On the Power of Correlated Randomness in Secure Computation

Yuval Ishai[1,⋆], Eyal Kushilevitz[1,⋆⋆], Sigurd Meldgaard[2,⋆⋆⋆], Claudio Orlandi[2,†], and Anat Paskin-Cherniavsky[1,‡]

[1] Technion, Haifa, Israel {yuvali,eyalk,anatp}@cs.technion.ac.il
[2] Aarhus University, Denmark {stm,orlandi}@cs.au.dk

Abstract We investigate the extent to which correlated secret randomness can help in secure computation with no honest majority. It is known that correlated randomness can be used to evaluate any circuit of size s with perfect security against semi-honest parties or statistical security against malicious parties, where the communication complexity grows linearly with s. This leaves open two natural questions: (1) Can the communication complexity be made independent of the circuit size? (2) Is it possible to obtain *perfect* security against malicious parties?
We settle the above questions, obtaining both positive and negative results on unconditionally secure computation with correlated randomness. Concretely, we obtain the following results.

Minimizing communication. Any multiparty functionality can be realized, with perfect security against semi-honest parties or statistical security against malicious parties, by a protocol in which the number of bits communicated by each party is linear in its input length. Our protocol uses an exponential number of correlated random bits. We give evidence that super-polynomial randomness complexity may be inherent.

Perfect security against malicious parties. Any finite "sender-receiver" functionality, which takes inputs from a sender and a receiver and delivers an output only to the receiver, can be *perfectly* realized given correlated randomness. In contrast, perfect security is generally impossible for functionalities which deliver outputs to both parties. We also show useful functionalities (such as string equality) for which there are *efficient* perfectly secure protocols in the correlated randomness model.

Perfect correctness in the plain model. We present a general approach for transforming perfectly secure protocols for sender-receiver functionalities in the correlated randomness model into secure protocols

⋆ Supported by the European Research Council as part of the ERC project CaC (grant 259426), ISF grant 1361/10, and BSF grant 2008411. Research done in part while visiting UCLA.
⋆⋆ Supported by ISF grant 1361/10 and BSF grant 2008411.
⋆⋆⋆ Supported by ERC grant 259426. Research mostly done while visiting the Technion.
† Supported by The Danish Council for Independent Research (DFF). Research done in part while visiting UCLA and Bar-Ilan University.
‡ Supported by ERC grant 259426.

in the plain model which offer *perfect correctness* against a malicious sender. This should be contrasted with the impossibility of perfectly sound zero-knowledge proofs.

1 Introduction

Secure computation is a fundamental problem that has been extensively studied since the 1980s, originating from the seminal works of [35,20,6,11]. In this paper, we study the power of *correlated randomness* in secure two-party computation and multiparty computation with no honest majority. That is, we consider secure computation with a randomness distribution phase which takes place before the inputs are known. In this phase the parties receive a sample from a predetermined joint distribution. While each party only receives its own random string from this sample, these random strings are correlated as specified by the joint distribution.

From a theoretical point of view, the correlated randomness model is interesting because it can be used to circumvent impossibility results for the plain model such as the impossibility of information-theoretic security, analogously to the use of shared secret randomness for encryption. This model can also be of practical relevance, as it can be instantiated in the following ways:

- MPC WITH PREPROCESSING. It is often the case that parties can use idle time before they have any input to run a secure "offline protocol" for generating and storing correlated randomness. This correlated randomness is later consumed by an "online protocol" which is executed once the inputs become available. This paradigm for MPC is particularly useful when it is important that the outputs are known shortly after the inputs are (i.e., for low-latency computation). Note that if the online protocol is unconditionally secure, then it has the potential efficiency advantage of not requiring any "cryptographic" operations. If the online protocol is *perfectly secure*, then it has the additional potential advantage of a *finite* complexity that does not grow with a statistical security parameter. From here on we will refer to MPC with correlated randomness also as *MPC with preprocessing*.
- COMMODITY-BASED MPC. In the setting of commodity-based cryptography [4], the parties can "purchase" correlated randomness from one or more external servers. Security in this model is guaranteed as long as at most t of the servers are corrupted, for some specified threshold t, where corrupted servers may potentially collude with the parties. In contrast to the obvious solutions of employing a server as a trusted party or running an MPC protocol among the servers, the servers are only used during an offline phase before the inputs are known, and do not need to be aware of the existence of each other.
- HONEST-MAJORITY MPC. Recent large-scale practical uses of MPC [10,9] employed three servers and assumed that at most one of these servers is corrupted by a semi-honest adversary. Protocols in the correlated randomness model can be translated into protocols in this 3-server model by simply letting one server generate the correlated randomness for the other two.

A prime example of a cryptographic task that can benefit from having access to correlated randomness is oblivious transfer (OT) [30,16]. Beaver [3] shows that having access to the inputs and outputs of a random instance of OT can be used to realize OT on any inputs with unconditional security. This, together with the fact that OT is complete for secure computation [24,22], shows that every functionality can be securely computed given access to an appropriate source of correlated randomness and no additional assumptions.

While the OT protocol from [3] has both *perfect security* and *optimal communication complexity*, the protocols obtained using the compilers of [24,22] only achieve *statistical security* and their communication complexity grows linearly with the *circuit size* of the functionality. The same holds for more recent unconditionally secure MPC protocols in the preprocessing model [7,13]. This leaves open the following natural questions:

Question 1. What is the *communication complexity* of unconditionally secure computation in the preprocessing model? Can the communication be made independent of the circuit size?

Question 2. Are there general protocols in the preprocessing model that achieve *perfect security* against malicious parties?

While the first question is clearly motivated by the goal of (theoretical and practical) efficiency, we argue that this is also the case for the second question. Consider a scenario where two parties wish to securely evaluate a functionality $f(x, y)$ where x and y are taken from small input domains. Viewing the input size as constant, it can be shown that the asymptotic complexity of any statistically secure protocol with simulation error of $2^{-\sigma}$ must grow (at least) linearly with σ, whereas any perfectly secure protocol has constant complexity. Finally, the question of perfect security is conceptually interesting, as there are very few examples of perfectly secure cryptographic protocols with security against malicious parties.

Our Results: We essentially settle the above questions, obtaining both positive and negative results on unconditionally secure computation with correlated randomness. In doing so, we present a number of efficient protocols that can be useful in practice, especially when securely computing (many instances of) "unstructured" functions on small input domains. Concretely, we obtain the following results.

Communication complexity. We show that any multiparty functionality can be realized, with perfect security against semi-honest parties or statistical security against malicious parties, by a protocol in which the number of bits communicated by each party is linear in its input length. A disadvantage of our protocols is that their storage complexity (i.e., the number of bits each party receives during preprocessing) grows exponentially with the input length. We give evidence that this disadvantage is inherent even when the honest parties are computationally unbounded. Concretely, if every two-party functionality had a protocol

with polynomial storage complexity, this would imply an unexpected answer to a longstanding open question on the complexity of information-theoretic private information retrieval [12] (see Theorem 14).

We also prove a separation between the communication pattern required by unconditionally secure MPC in the preprocessing model and the communication with no security requirement. Concretely, for most functionalities (even ones with a short output) it is essential that the communication by *each party* grows linearly with its input length. In contrast, without security requirements it is always possible to make the communication by one of the parties comparable to the length of the output, independently of the input length. The same is true in the computational model of security under standard cryptographic assumptions. Concretely, such a communication pattern is possible either without preprocessing using fully homomorphic encryption [19], or with preprocessing by using garbled circuits [34] (provided that the inputs are chosen independently of the correlated randomness [5]).

Perfect security. We show that any "sender-receiver" functionality, which takes inputs from both parties and delivers an output only to the receiver, can be *perfectly* realized in the preprocessing model. In contrast, we show that perfect security is generally impossible for functionalities which deliver outputs to both parties, even for non-reactive functionalities and even if one settles for "security with abort" without fairness (Thm. 4). A similar impossibility result for bit commitment (a reactive functionality) was obtained in [8].

The communication and storage complexity of our perfectly secure protocols are comparable to those of the statistical protocols, except for eliminating the dependence on a security parameter. In particular, the storage complexity grows exponentially with the bit-length of the inputs. We present storage-efficient protocols for several natural functionalities, including string equality (see Section 1.2 below), set intersection, and inner product (Appendix A). Our positive results for general functionalities are summarized in Table 1, and for specific sender-receiver functionalities in Table 2.

Protocol	Communication	Storage	Parties	Security
[20,3]	$O(s)$	$O(s)$	k	perfect, passive
[24,22]	$O(s) + \mathsf{poly}(\sigma)$	$O(s) + \mathsf{poly}(\sigma)$	k	statistical, active
Theorem 1	$O(n)$	$O(2^n m)$	k	perfect, passive
Theorem 2	$O(n + \sigma)$	$O(2^n(m + \sigma))$	k	statistical, active
Theorem 3	$O(n)$	$O(2^n(m + n))$	2	perfect, active

Table 1. Comparison of our positive results with previous work: s is the size of a boolean circuit computing the functionality, n is the length of the inputs, m is the output length, and σ is a statistical security parameter. In the asymptotic complexity expressions, the number of parties k is viewed as constant. The protocol of Theorem 3 applies only to sender-receiver two-party functionalities.

Protocol	f	Communication	Storage	Computation	Security
Sec. 1.2	$x =_? y$	$2\|x\|$	$O(\|x\|)$	$\mathsf{poly}(\|x\|)$	perfect, active
Thm. 7	$x \cap y$	$\mathsf{poly}(\|x\|) + \|y\|$	$\mathsf{poly}(\|x\|)$	$\exp(\|x\|, \|y\|)$	perfect, active
Thm. 7	$x \cap y$	$\mathsf{poly}(\|x\|, k) + \|y\|$	$\mathsf{poly}(\|x\|, \sigma)$	$\mathsf{poly}(\|x\|, \|y\|, \sigma)$	statistical, active
Thm. 8	$\langle x, y \rangle$	$2\|x\|$	$O(\|x\|)$	$\mathsf{superpoly}(\|x\|)$	perfect, active

Table 2. Sender-receiver protocols for specific tasks. Two variants of set intersection are given: a perfectly secure with exponential computation, and a statistically secure with efficient computation.

Perfect correctness in the plain model. We present a somewhat unexpected application of our positive results in the preprocessing model to security in the plain model. Consider the goal of securely evaluating a sender-receiver functionality f. We say that a protocol for f is *perfectly correct* if the effect of any (unbounded) malicious sender strategy on the honest receiver's output can be perfectly simulated, via some distribution over the sender's inputs, in an ideal evaluation of f. For example, consider the string equality functionality $f(x, y)$ which receives an n-bit string from each party, and delivers 1 to the receiver if $x = y$ and 0 otherwise. A perfectly correct protocol for f should guarantee, for instance, that if the honest receiver picks its input at random, then the receiver should output 1 with *exactly* 2^{-n} probability, no matter which strategy the sender uses.

The impossibility of perfectly sound zero-knowledge proofs (which carries over to the preprocessing model, see Theorem 15) shows that perfect correctness cannot always be achieved when the honest parties are required to be efficient. We complement this by a positive result which applies to *all* functionalities on a small input domain as well as some natural functionalities on a large input domain (like string equality). Our result is based on a general approach for transforming perfectly secure protocols for sender-receiver functionalities in the preprocessing model into (computationally) secure protocols in the plain model which additionally offer perfect correctness against a malicious sender.

To summarize, we have the following *lower bounds:*

- We show limits to what functionalities can be implemented perfectly. Theorem 4 shows that not all two-party functionalities have protocols with perfect security and abort. This is generalized in Theorem 9 to show a function that requires $\Omega(\log \frac{1}{\epsilon})$ communication to compute with ϵ-security.
- We lower bound the amount of communication that a secure protocol for a non-trivial functionality must use. Theorem 11 for the perfect case and Theorems 12, 13 for the statistical case show that for general functionalities the communication complexity of our protocols is optimal. Another generalization (Theorem 10) shows that the negative results extends to the case of expected round complexity.
- We show that superpolynomial preprocessing is needed in general. Theorem 14 explains that improving on the preprocessing needed for sender-receiver functionalities will imply a breakthrough in information theoretic PIR.

On the positive side, we show in Theorem 5 and 6 how to use perfectly secure sender-receiver protocols in the preprocessing model to implement *perfectly correct* protocols in the plain model (if the preprocessing is small enough).

1.1 Related Work

Beaver [3] showed that OT can be realized with perfect security given preprocessing. Later Beaver [4] generalized the above to the *commodity-based model*, a setting where there are multiple servers providing precomputed randomness, only a majority of which are honest (in the full version, we describe a general approach for applying our results in the commodity-based model). Beaver also notes that perfect security is not possible in general because commitment cannot be realized perfectly, and a proof of this appeared in [8]. However, the question was left open for standard (non-reactive) functionalities.

Since OT can be precomputed [3] and as it is complete for secure computation [24], it is possible to compute any function with statistical security. The result of [22] improves the asymptotical complexity of [24], while [2,7,27] offer efficient statistically secure protocols in the preprocessing model for arithmetic and Boolean circuits respectively. A recent result [14] shows that this can be done with no overhead during the online phase by giving a protocol with optimal communication complexity for the case of "generic preprocessing" (i.e., the preprocessing does not depend on the function to be evaluated – only on its size). Our results achieve better online communication complexity as we do not rely on a circuit representation.

A protocol for computing secret shares of the inner product against malicious adversaries was proposed in [15]. In Appendix A, we give a protocol for computing the inner product where one party learns the output. In the setting of malicious corruptions, it is not trivial to reconstruct the results from the shares, and therefore our protocol takes a substantially different approach than [15].

In [32], a perfectly secure protocol for oblivious polynomial evaluation in the preprocessing model is presented. [32] also presents a protocol for equality which is claimed to be perfectly secure but it is however not perfectly secure according to the standard simulation-based definition — see Section 1.2 below for a perfectly secure protocol for equality.

The type of correlated randomness needed for realizing multiparty computation with unconditional security in the presence of an honest majority is studied in [17,18]. Statistically secure commitment protocol from correlated randomness are constructed in [31]. Finally [33] gives linear lower bounds on the storage complexity of secure computation with correlated randomness.

1.2 Warmup: Equality Test

To introduce some of the notation and the techniques that we use later to prove more general results, we describe the simple protocol in Figure 1 for equality testing in the preprocessing model.

Functionality:

- The receiver has input $x \in X$, the sender input $y \in X$;
- The receiver learns 1 if $x = y$ or 0 otherwise. The sender learns nothing;

Preprocessing:

1. Sample a random 2-wise independent permutation $P : X \to X$, and a random string $r \in_R X$. Compute $s = P(r)$;
2. The preprocessing outputs (r, s) to the receiver and P to the sender;

Protocol:

1. The receiver computes $u = x + r$ and sends to sender;
2. The sender computes $v = P(u - y)$ and sends to the receiver;
3. The receiver outputs 1 if $v = s$, and 0 otherwise;

Figure 1. A perfectly secure protocol for equality with preprocessing.

We consider at the *sender-receiver* version of the functionality, where only the receiver gets output from the protocol. In this setting, we have a receiver and a sender holding respectively x, y in some group X. At the end of the protocol, the receiver learns whether $x = y$ or not. The protocol achieves perfect security against malicious adversaries and it is optimal in terms of communication complexity. Correctness follows from $v = P(u - y) = P(r + x - y)$, and this is equal to s iff $x = y$.

One can prove that the protocol is perfectly secure by a simulation argument: The simulator has access to all preprocessed information. In case of a corrupted *sender*, the simulator proceeds as follows: the simulator sends a random u to the adversary and, when the adversary replies v, the simulator computes $y = u - P^{-1}(v)$ and inputs it to the ideal functionality. In case of a corrupted *receiver*, the simulator extracts the input string x (using u, r) and inputs it to the ideal functionality for equality. If the ideal functionality outputs 1, the simulator sends $v = s$ to the corrupted receiver, but if it outputs 0, the simulator chooses $v \in_R X$ such that $v \neq s$. This simulation is perfect, as the adversary's view of the protocol is distributed identically both in the real execution and in the simulation. Note that it is enough for P to be drawn from a family of pairwise independent permutations, since the receiver only learns the permutation at two indices.

The protocol is also UC-secure (the simulation is straight line) and is *adaptively* secure. This is the case for all the protocols presented in this work.

2 Preliminaries

Notation. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. We use $Z^{X \times Y}$ to denote the set of matrices over Z whose rows are labeled by the elements of X and whose columns are labeled by the elements of Y.

Computational model. We assume perfect uniform sampling from $[m]$, for any positive integer m, as an atomic computational step.

Network model. We consider protocols involving n parties, denoted $P_1, \ldots, P_n$. The parties communicate over synchronous, secure and authenticated point-to-point channels. In some constructions we also use a broadcast channel. We note that, in the preprocessing model, all these channels can be implemented with unconditional security over insecure point-to-point channels. Specifically, secure channels can be perfectly implemented in the preprocessing model using a one-time pad, authentication (with statistical security) using a one-time message authentication code (MAC), and broadcast (with statistical security) using the protocol of [29].

Functionalities. We consider non-reactive secure computation tasks, defined by a deterministic or randomized *functionality* $f : X_1 \times \ldots \times X_n \to Z_1 \times \ldots \times Z_n$. The functionality specifies a mapping from n inputs to n outputs which the parties want to compute. We will often consider a special class of two-party functionalities referred to as *sender-receiver functionalities*. A sender-receiver functionality $f : X \times Y \to Z$ gets an input x from P_1 (the *receiver*), an input y from P_2 (the *sender*) and delivers the output z only to the receiver.

Protocols with preprocessing. An n-party protocol can be formally defined by a *next message function*. This function, on input (i, x_i, r_i, j, m), specifies an n-tuple of messages sent by party P_i in round j, when x_i is its inputs, r_i is its randomness and m describes the messages it received in previous rounds. (If a broadcast channel is used, the next message function also outputs the message broadcasted by P_i in Round j.) The next message function may also instruct P_i to terminate the protocol, in which case it also specifies the output of P_i. In the *preprocessing model*, the specification of a protocol also includes a joint distribution $\mathcal{D}$ over $R_1 \times R_2 \ldots \times R_n$, where the R_i's are finite randomness domains. This distribution is used for sampling correlated random inputs $(r_1, \ldots, r_n)$ which the parties receive before the beginning of the protocol (in particular, the preprocessing is independent of the inputs). The next message function, in this case, may also depend on the private random input r_i received by P_i from $\mathcal{D}$. We assume that for every possible choice of inputs and random inputs, all parties eventually terminate.

Security definition. We work in the standard ideal-world/real-world simulation paradigm. Our positive results hold for the strongest possible security model, namely UC-security with adaptive corruptions, while our negative results hold for the weaker model of standalone security against static corruptions. We consider both semi-honest (passive) corruptions and malicious (active) corruptions. Using the standard terminology of secure computation, the preprocessing model can be thought of as a *hybrid model* where the parties have a one-time access to an ideal randomized functionality $\mathcal{D}$ (with no inputs) providing them with correlated, private random inputs r_i. We consider by default *full security* (with guaranteed output delivery) for sender-receiver functionalities, and *security with abort* for general functionalities. We mainly focus on the cases of *statistical* or *perfect* security, though some of our results refer to computational security as well. We will sometimes refer separately to *correctness* and *privacy* – the former considers only the effect of the adversary on the outputs and the latter considers

only the view of the adversary. The full security definition is omitted for lack of space.

3 Optimal Communication For General Functionalities

In this section, we settle the communication complexity of MPC in the preprocessing model. For simplicity, we restrict the attention to non-reactive functionalities, but the results of this section apply also to reactive functionalities.

3.1 Upper Bounds on Communication Complexity

The following is a summary of our upper bounds. These will follow from the Claims 1, 2 and 3 (some of which are in later sections) and by inspection of the protocols.

Theorem 1. *For any n-party functionality $f : X_1 \times \ldots \times X_n \to Z_1 \times \ldots \times Z_n$, there is a protocol π which realizes f, in the preprocessing model, and has the following features against semi-honest parties: (1) π is perfectly secure; (2) It uses two rounds of communication; (3) Let $\alpha = \sum_{i\in[n]} \log |X_i|$ be the total input length. Then, the total communication complexity is $O(\alpha)$ and the storage complexity is $O(\alpha 2^\alpha)$.*

Theorem 2. *For any n-party functionality $f : X_1 \times \ldots \times X_n \to Z_1 \times \ldots \times Z_n$ and $\epsilon > 0$, there is a protocol π which realizes f, in the preprocessing model against a malicious adversary, such that: (1) π is statistically ϵ-secure with abort; (2) It uses two rounds of communication (given broadcast); (3) The total communication complexity is $O(\alpha + n \log 1/\epsilon)$ and the storage complexity is $O(2^\alpha \cdot (\alpha + n \log 1/\epsilon))$, where α being the total input length, as above.*

Theorem 3. *For any 2-party sender-receiver functionality $f : X \times Y \to Z$, there is a protocol π which realizes f, in the preprocessing model against a malicious adversary, such that: (1) π is perfectly secure; (2) It uses two rounds of communication; (3) The total communication complexity is $\log |X| + \log |Y|$ and the storage complexity is $O(|X| \cdot |Y| \cdot \log |Y|)$.*

3.2 Semi-Honest Two-Party Protocol, via One-Time Truth Table

For the sake of exposition, we focus on protocols where both parties P_1, P_2 receive the same output $f(x, y) \in Z$, for some function $f : X \times Y \to Z$. We view X, Y and Z as groups and use additive notation for the group operation, i.e., $(X, +), (Y, +), (Z, +)$.

In Figure 2 we present a simple protocol that is secure against a semi-honest adversary if the parties have access to a preprocessing functionality dealing correlated randomness. The protocol has communication complexity $\log |X| + \log |Y| + 2 \log |Z|$. A protocol with communication complexity $\log |X| + \log |Y| + \log |Z|$

follows from the protocol in Section 4.[3] We start by presenting this slightly less efficient protocol here, as this protocol is easier to generalize for security against malicious parties and for the multiparty case (see Figure 3 and the full version).

The protocol uses *one-time truth tables* (OTTT). Intuitively, OTTT can be seen as the one-time pad of secure function evaluation. The parties hold shares of a permuted truth-table, and each party knows also the permutation that was used for its input. In the two-party case, the truth-table can be seen as a matrix, where one party knows the permutation of the rows and the other knows the permutation of the columns. In fact, given that every truth table will be only used once, a random cyclic-shift can be used instead of a random permutation.

Functionality:

- P_1 has input $x \in X$, P_2 has input $y \in Y$.
- Both parties learn $z = f(x, y)$.

Preprocessing:

1. Sample random $r \in X, s \in Y$ and let A be the permuted truth table; i.e.,

$$A_{x+r,y+s} = f(x, y) \ ;$$

2. Sample a random matrix $M^1 \in Z^{X \times Y}$ and let $M^2 = A - M^1$;
3. Output (M^1, r) to P_1 and (M^2, s) to P_2;

Protocol:

1. P_1 sends $u = x + r$ to P_2;
2. P_2 sends $v = y + s$ and $z_2 = M^2_{u,v}$ to P_1;
3. P_1 sends to P_2 the value $z_1 = M^1_{u,v}$;
4. Both parties output $z = z_1 + z_2$;

Figure 2. Semi-Honest Secure Protocol using One-Time Truth Table

Claim 1. The protocol in Figure 2 securely computes f with perfect security against semi-honest corruptions.

Proof. When both parties are honest, the protocol indeed outputs the correct value:

$$z = z_1 + z_2 = M^1_{u,v} + M^2_{u,v} = A_{u,v} = A_{x+r,y+s} = f(x, y).$$

Security against semi-honest parties can be argued as follows: the view of P_2 can be simulated by choosing a random $u \in X$ and defining $z_1 = z - M^2_{u,v}$. The view of P_1 can be simulated by choosing a random $v \in Y$ and defining $z_2 = z - M^1_{u,v}$. As both in the simulation and in the real protocol the values u, v, z_1, z_2 are distributed uniformly at random in the corresponding domains, the protocol achieves perfect security.

[3] The protocol of Section 4 has complexity $\log |X| + \log |Y|$ but only one party gets output; however, in the semi-honest case, this party may simply transfer the output ($\log |Z|$ bits) to the other party.

3.3 One-Time Truth Tables with Malicious Security

The above protocol is only secure against semi-honest adversaries, as a malicious party P_i could misreport its output share z_i and therefore change the output distribution. To fix this problem, we will enhance the OTTT protocol using information theoretic message authentication codes (MAC): the preprocessing phase will output keys for a one-time MAC to both parties, and will add shares of these MACs to the truth table. The resulting protocol is only statistically secure as an adversary will always have a (negligibly small) probability to output a fake share z_i together with a valid MAC. As we will see later (Section 4.1), this is inherent; i.e., it is impossible to securely compute every function with perfect security, even in the preprocessing model.

Definition 1 (One-Time MAC). *A pair of efficient algorithms* $(\mathsf{Tag}, \mathsf{Ver})$ *is a* one-time ϵ-secure message authentication code scheme *(MAC), with key space* $\mathcal{K}$ *and MAC space* $\mathcal{M}$*, if* $\mathsf{Ver}_k(m, \mathsf{Tag}_k(m)) = 1$ *with probability* 1 *and for every (possibly unbounded) adversary* $\mathcal{A}$*:*

$$\Pr[k \leftarrow \mathcal{K}, m \leftarrow \mathcal{A}, (m', t') \leftarrow \mathcal{A}(m, \mathsf{Tag}_k(m)) : \mathsf{Ver}_k(m', t') = 1 \wedge m \neq m'] \ < \ \epsilon \ .$$

The MAC can be instantiated with the standard "$am + b$" construction: Let $\mathbb{F}$ be a finite field of size $|\mathbb{F}| > \epsilon^{-1}$, and let $k = (a, b) \in \mathbb{F}^2$. To compute a MAC tag, let $\mathsf{Tag}_k(m) = am + b$ and for verification compute $\mathsf{Ver}_k(m, t) = 1$ iff $t = am + b$. Without loss of generality, the range of f, the function to be computed, is $Z = \mathbb{F}$. [4] For the purpose of this application, we will write the MAC space $\mathcal{M}$ as an additive group $(\mathcal{M}, +)$.

MAC enhanced OTTT: In Figure 3, the protocol for general two-party computation in the preprocessing model using OTTT is presented. Note that, as all the MAC signatures are secret-shared and only one is reconstructed, we can use the same MAC key for all the entries in the matrix. We assume, for notational simplicity, that both parties obtain the same output z; the general case may be handled similarly.

Claim 2. The protocol in Figure 3 computes f with ϵ-security against a malicious adversary.

The proof of this claim is pretty straightforward and is therefore deferred to the full version of this paper. In the full version we also show how this protocol can be generalized to n parties. The main issue here is to have all the honest parties to output the same value (in particular, if one honest party outputs $\perp$ then all honest parties must output $\perp$). This is done using *unanimously identifiable commitments* from [21,28].

In the full version, we also prove negative results which complement the above positive results (see also Appendix B). In particular, we show that the communication complexity of the above protocols is optimal (for non-trivial functions) and give evidence that the exponential storage complexity (or randomness complexity) is inherent.

[4] Note that we still only need $\log_2 |\mathrm{Im}(f)|$ bits to encode the output of the function.

Functionality:

- P_1 has input $x \in X$ and P_2 has input $y \in Y$.
- Both parties learn $f(x, y)$.

Preprocessing:

1. Sample random keys for an ϵ-secure MAC scheme $k_1, k_2 \in \mathcal{K}$;
2. Sample random $r \in X, s \in Y$ and let $A \in (Z \times \mathcal{M} \times \mathcal{M})^{X \times Y}$ be a matrix s.t.
$$A_{x+r,y+s} = \left(f(x,y), \mathsf{Tag}_{k_1}(f(x,y)), \mathsf{Tag}_{k_2}(f(x,y))\right) \ ;$$
3. Sample a random matrix $M^1 \in (Z \times \mathcal{M} \times \mathcal{M})^{X \times Y}$ and let $M^2 = A - M^1$;
4. Output (M^1, r, k_1) to P_1 and (M^2, s, k_2) to P_2;

Protocol:

1. P_1 sends $u = x + r$ to P_2;
2. P_2 sends $v = y + s$ and $z_2 = M^2_{u,v}$ to P_1;
3. P_1 sends $z_1 = M^1_{u,v}$ to P_2;
4. Each party P_i parses $z_1 + z_2$ as (z, t_1, t_2);
5. If $\mathsf{Ver}_{k_i}(z, t_i) = 1$, party P_i outputs z, otherwise it outputs $\perp$;

Figure 3. Malicious Secure Protocol using One-Time Truth Table

4 Perfect Security for Sender-Receiver Functionalities

In this section we show that, if only one party receives output, it is possible to achieve *perfect* security even against a *malicious* adversary. We will show, in Section 4.1, that this is not the case for general functionalities where all parties receive outputs.

The protocol is presented in Figure 4. The structure of the protocol is similar to previous constructions, in the sense that the preprocessing samples some random permutations, and then during the online phase the parties apply the random permutations on their inputs and exchange the results. However, the protocol uses the asymmetry between the sender and receiver: every row of the truth table (corresponding to each input of the receiver) is permuted using a different random permutation. The sender learns this set of permutations, permuted under a receiver permutation (implemented by a random circular shift, as in previous constructions). The receiver learns the truth table where each row is permuted according to the corresponding permutation.

In the online phase, the sender uses the first message of the receiver to determine which of the permutation to apply to his input. The receiver, using this value, can perform a look-up in the permuted truth table and output the correct result. The protocol is intuitively perfectly private as both parties only see each other's input through a random permutation. Perfect correctness is achieved because, in contrast to previous constructions, every message sent by the sender uniquely determines its input (together with the preprocessing information).

Claim 3. The protocol in Figure 4 securely computes the sender-receiver functionality f with perfect security and optimal communication complexity against malicious corruptions.

Functionality:

- R has input $x \in X$, S has input $y \in Y$.
- R learns $z = f(x, y)$;

Preprocessing:

1. Sample random $r \in X$;
2. Sample random permutations $\{P_x\}_{x \in X}$ with $P_x : Y \to Y$, and let $\{Q_i\}_{i \in X}$ be a "shifted" sequence of those permutations, where $Q_{x+r} = P_x$;
3. Compute the permuted truth table $A_{x,P_x(y)} = f(x, y)$;
4. Output (A, r) to R and $\{Q_i\}_{i \in X}$ to S;

Protocol:

1. R sends $u = x + r$ to S;
2. S sends $v = Q_u(y)$ to R;
3. R outputs $f(x, y) = A_{x,v}$ (if $v \notin Y$, then R outputs $f(x, y_0)$, for some fixed value y_0);

Figure 4. Perfect Secure Protocol for Sender-Receiver Functionalities, Malicious Adversaries

Proof. When both parties are honest the output is correct:

$$A_{x,v} = A_{x,Q_u(y)} = A_{x,P_x(y)} = f(x, y).$$

If S^* is a corrupted sender, the simulator samples the preprocessing for S consisting of the permutations $\{Q_i\}_{i \in X}$. The simulator then picks a random message u, as the first message of the protocol. Then, it runs $v \leftarrow S^*(\{Q_i\}, y, u)$, and it extracts an effective input $y' = Q_u^{-1}(v)$ and inputs y' to the ideal functionality to get the ideal-world output $z = f(x, y')$. The simulator outputs the simulated view $(\{Q_i\}, v)$. Observe that u and $\{Q_i\}$ are distributed as in the real world and independently of x. The simulated view considered jointly with f's output on the effective input (i.e., z) is thus distributed identically to the view of S^* jointly with the receiver's output, in the real-world execution.

For a corrupted receiver R^*, the simulator samples $A, r, \{Q_i\}$, runs $u \leftarrow R^*(A, r, x)$, extracts $x' = u - r$, inputs it to the ideal functionality, receives $z = f(x', y)$, computes $u = A_{x,v}^{-1}(z)$ and outputs the simulated view (A, r, u). This is distributed identically to the real-world view of R^*.

Note that even for the case of semi-honest security, this protocol is more efficient than a protocol using 1-out-of-n OT, where the sender acts as the transmitter and offers $f(x_1, y), \ldots, f(x_m, y)$ for each possible $x_i \in X$ and the receiver acts as the chooser and selects x. Such protocol would have (online) communication complexity $O(|X| \log |Z|)$, while our protocol requires only $\log(|X|) + \log(|Y|)$ bits of communication.

4.1 Impossibility of Perfect Security for General Functionalities

The following theorem shows that the above positive result cannot be extended to general functionalities (see Appendix B and the full version for a tight tradeoff between communication and error probability).

Theorem 4. *Let $f(x_1, x_2) = (x_1 \oplus x_2, x_1 \oplus x_2)$. Then, there is no protocol for f, in the preprocessing model, which is* perfectly secure with abort*.*

Proof. Assume towards a contradiction that π perfectly realizes f with abort given preprocessing $\mathcal{D}$. Consider the experiment of running π on a uniformly random choice of inputs $(x_1, x_2) \in \{0,1\}^2$ and correlated random inputs (r_1, r_2) drawn from $\mathcal{D}$. Let i_1 be the *minimal* number such that, at the beginning of round i_1, the output of P_1 is always *determined* (over all choices of inputs and random inputs) regardless of subsequent messages (which may possibly be sent by a malicious P_2^*). That is, when running the above experiment, before round i_1, party P_1 may have an uncertainty about the output; but, at the beginning of round i_1, the view of P_1 always determines a unique output value $b \in \{0,1\}$ such that P_1 will either output b or $\perp$. The value i_2 is defined symmetrically. Note that i_1, i_2 are well defined, because the outputs are always determined at the end of the execution, and, moreover, they are distinct because only one party sends a message in each round.

Assume, without loss of generality, that $i_1 < i_2$ and, moreover, that in the above experiment there is an execution which terminates with P_2 outputting 0, but where in the beginning of round i_1 the output of P_2 is not yet determined (namely, there are messages of P_1^* that would make it output 1).

We can now describe a malicious strategy P_1^* that would make an honest P_2, on a random input x_2, output 1 with probability $p > 1/2$. Since this is impossible in the ideal model, we get the desired contradiction. The malicious P_1^* proceeds as follows.

- Run the protocol honestly on a random input x_1 until the beginning of round i_1. Let b be the output value determined at this point.
- If $b = 1$, continue running the protocol honestly.
- Otherwise, continue the protocol by sending a uniformly random message in each round.

In the event that $b = 1$, which occurs with probability $1/2$, P_2 will always output 1. In the event that $b = 0$, by the above assumption there exist subsequent messages of P_1^* making P_2 output 1, and hence also in this case P_2 outputs 1 with nonzero probability. Overall P_2 outputs 1 with probability $p > 1/2$.

5 Perfect Correctness in the Plain Model

Theorem 15 shows that the impossibility of perfectly sound zero-knowledge proofs for NP carries over to the preprocessing model. This implies that some sender-receiver functionalities cannot be securely realized with perfect correctness in the plain model. In this section, we show that the class of functionalities that can be securely realized with perfect correctness is actually quite rich. To the best of our knowledge, this important fundamental question has been neglected in the literature so far.

We present a general transformation from perfect sender-receiver protocols in the preprocessing model, to protocols with perfect correctness in the plain model.

This is possible for functionalities for which the preprocessing can be realized in the plain model with perfect privacy (and computational correctness): the main conceptual contribution is to show how we can turn perfect privacy into perfect correctness by using an offline/online protocol.

The high level idea of this transformation is to use the "reversibility" of correlated randomness for turning perfect privacy in the plain model into perfect correctness in the plain model. Concretely, let π be a perfectly secure protocol for f in the preprocessing model. Using standard techniques (a combination of perfectly private OT protocols [26,1] with an information-theoretic variant of the garbled circuit technique [35,24]), one can get a *perfectly private* protocol π' (with unbounded simulation) for all sender-receiver functionalities in NC^1. We then use π', with the sender in π playing the role of the receiver in π', for generating the correlated randomness required by π. In this subprotocol the receiver picks its randomness r_x from the correct marginal distribution and the sender obtains as its output from π' a random input r_y sampled from the conditional distribution defined by r_x. This subprotocol prevents a malicious sender from learning *any* information about r_x other than what follows from r_y. Running π on top of the correlated randomness (r_x, r_y) generated by the subprotocol gives a perfectly correct protocol for f.

The approach described so far only guarantees security against semi-honest parties (in addition to perfect correctness against a malicious sender); however, using a GMW-style compiler we get (computational) security against malicious parties while maintaining perfect correctness against a malicious unbounded sender.

Formally, let $\mathcal{D}$ be a distribution over $R_1 \times R_2$ such that all probabilities in the support of $\mathcal{D}$ are rational. Let $\mathcal{D}_{r_1}$ be a family of distributions over R_2 such that the two distributions $\{(r_1, r_2) : (r_1, r_2) \leftarrow \mathcal{D}\}$ and $\{(r_1, r_2') : (r_1, r_2) \leftarrow \mathcal{D}, r_2' \leftarrow \mathcal{D}_{r_1}\}$ are identically distributed. Let $\mathsf{Pre}^{\mathcal{D}} : R_1 \to R_2$ be a randomized functionality[5] that, on input r_1 from party R outputs r_2, sampled according to $\mathcal{D}_{r_1}$ to S (if r_1 is not in the support of $\mathcal{D}$, the function outputs $\perp$). Applications and proofs of the following theorems are discussed in the full version.

Theorem 5. *Let f be a sender-receiver functionality that admits a perfectly secure protocol π_{online}, in the presence of preprocessing $\mathcal{D}$, where all probabilities in support($\mathcal{D}$) are rational. Let π_{pre} be a protocol that realizes $\mathsf{Pre}^{\mathcal{D}}$ which is semi-honest secure and perfectly private against malicious S.*
Then, it is possible to securely compute f with semi-honest security and perfect correctness*.*

Theorem 6. *Assuming one-way permutations exist, the result of Theorem 5 holds with security against malicious parties.*

[5] As discussed in Section 2, our computational model allows perfect sampling from the uniform distribution over $[m]$, for all integers m.

References

1. W. Aiello, Y. Ishai, and O. Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, pages 119–135, 2001.
2. D. Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, pages 420–432, 1991.
3. D. Beaver. Precomputing oblivious transfer. In *CRYPTO*, pages 97–109, 1995.
4. D. Beaver. Commodity-based cryptography (extended abstract). In *STOC*, pages 446–455, 1997.
5. M. Bellare, V. T. Hoang, and P. Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In *ASIACRYPT*, pages 134–153, 2012.
6. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.
7. R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, pages 169–188, 2011.
8. C. Blundo, B. Masucci, D. R. Stinson, and R. Wei. Constructions and bounds for unconditionally secure non-interactive commitment schemes. *Des. Codes Cryptography*, 26(1-3):97–110, 2002.
9. D. Bogdanov, R. Talviste, and J. Willemson. Deploying secure multi-party computation for financial data analysis - (short paper). pages 57–64, 2012.
10. P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. P. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, M. I. Schwartzbach, and T. Toft. Secure multiparty computation goes live. In *Financial Cryptography*, pages 325–343, 2009.
11. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.
12. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *FOCS*, pages 41–50, 1995.
13. I. Damgård, V. Pastro, N. P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pages 643–662, 2012.
14. I. Damgård and S. Zakarias. Constant-overhead secure computation of boolean circuits using preprocessing. In *TCC*, 2013.
15. R. Dowsley, J. van de Graaf, D. Marques, and A. C. A. Nascimento. A two-party protocol with trusted initializer for computing the inner product. In *WISA*, pages 337–350, 2010.
16. S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.
17. M. Fitzi, N. Gisin, U. M. Maurer, and O. von Rotz. Unconditional byzantine agreement and multi-party computation secure against dishonest minorities from scratch. In *EUROCRYPT*, pages 482–501. 2002.
18. M. Fitzi, S. Wolf, and J. Wullschleger. Pseudo-signatures, broadcast, and multiparty computation from correlated randomness. In *CRYPTO*, pages 562–578. 2004.
19. C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
20. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

21. Y. Ishai, R. Ostrovsky, and H. Seyalioglu. Identifying cheaters without an honest majority. In *TCC*, 2012.
22. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.
23. E. Kaplan, M. Naor, and O. Reingold. Derandomized constructions of k-wise (almost) independent permutations. volume 55, pages 113–133. 2009.
24. J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.
25. G. Kuperberg, S. Lovett, and R. Peled. Probabilistic existence of rigid combinatorial structures. pages 1091–1106, 2012.
26. M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.
27. J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra. A new approach to practical active-secure two-party computation. In *CRYPTO*, pages 681–700, 2012.
28. A. Patra, A. Choudhary, and C. P. Rangan. Round efficient unconditionally secure mpc and multiparty set intersection with optimal resilience. In *INDOCRYPT*, pages 398–417, 2009.
29. B. Pfitzmann and M. Waidner. Information-theoretic pseudosignatures and byzantine agreement for $t \geq n/3$. *IBM Research Report RZ 2882 (#90830)*, 1996.
30. M. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Harvard Aiken Computation Laboratory, 1981.
31. R. Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. *Manuscript*, 1999.
32. R. Tonicelli, R. Dowsley, G. Hanaoka, H. Imai, J. Müller-Quade, A. Otsuka, and A. C. A. Nascimento. Information-theoretically secure oblivious polynomial evaluation in the commodity-based model. *IACR Cryptology ePrint Archive*, 2009:270, 2009.
33. S. Winkler and J. Wullschleger. On the efficiency of classical and quantum oblivious transfer reductions. In *CRYPTO*, pages 707–723, 2010.
34. A. C. Yao. How to generate and exchange secrets. pages 162–167, 1986.
35. A. C.-C. Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.

A Protocols For Specific Tasks

In this section, we present protocols for a number of specific sender-receiver tasks. We focus on the case of perfect security in the malicious model. All of these protocols have a 2-move structure: the receiver sends a message m_X, the sender replies with a message m_Y, and the receiver computes its output. Due to space limitations the proofs are deferred to the full version.

A.1 Set Intersection

In Figure 5, we present a protocol for computing the intersection of two sets x, y of fixed sizes (k, l, respectively) over some domain U.

Theorem 7. *The protocol in Figure 5 realizes the sender-receiver functionality* set intersection *with perfect security.*

Functionality:

- R has input x consisting of k distinct elements $x_1, \ldots, x_k \in U$; S has input y consisting of l distinct elements $y_1, \ldots, y_l \in U$;
- R learns the intersection $z = x \cap y$.

Preprocessing:

1. Pick a random permutation $P : U \to U$ and k distinct elements $r_1, \ldots, r_k \in_R U$;
2. S gets P and R gets $r_1, \ldots, r_k$ and $s_1 = P(r_1), \ldots, s_k = P(r_k)$;

Protocol:

1. R picks a random permutation $Q : U \to U$, under the constraint that $Q(x_i) = r_i$, for all $i \in [k]$. It sends Q to S;
2. S computes $\mathcal{M} = \{P(Q(y_j)) \mid j \in [l]\}$. It sends $\mathcal{M}$ (size-l sorted set) to R;
3. R outputs a set $\mathcal{I}$ consisting of all i such that $s_i \in \mathcal{M}$;

Figure 5. Protocol for Set Intersection

Optimizing the protocol In the above protocol, both the randomness and the first message have size $O(|U| \log |U|)$ (the space it takes to describe a permutation). This may be super-exponential in the input size. Like for the equality protocol, we can optimize by taking advantage of k-wise independent families of permutations, as these may have smaller descriptions (the existence of small permutation families with this property was recently proven in [25], but this is only an existential result. Instead we can use the efficient explicit constructions of [23] but achieve only statistical security).

A.2 Inner product

Let $\mathbb{F}_p$ be a finite field of size p, and let $t \geq 1$. The inner product functionality $\mathrm{IP}_{t,p}$ is as follows:

- S has input $y \in {\mathbb{F}_p}^t$ and R has a linear function $x : {\mathbb{F}_p}^t \to \mathbb{F}_p$, represented by a vector $x \in {\mathbb{F}_p}^t$, so that $x(y) = \langle x, y \rangle = \sum_{i \leq t} x_i y_i$.
- R outputs $x(y)$.

Some Algebraic Preliminaries Let $e_i \in \mathbb{F}^t$ be the i-th unit vector of length t. Let $\mathbb{F}^*$ be the multiplicative group of a finite field $\mathbb{F}$. By default, vectors are column vectors. $\mathbb{F}^{m \times n}$ is the set of $m \times n$ matrices over $\mathbb{F}$. Let $GL(n, p)$ be the group (under matrix multiplication) of invertible matrices in ${\mathbb{F}_p}^{n \times n}$. M_i is the i'th row of a matrix M. Given vectors $v_1, \ldots, v_n \in \mathbb{F}^m$, let $(v_1; \ldots; v_n)$ denote the matrix $M \in \mathbb{F}^{n \times m}$ with rows $M_i = v_i^T$. We will also need the following algebraic primitive:

Definition 2 (Good exhaustive operator). *Let $L : {\mathbb{F}_p}^t \to {\mathbb{F}_p}^t$ be the linear, injective operator defined via $L(y) = y^T L$ (L represents both the operator and the matrix implementing it). Consider the (infinite) sequence $\mathrm{seq}_L = (v, L(v), \ldots, L^{(i)}(v), \ldots)$ generated by L for some v.*

We say that L is a g.e.o. *for $v \in {\mathbb{F}_p}^t$ if:*

1. seq_L *is periodic with period length* $p^t - 1$ *for* v *(i.e., all elements in* $\mathbb{F}_p^t$*, except 0, appear in* seq_L*).*
2. *The the first* t *elements in* seq_L *forms a basis for* ${\mathbb{F}_p}^t$*.*

Lemma 1 (instantiating *good exhaustive operators*). *Consider the operator* L_x *where* $L_x(y) = x \cdot y$ *where* x, y *are viewed as elements of* $\mathbb{F}_{p^t}$ *(the multiplication is over* $\mathbb{F}_{p^t}$*). Viewed as a linear function from* ${\mathbb{F}_p}^t$ *to* ${\mathbb{F}_p}^t$*, we have* $L_x(y) = y^T L_x$*. Let* g *be a generator of* $\mathbb{F}_{p^t}^*$*, then* L_g *is a* g.e.o. *for* $v = e_t$ *(could use any other vector* v*).*

Functionality:
- Inputs: R gets $x \neq 0 \in \mathbb{F}_p^t$, and S gets $y \in \mathbb{F}_p^t$.
- Output: R outputs $\langle x, y \rangle$.

Primitives: Let $L \in {\mathbb{F}_p}^{t \times t}$ be a *g.e.o.* for e_t. For $a \in \mathbb{F}_p^t \setminus \{0\}$, we let $ind(a)$ denote the index of the first appearance of a in seq_L.

Preprocessing:
1. S gets a random vector $r_2 = y' \in \mathbb{F}_p^t$.
2. R gets (x', p_2), where x' is randomly chosen at $\mathbb{F}_p^t \setminus \{0\}$ and $p_2 = \langle y', x' \rangle$.

Protocol:
1. R sends $\delta = ind(x') - ind(x) \pmod{p^t - 1}$ to S.
2. S sends the vector $m = (e_t^T L^0(y + L^\delta y'), e_t^T L(y + L^\delta y'), \ldots, e_t^T L^{t-1}(y + L^\delta y'))$.
3. R sets $M = (e_t^T; e_t^T L; \ldots; e_t^T L^{t-1})$, $r = x^T M^{-1}$. It outputs $rm - p_2$.

Figure 6. A protocol for $\mathrm{IP}+_{t,p}$

The protocol. In Figure 6, we present a protocol for a slightly modified functionality, $\mathrm{IP}+_{t,p}$, where x is restricted to be non-zero. This also implies a protocol for $\mathrm{IP}_{t,p}$, as on input $x = 0$ the receiver can adopt any input $x' \neq 0$, and output 0 at the end, ignoring the communication.

Theorem 8. *The above protocol is a perfectly secure protocol, with preprocessing, for the functionality* $IP+_{t,p}$*, for all* $t \geq 1$ *and prime* p*. The communication complexity, randomness size and* S*'s work are polynomial in* $|x| = t \log p$*, while* R*'s computation is as hard as finding discrete log in* $\mathbb{F}_{p^t}^*$*.*

B "Teasers" from the Full Version

The full version of this article contains several other results. Due to space limitation, we can only state the theorems here and invite the interested reader to look at the full version for further discussion, proofs and applications of the following theorems.

Theorem 9. *Every protocol with preprocessing Π that ϵ-securely computes the functionality $f(x_1, x_2) = (x_1 \oplus x_2, x_1 \oplus x_2)$ with abort, has communication complexity $\Omega(\log \frac{1}{\epsilon})$.*

Theorem 10. *Let $f(x_1, x_2) = (x_1 \oplus x_2, x_1 \oplus x_2)$. Then there is no protocol for f in the preprocessing model which is* perfectly secure with abort*, having expected communication complexity t, for any $t \in \mathbb{N}$.*

Theorem 11. *Given a perfectly secure protocol for some sender-receiver functionality $f : X \times Y \to Z$ in the semi-honest model, with sender message domain M_Y, and receiver message domain M_X. Then,*

- *If for all $y_1 \neq y_2 \in Y$, there exists $x \in X$ such that $f(x, y_1) \neq f(x, y_2)$, then $|M_Y| \geq |Y|$.*
- *If for every $z_1, z_2 \in Z$ and $x_1 \neq x_2 \in X$, we have $\{y | f(x_1, y) = z_1\} \neq \{y | f(x_2, y) = z_2\}$, then $|M_X| \geq |X|$.*

Theorem 12. *Let $c > 0$ be a constant, and consider a sender-receiver functionality $f : X \times Y \to \{0, 1\}$ where $\log |X| = n$, $\log |Y| = m$. Assume there exists a subset $X' \subseteq X$ of size $c \cdot m$, such that $\{(x', f(x', I_Y))\}_{x' \in X}$ determines I_Y, for all $I_Y \in Y$). Then, there exists $\epsilon > 0$, depending only on c, such that in any ϵ-secure protocol with preprocessing for f in the semi-honest model, the sender-side communication is $\Omega(m)$.*

Theorem 13. *Let $c_1, c_2, c_3 > 0$ be constants such that $(1 + c_3)(1 - c_2) < 1$. Consider a sender-receiver functionality $f : X \times Y \to \{0, 1\}$ where $\log |X| = n$, $\log |Y| = m$ satisfying*

- *For all $x \neq x' \in X$, we have $H(f(x', I_Y) | f(x, I_Y)) \geq c_1$, where I_Y is picked uniformly from Y.*
- *For all $y \neq y' \in Y$, we have $\Pr(f(I_X, y) = f(I_X, y')) \geq c_2$, where I_X is picked uniformly at random.*
- *There exists a subset $Y' \subseteq Y$ of size $(1 + c_3)n$, such that $\{y, f(I_X, y)\}_{y \in Y}$ determines I_X, for all $I_X \in X$).*

Then, there exists $\epsilon > 0$, depending only on the c_i's, such that in any ϵ-secure protocol with preprocessing for f, in the semi-honest model, the receiver-side communication is $\Omega(n)$.

Theorem 14. *Suppose there is a semi-honest statistically secure protocol in the preprocessing model for every sender-receiver functionality $f : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$ with correlated randomness complexity $r(n)$ (i.e., where $r_X, r_Y \in \{0, 1\}^{r(n)}$) and communication complexity $c(n)$. Then, there is a 3-server statistical PIR protocol with communication complexity $O(r(\log N) + c(\log N) + \log N)$, where N is the database size.*

Theorem 15. *If $NP \not\subseteq BPP$, there exists a sender-receiver functionality that cannot be efficiently computed with semi-honest security and perfect correctness (even in the preprocessing model).*