocol, Π_{SPD} (for “small public discussion”), is shown in Figure 2. Keep in mind the high-level understanding of the protocol: The first two rounds are simply parallel versions of Rounds 1 and 2 of Π_{Gen} , run with different (big and small) parameters. In Round 3, we complete the small instance of Π_{Gen} as usual, and use the resulting

Protocol Π_{SPD}

1. ($\mathcal{R} \xrightarrow{PRI} \mathcal{S}$). **(small)** For each wire i , \mathcal{R} chooses a random $\hat{f}_i \in \mathbb{F}_{\hat{q}}[X]$ such that $\deg(\hat{f}_i) \leq \hat{K}$. \mathcal{R} sends the Reed-Solomon (RS) codeword $\hat{C}_i = (\hat{f}_i(1), \hat{f}_i(2), \dots, \hat{f}_i(\hat{N}))$ along wire i . Let \hat{C}_i^* be the codeword received by \mathcal{S} , and $\hat{f}_i^* = \text{Dec}_{RS}(\hat{C}_i^*)$.
(big) For each wire i , \mathcal{R} chooses a random $f_i \in \mathbb{F}_q[X]$ such that $\deg(f_i) \leq K$. \mathcal{R} sends the RS codeword $C_i = (f_i(1), f_i(2), \dots, f_i(N))$ along wire i . Let C_i^* be the codeword received by \mathcal{S} , and $f_i^* = \text{Dec}_{RS}(C_i^*)$.
2. ($\mathcal{S} \xrightarrow{PUB} \mathcal{R}$). **(small)** \mathcal{R} chooses a random subset $\hat{J} = \{\hat{j}_1, \dots, \hat{j}_\ell\} \subset [\hat{N}]$ of codeword indices, $|\hat{J}| = \ell$. \mathcal{R} performs codeword verification as in Section 2 by sending \hat{J} , as well as $\{\hat{C}_i|_{\hat{j}}\}$ for each wire i , over the public channel.
(big) \mathcal{R} chooses a random subset $J = \{j_1, \dots, j_\ell\} \subset [N]$ of codeword indices, $|J| = \ell$. \mathcal{R} performs codeword verification as in Section 2 by sending J , as well as $\{C_i|_J\}$ for each wire i , over the public channel.
3. ($\mathcal{S} \xrightarrow{PUB+PRI} \mathcal{R}$). \mathcal{S} rejects any wire i which is syntactically incorrect or which fails one of the consistency checks in Round 2. Call the set of remaining, accepted wires **ACC**.
(small) Let $\hat{\alpha}^*$ denote the concatenation of \hat{f}_i^* for each $i \in \text{ACC}$, padded with $0 \in \mathbb{F}_{\hat{q}}$ so its length is $\hat{K}n$. Applying $\text{Ext}_{\hat{q}} : \mathbb{F}_{\hat{q}}^{\hat{K}n} \rightarrow \mathbb{F}_{\hat{q}}^{\hat{r}}$ of Section 2, \mathcal{S} obtains $s^* = \text{Ext}_{\hat{q}}(\hat{\alpha}^*)$.
(big) Let α^* denote the concatenation of f_i^* for each $i \in \text{ACC}$, padded with $0 \in \mathbb{F}_q$ so its length is Kn . Applying the randomness extractor $\text{Ext}_q : \mathbb{F}_q^{Kn} \rightarrow \mathbb{F}_q^r$, \mathcal{S} obtains $R^* = \text{Ext}_q(\alpha^*)$.
 Now M_S and R^* are both vectors in \mathbb{F}_q^r ; \mathcal{S} puts $C = R^* + M_S$. Now \mathcal{S} applies the Reed Solomon code $\mathbb{F}_q^r \rightarrow \mathbb{F}_q^{Kn}$ to C , obtaining a codeword $D \in \mathbb{F}_q^{Kn}$. Let $D = (D_1, \dots, D_n)$ where each $D_i \in \mathbb{F}_q^K$. View D_i as a bit-string of length $K \log q$, and let $E_i = \text{Enc}(D_i)$, so that $|E_i| = cK \log q$ (in bits). \mathcal{S} sends E_i on wire $i \in \text{ACC}$; let E_i^* denote the message received by \mathcal{R} on wire i .
 To authenticate each E_i , \mathcal{S} chooses a random subset $J' \subseteq [cK \log q]$, $|J'| = \ell_{3/2}$. Put $\text{auth}_S = (J', \{E_i|_{J'}\}_{i \in \text{ACC}})$; we have $|\text{auth}_S| \leq \hat{m}$ (with equality if every wire is in **ACC**). Padding as necessary, view auth_S as an element of $\mathbb{F}_{\hat{q}}^{\hat{r}}$. \mathcal{S} sets $V = s^* + \text{auth}_S$ and sends (V, B) over the public channel, where B is an n -bit string representing the set **ACC**.
Receiver: \mathcal{R} learns **ACC** from B . For $i \in \text{ACC}$, he forms α , the concatenation of f_i for each $i \in \text{ACC}$ (padded with $0 \in \mathbb{F}_q$ to length Kn). He applies Ext_q to obtain $R = \text{Ext}_q(\alpha) \in \mathbb{F}_q^r$.
 Similarly, for $i \in \text{ACC}$, he forms $\hat{\alpha}$, the concatenation of \hat{f}_i for each $i \in \text{ACC}$ (padded with $0 \in \mathbb{F}_{\hat{q}}$ to length $\hat{K}n$). He applies $\text{Ext}_{\hat{q}}$ to obtain $s = \text{Ext}_{\hat{q}}(\hat{\alpha}) \in \mathbb{F}_{\hat{q}}^{\hat{r}}$.
 Next \mathcal{R} forms $V - s$, which he parses as $\text{auth}_{\mathcal{R}} = (J'^*, \{\text{check}_i\}_{i \in \text{ACC}})$. For each (correctly formed) E_i^* , \mathcal{R} verifies its authenticity by checking that $E_i^*|_{J'^*} = \text{check}_i$. For those which pass, he recovers $D_i^* = \text{Dec}(E_i^*)$, $D_i^* \in \mathbb{F}_q^K$. Once \mathcal{R} has recovered at least $n - t$ valid D_i^* 's, he has $K(n - t) = r$ symbols in \mathbb{F}_q , which he uses to decode the RS code used by \mathcal{S} to encode C . (This is simply interpolation.) Call the result $C^* \in \mathbb{F}_q^r$. Finally, \mathcal{R} obtains $M_{\mathcal{R}} = C^* - R$.
 (On failure to authenticate at least $n - t$ E_i^* 's, or to parse $\text{auth}_{\mathcal{R}}$ correctly, \mathcal{R} outputs \perp .)

Fig. 2. SMT-PD protocol with small (logarithmic) public communication and optimal private communication.

shared secret to blind the (public-channel) authentication of the C_i 's which encode C . The latter have been sent on the unreliable private wires, unlike in Π_{Gen} , where no authentication was required in Round 3 since C itself was sent on the public channel.

Theorem 6. *Protocol Π_{SPD} (Fig. 2) is a valid $(3, 2)$ -round $(0, 3\delta)$ -SMT-PD protocol. It has communication complexity $O(\frac{mn}{n-t})$ on the private wires and $O(n \log(t/\delta) \log m)$ on the public channel, provided $m = \Omega(n \log(t/\delta) \log q)$.*

4 Private Communication Lower Bound

In this section we prove a lower bound of $\Omega(\frac{nm}{n-t})$ for the expected communication complexity on the private wires, for any (ϵ, δ) -SMT-PD protocol (where ϵ and δ are considered constants). Since protocol Π_{Gen} of the previous section meets this bound, we provide a complete answer to the question raised in [SJST09] of determining the optimal transmission rate on private wires for an (ϵ, δ) -SMT-PD protocol.

Our communication lower bound holds even for a weakened adversary who is *passive* and *non-adaptive*—that is, \mathcal{A} chooses which wires to corrupt at the start of the protocol and only eavesdrops thereafter. It also holds even if we modify δ -reliability so that the probability that $M_{\mathcal{R}} = M_{\mathcal{S}}$ is taken over the choice of $M_{\mathcal{S}}$ as well (and not just the players' coins). Further, as noted in the Introduction, it also holds in the case of SMT with no public channel, *mutatis mutandis*.

For the lower bound, we assume that $M_{\mathcal{S}}$ is chosen uniformly at random from \mathcal{M} ; in this case $H(M_{\mathcal{S}}) = \log |\mathcal{M}|$. In the following lemmas we assume Π is a valid (ϵ, δ) -SMT-PD protocol, and probabilities are over all players' coins as well as the random selection of $M_{\mathcal{S}} \in \mathcal{M}$.

The first two lemmas are complementary, establishing entropy versions of ϵ -privacy and δ -reliability, respectively. Namely, in Lemma 7, we show that in any ϵ -private protocol, the entropy of $M_{\mathcal{S}}$ remains high given the adversary's view. Then in Lemma 8, we show that for any δ -reliable protocol (with passive adversary), the entropy of $M_{\mathcal{S}}$ given the entire transcript of communications is low. Though these statements are quite intuitive, their proofs are relatively delicate.

Lemma 7. *For all adversaries \mathcal{A} and all ϵ -private protocols, $H(M_{\mathcal{S}} | \text{View}_{\mathcal{A}}) \geq -\log(1/|\mathcal{M}| + 2\epsilon)$.⁸*

The *transcript* T of an (ϵ, δ) -SMT-PD protocol execution is the random variable consisting of the list of messages the players send on public and private channels over the course of the protocol. Thus in the case of a passive adversary, T is completely determined by $M_{\mathcal{S}}$, $C_{\mathcal{S}}$, and $C_{\mathcal{R}}$. For a given set of wires S , we will let T_S denote the transcript restricted to communications on the wires in S . In the sequel we use PUB, PRIV, CORR, and SEC to denote respectively the public channel, private wires, corrupted wires, and secure (uncorrupted and private) wires.

We use $H_2(\cdot)$ to denote the binary entropy function, $H_2(p) = -p \log p - (1 - p) \log(1 - p)$.

⁸ This entropy lemma is not directly equivalent to a seemingly related probability version (as in [SJST09], Lemma 2).

Lemma 8. For all δ -reliable protocols, $H(M_S | T) \leq H_2(\sqrt{\delta}) + 2\sqrt{\delta}H(M_S)$.

Given Lemmas 7 (a proof of “high” entropy) and 8 (a proof of “low” entropy), we take the difference of the two inequalities (leaving still a “high” amount of entropy), and show that this bounds from below $H(T_{SEC} | SEC)$. This is intuitive: the adversary knows which wires are secure, and yet it is only from these wires that \mathcal{S} and \mathcal{R} can leverage any privacy at all. Therefore the entropy of the messages on them should be high.

Lemma 9. $-\log(1/|\mathcal{M}| + 2\epsilon) - H_2(\sqrt{\delta}) - 2\sqrt{\delta} \log |\mathcal{M}| \leq H(T_{SEC} | SEC)$.

Our main lower bound theorem follows. The idea is straightforward. Since the set of secure wires is unknown to \mathcal{S} and \mathcal{R} (for a passive adversary, say), it must be that, in an average sense, every set of $n - t$ private wires carries the requisite entropy. Then we use Han’s inequality (see proof in [GGO09]) to “average” the entropy over all subsets of $n - t$ wires and obtain an estimate for the total entropy on private wires, completing the proof.

Theorem 10. Let Π be any (ϵ, δ) -SMT-PD protocol with $n \leq 2t$, in the presence of a passive, non-adaptive adversary \mathcal{A} . Let C denote the expected communication (in bits) over the private wires (the expectation is taken over all players’ coins and the choice of $M_S \in \mathcal{M}$). Then

$$C \geq \frac{n}{n-t} \cdot (-\log(1/|\mathcal{M}| + 2\epsilon) - H_2(\sqrt{\delta}) - 2\sqrt{\delta} \log |\mathcal{M}|)$$

In particular, if $\epsilon = O(1/|\mathcal{M}|)$ and $\delta = O(1)$, then $C = \Omega(mn/(n-t))$.

Corollary 11. Provided that $\epsilon = O(1/|\mathcal{M}|)$, and $\delta = O(1)$, protocols Π_{Gen} and Π_{SPD} have optimal private communication complexity $O(\frac{nm}{n-t})$ for messages of size $m = \Omega(n\ell)$ and $m = \Omega(n\ell \log q)$, respectively.

5 Amortized Use of the Public Channel

A natural question when considering SMT-PD as a subroutine in a larger protocol is whether some of the lower bounds on resource use for a single execution of SMT-PD can be beaten *on average* through amortization. For instance, an almost-everywhere secure computation protocol may invoke an SMT-PD subroutine every time any two nodes in the underlying network need to communicate. Must they use the public channel twice every single time, or can the nodes involved, say, save some state information which allows them to reduce their use of the public channel in later invocations?

Our next result shows that amortization can in fact drastically reduce the use of the public channel: indeed, it is possible to limit the total number of uses of the public channel to *two*, no matter how many messages are ultimately sent between two nodes. (Since two uses of the public channel are required to send any reliable communication whatsoever, this is best possible.)

Of course, \mathcal{S} and \mathcal{R} may use the first execution of SMT-PD to establish a shared secret key, which can be used for message encryption and authentication on the common

wires. The Sender computes a ciphertext and sends it (with authentication) on every common wire. With overwhelming probability, no forged message is accepted as authentic, and the Receiver accepts the unique, authentic message which arrives on any good wire. However, since we are considering the information-theoretic setting, each use of the shared key reduces its entropy with respect to the adversary’s view. If the parties know in advance an upper bound on the total communication they will require, and can afford to send a proportionally large shared key in the first execution of SMT-PD, then this approach is tenable by itself.

In some situations, however, the players may not know a strict upper bound on the number of messages they will send. And even when they do, it may happen that the protocol terminates early with some probability, so that an initial message with large entropy is mostly wasted. With these considerations in mind, we now explore strategies which allow \mathcal{S} and \mathcal{R} to communicate *indefinitely* after using only two broadcast rounds and a limited initial message. Our approach is to separate Sender and Receiver’s interaction following the first execution of SMT-PD into two modes: a *Normal Mode* and a *Fault-Recovery Mode*.

In the Normal Mode, \mathcal{S} and \mathcal{R} communicate over the common wires without making use of their shared key; they are successful provided the adversary does not actively interfere. If the adversary does interfere, one of the players (say \mathcal{R}) will detect this and enter Fault-Recovery Mode, in which he uses the shared key to broadcast information about the messages he received on each common wire, allowing \mathcal{S} to determine at least one corrupted wire (which he then informs \mathcal{R} about, authentically).

In this way, \mathcal{S} and \mathcal{R} communicate reliably and privately so long as the adversary is passive; and any time he is active, they are able to eliminate at least one corrupted wire.⁹ (Of course, once they have eliminated all t corrupt wires, communication becomes *very* efficient.) In the sequel, we describe implementations of Normal Mode and Fault-Recovery Mode, as well as how the two modes interact with each other.

Normal Mode. Let us first define a weaker version of SMT by public discussion in which reliability is only guaranteed for a passive adversary. Let Π be a protocol which attempts to send a message from \mathcal{S} to \mathcal{R} using *only the common wires* (and not relying on any shared secret key). Then we say Π is a *Weak* (ϵ, δ) *SMT-PD* protocol if it satisfies Definition 2 where we (1) add to the adversary’s view a bit indicating whether \mathcal{R} accepted a message or not (see next point), and (2) replace RELIABILITY with:

WEAK RELIABILITY:

- (*Correctness with passive adversary*) If the adversary only eavesdrops, then \mathcal{R} receives the message correctly.
- (*Detection of active adversary*) If the adversary actively corrupts any wire, then with probability $\geq 1 - \delta$, either \mathcal{R} receives the message correctly ($M_{\mathcal{R}} = M_{\mathcal{S}}$), or \mathcal{R} outputs “Corruption detected.”

The first change above affects ϵ -privacy since it alters the definition of $\text{View}_{\mathcal{A}}$; this is necessary because in the compiled, amortized protocol using Weak SMT-PD as a subroutine, the adversary will learn whether \mathcal{R} accepted a message based on whether \mathcal{R} does or does not enter Fault-Recovery Mode.

⁹ This is akin to the “slow” PSMT original protocol in [DDWY93].

We remark in passing that Weak SMT-PD is similar in spirit to *almost* SMT from the standard (non-public discussion) model [KS07], in that both are relaxations which allow one-round transmission (for Weak SMT-PD, only with a passive adversary). The difference is that in the ordinary model, definitions for almost SMT require that the message be correctly received with overwhelming probability regardless of the adversary’s actions; in the public discussion model, when the adversary controls a majority of wires, this is impossible, so we only require that corruptions be detected. Indeed, we cannot guarantee reliability in a single round even when the adversary simply *blocks* transmission on corrupted wires (otherwise a minority of wires would carry enough information to recover the message, thus violating privacy).

If we do not require the Weak SMT-PD protocol to finish in *one round*, then there is a simple solution: use the common wires to *simulate* the public channel wire in an ordinary SMT-PD protocol. Any time a party would use the public channel, they instead send the public-channel message over *every* common wire. Two possibilities arise: (1) The adversary never tampers with any such “virtual” public channel invocation. In this case, the virtual public channel functions like an actual public channel, and the protocol succeeds with the same probability as the underlying SMT-PD protocol. (2) The adversary at some point tampers with a virtual public channel invocation. If he does, then the receiving party in that round will detect tampering, and can notify the other player by sending a flag on every channel (or, if the receiving player is \mathcal{R} and it is the final round, he just outputs “Corruption Detected”).¹⁰

The above Weak SMT-PD protocol is conceptually simple (given a pre-existing SMT-PD protocol!), but we might hope to do Weak SMT-PD in a single round, as opposed to the three rounds required for ordinary SMT-PD. The following simple scheme shows one way this can be done.

Assume the Sender wants to send a single field element $M_S = \alpha \in \mathbb{F}_q$. The one-round protocol, $\Pi_{W-SMT-PD}$, is shown in Figure 3. Essentially, the sender performs a $3t + 2$ -out-of- $3n$ Shamir secret sharing of the message; however, rather than sending externally specified shares on each wire i (such as $f(1), f(2), f(3)$ on wire 1), he chooses a set of *random* points on which to evaluate f .

Lemma 12. *The protocol of Figure 3 is a Weak (δ, δ) -SMT-PD protocol for q sufficiently large $(\Omega(t/\delta))$.*

We are now ready to describe Normal Mode for \mathcal{S} and \mathcal{R} : it is simply the repeated execution of the Weak SMT-PD protocol, with the two players alternating the role of Sender and Receiver, until one of them as Receiver outputs “Corruption detected.” At that time, that player’s next message to the other party will alert them to enter Fault-Recovery Mode.

Fault-Recovery Mode. Specifically, suppose \mathcal{R} detects corruption in a message sent by \mathcal{S} . He will then use the shared secret established in the initial execution of (ordinary) SMT-PD to secretly and authentically send the following on all wires: (1) a flag

¹⁰ We do not consider here whether such a protocol preserves (ϵ) -privacy when the adversary knows whether \mathcal{R} detects corruption; obviously this depends on the details of the protocol. Therefore this is not quite a black-box reduction.

Protocol $\Pi_{W-SMT-PD}$

1. ($\mathcal{S} \xrightarrow{PRI} \mathcal{R}$). \mathcal{S} chooses a random polynomial $f \in \mathbb{F}_q[x]$ with $\deg(f) \leq 3t + 1$ and $f(0) = \alpha$, and a random sequence $x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, \dots, x_{n1}, x_{n2}, x_{n3}$ of $3n$ distinct elements of $\mathbb{F}_q \setminus \{0\}$. On wire i \mathcal{S} sends to \mathcal{R} the three pairs $(x_{i1}, f(x_{i1})), (x_{i2}, f(x_{i2})), (x_{i3}, f(x_{i3}))$.
Receiver: On wire i , \mathcal{R} receives (x_{ij}^*, y_{ij}^*) for $j = 1, 2, 3$. He verifies that all $3n$ x_{ij}^* 's are distinct, and that the $3n$ points (x_{ij}^*, y_{ij}^*) lie on a polynomial f^* of degree $\leq 3t+1$. If so, he outputs $M_{\mathcal{R}} = f^*(0)$; otherwise (or in case some wire is syntactically incorrect) he outputs "Corruption detected."

Fig. 3. A one-round Weak SMT-PD protocol.

signalling Fault-Recovery Mode; (2) a list of specific wires known to be corrupted (if any); (3) the received transmission on all wires not known to be corrupt.

Since at least one of the wires is not corrupted, \mathcal{S} will receive this communication on it and (verifying its authenticity) enter Fault-Recovery Mode also. \mathcal{S} recovers the set of received transmissions and determines which ones were tampered with. He then sends the following to \mathcal{R} , again using the shared secret for privacy and authentication: (1) the message $M_{\mathcal{S}}$ on which \mathcal{R} detected corruption; (2) an updated list of specific wires known to be corrupted. At this time, \mathcal{R} has received the intended message and Normal Mode resumes with \mathcal{R} now playing the role of Sender.

Each time Fault-Recovery Mode occurs, \mathcal{S} and \mathcal{R} are able to detect at least one previously unknown corrupt wire. If at any point \mathcal{S} and \mathcal{R} have jointly detected t wires as corrupt, they will simply send all future transmissions on the remaining, good wires, guaranteeing perfect privacy and reliability.

Theorem 13. *Given an initial shared secret consisting of $O(n^2)$ field elements, \mathcal{S} and \mathcal{R} can communicate indefinitely using only the private wires. The probability that one of them will ever accept an incorrect message is $\leq t\delta$. Moreover, with probability $\geq 1 - t\delta$, \mathcal{A} gains at most δ information on each of t different messages, and no information on any other message.*

References

- [ACH06] S. Agarwal, R. Cramer, and R. de Haan. Asymptotically optimal two-round perfectly secure message transmission. In *Advances in Cryptology—CRYPTO'06*, 2006.
- [BG93] P. Berman and J. Garay. Fast consensus in networks of bounded degree. *Distributed Computing*, 2(7):62–73, 1993. Preliminary version in *WDAG'90*.
- [BGW88] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, pages 1–10, 1988.
- [CCD88] D. Chaum, C. Crepeau, and I. Damgard. Multiparty unconditionally secure protocols. In *STOC*, pages 11–19, 1988.

- [CPRS08] A. Choudhary, A. Patra, C. P. Rangan, and K. Srinathan. Unconditionally reliable and secure message transmission in undirected synchronous networks: Possibility, feasibility and optimality. Cryptology ePrint Archive, Report 2008/141, 2008.
- [DDWY93] D. Dolev, C. Dwork, O. Waarts, and M. Young. Perfectly secure message transmission. *Journal of ACM*, 1(40):17–47, 1993.
- [DORS08] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 2008.
- [DPPU86] C. Dwork, D. Peleg, N. Pippinger, and E. Upfal. Fault tolerance in networks of bounded degree. In *STOC*, pages 370–379, 1986.
- [FFGV07] M. Fitzi, M. Franklin, J. Garay, and S. Harsha Vardhan. Towards optimal and efficient perfectly secure message transmission. In *TCC*, 2007.
- [FM97] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM J. Comput.*, 26(4):873–933, 1997.
- [FW98] M. Franklin and R. Wright. Secure communications in minimal connectivity models. In *Advances in Cryptology—EUROCRYPT’98*, pages 346–360, 1998.
- [GGO09] J. Garay, C. Givens, and R. Ostrovsky. Secure message transmission with small public discussion. Cryptology ePrint Archive, Report 2009/519, 2009.
- [GM98] J. Garay and Y. Moses. Fully polynomial Byzantine agreement for $n > 3t$ processors in $t + 1$ rounds. *SIAM J. Comput.*, 27(1):247–290, 1998. Prelim. in *STOC ’92*.
- [GO08] J. Garay and R. Ostrovsky. Almost-everywhere secure computation. In *Advances in Cryptology—Eurocrypt’08*, pages 307–323, 2008.
- [GRR98] R. Gennaro, M. Rabin, and T. Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proc. 17th Annual ACM Symp. on Principles of Distributed Computing, PODC*, pages 101–111. ACM, 1998.
- [KK06] J. Katz and C. Koo. On expected constant-round protocols for Byzantine agreement. In *Advances in Cryptology – CRYPTO 2006*, pages 445–462, 2006.
- [KS07] K. Kurosawa and K. Suzuki. Almost secure (1-round, n-channel) message transmission scheme. Cryptology ePrint Archive, Report 2007/076, 2007.
- [KS08] K. Kurosawa and K. Suzuki. Truly efficient 2-round perfectly secure message transmission scheme. In *Advances in Cryptology – EUROCRYPT’08*, pages 324–340, 2008.
- [KZ06] J. Kamp and D. Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. *SIAM J. Comput.*, 36(5):1231–1247, 2006.
- [Lam87] L. Lamport. A fast mutual exclusion algorithm. *ACM Transactions on Computer Systems*, 5(1):1–11, 1987.
- [MS83] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes*. North Holland, January 1983.
- [SA96] H. Sayeed and H. Abu-Amara. Efficient perfectly secure message transmission in synchronous networks. *Information and Computation*, 1(126):53–61, 1996.
- [SJUST09] H. Shi, S. Jiang, R. Safavi-Naini, and M. Tuhin. Optimal secure message transmission by public discussion. In *IEEE Symposium on Information Theory*, 2009.
- [SNP04] K. Srinathan, A. Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In *Advances in Cryptology—CRYPTO’04*, pages 545–561, 2004.
- [SPR07] K. Srinathan, N. R. Prasad, and C. P. Rangan. On the optimal communication complexity of multiphase protocols for perfect communication. *IEEE Symposium on Security and Privacy*, 0:311–320, 2007.
- [Upf92] E. Upfal. Tolerating linear number of faults in networks of bounded degree. In *PODC*, pages 83–89, 1992.