Constant-Rate Oblivious Transfer from Noisy Channels

Yuval Ishai[*] Eyal Kushilevitz[†] Rafail Ostrovsky[‡] Manoj Prabhakaran[§]
Amit Sahai[¶] Jürg Wullschleger[‖]

Abstract

A *binary symmetric channel* (BSC) is a noisy communication channel that flips each bit independently with some fixed error probability $0 < p < 1/2$. Crépeau and Kilian (FOCS 1988) showed that oblivious transfer, and hence general secure two-party computation, can be *unconditionally* realized by communicating over a BSC. There has been a long line of works on improving the efficiency and generality of this construction. However, all known constructions that achieve security against *malicious* parties require the parties to communicate $poly(k)$ bits over the channel for each instance of oblivious transfer (more precisely, $\binom{2}{1}$-*bit*-OT) being realized, where k is a statistical security parameter. The question of achieving a constant (positive) rate was left open, even in the easier case of realizing a single oblivious transfer of a long string.

We settle this question in the affirmative by showing how to realize n independent instances of oblivious transfer, with statistical error that vanishes with n, by communicating just $O(n)$ bits over a BSC. As a corollary, any boolean circuit of size s can be securely evaluated by two parties with $O(s) + poly(k)$ bits of communication over a BSC, improving over the $O(s) \cdot poly(k)$ complexity of previous constructions.

[*]Technion, Haifa, Israel. Work done in part while visiting UCLA. Supported by ERC Starting Grant 259426, ISF grant 1361/10, and BSF grant 2008411. yuvali@cs.technion.il

[†]Technion, Haifa, Israel. Work done in part while visiting UCLA. Supported by ISF grant 1361/10 and BSF grant 2008411. eyalk@cs.technion.il

[‡]University of California, Los Angeles. Research supported in part by DARPA, IBM Faculty Award, Xerox Innovation Group Award, the Okawa Foundation Award, Intel, Teradata, NSF grants 0830803, 0916574, BSF grant 2008411 and U.C. MICRO grant. rafail@cs.ucla.edu

[§]University of Illinois, Urbana-Champaign. Supported by NSF grant CNS 07-47027. mmp@cs.uiuc.edu

[¶]University of California, Los Angeles. Research supported in part from a DARPA/ONR PROCEED award, NSF grants 0916574 and 0830803, a Xerox Foundation Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. sahai@cs.ucla.edu

[‖]Université of Montréal and McGill University. Research supported by the Canada-France NSERC-ANR project FREQUENCY. juerg@wulli.com

1 Introduction

One of the attractive features of modern cryptography is its ability to "turn lemons into lemonade." Indeed, traditional complexity-based cryptography turns *computational intractability*, a major obstacle tackled by Computer Science, into a blessing. The present work is concerned with a similar phenomenon in the context of information-theoretic cryptography: the ability to turn *noise*, a major obstacle tackled by Information Theory, into a blessing.

Originating from the seminal work of Wyner [39] on the usefulness of noise for secure communication, there has been a large body of work on basing various cryptographic primitives on different types of noisy communication channels. The most fundamental type of a noisy channel in information theory is the *binary symmetric channel* (BSC). A BSC with crossover probability p, where $0 < p < \frac{1}{2}$, flips each communicated bit independently with probability p.

In 1988, Crépeau and Kilian [10] showed that two parties can make use of a BSC to realize *oblivious transfer* (OT) [33, 16] with unconditional security. By OT we refer by default to $\binom{2}{1}$-*bit*-OT, a protocol which allows a receiver to select exactly one of two bits held by a sender without revealing the identity of the received bit to the sender. We require by default that security hold even against *malicious* parties. It is known that OT on a pair of m-bit strings reduces to $O(m)$ instances of bit-OT [4]. Much more broadly, OT can be used as a basis for general secure two-party computation [40, 20, 27, 25]. This settles the main *feasibility* question concerning the cryptographic power of a BSC.

In contrast to the basic feasibility question, the corresponding *efficiency* questions are far less understood. To explain the main relevant issues, it is instructive to draw an analogy with classical information theory. A naive approach to send n bits of information over a noisy channel is to do it bit-wise, by repeating every bit k times. A major breakthrough in information theory was the seminal result of Shannon [34] that by sending bits in blocks and by using the right encoding, one can achieve a *constant transmission rate,* namely use only a constant number of channel transmissions per information bit with error that vanishes with n. One can analogously define the notion of a *constant-rate protocol* for OT from BSC (or a *constant-rate reduction* of OT to BSC) as a protocol which realizes n independent instances of OT with negligible (in n) statistical error[1] by exchanging $O(n)$ bits over the channel.[2] In such a protocol, the amortized communication complexity for each instance of OT tends to a constant which is independent of the desired level of security.

The existence of a constant-rate protocol for OT from BSC has been a longstanding open question. The original protocol from [10] required $O(k^{11})$ bits of communication over a BSC to realize each instance of OT with error 2^{-k}. This communication overhead was subsequently improved by Crépeau [9] to $O(k^3)$. A major progress was made by Harnik et al. [21], who showed that constant rate can be achieved in the *semi-honest* model, in which parties do not deviate from the protocol except for trying to infer additional information from their view.

Constant-rate protocols for *string OT*, realizing a single selection between two n-bit strings by communicating $O(n)$ bits over the channel, are considerably easier to obtain. (Indeed, known reductions [4] can be used to get constant-rate string-OT from constant-rate bit-OT, but not the other way around.) Constant-rate string-OT protocols from an *erasure* channel, which erases every bit with probability $0 < p < 1$ and informs the receiver of the erasures, were presented in [31, 23].

To summarize the prior state of the art, constant-rate protocols for bit-OT from BSC were only known

[1] By the *error* of OT or other secure computation protocol we refer to the statistical simulation error under standard simulation-based definitions [5, 6, 19].

[2] This is the best one can hope for up to the exact constant. Indeed, it is known that $\Omega(n)$ bits over the BSC are necessary even if one additionally allows unlimited communication over a noiseless channel [36].

in the semi-honest model, and constant-rate string-OT protocols could only be based on an erasure channel. The existence of constant-rate bit-OT protocols from a BSC (or even from an erasure channel) as well as the existence of constant-rate string-OT protocols from a BSC were left open.

1.1 Our Results

We settle the above questions in the affirmative by presenting a statistically secure protocol which realizes n independent instances of OT, with 2^{-k} error, in which the parties communicate only $O(n) + \mathit{poly}(k)$ bits over a BSC.[3] This should be compared to the $n \cdot \mathit{poly}(k)$ bits required by previous constructions.

Combining the above main result with known results for secure two-party computation based on OT [25] we get the following corollaries:

- Any boolean circuit of size s can be securely evaluated by two parties with $O(s) + \mathit{poly}(k)$ bits of communication over a BSC, improving over the $O(s) \cdot \mathit{poly}(k)$ complexity of previous constructions.
- Applying the previous corollary, any discrete memoryless channel (described by rational crossover probabilities) can be faithfully emulated by a BSC at a constant rate.

Our techniques can be used to get similar results based on any "non-trivial" channel rather than just a BSC. We defer this generalization to the full version of this paper.

1.2 Overview of Techniques

Our construction uses a novel combination of previous results on OT from BSC [10, 21], recent techniques from the area of secure computation [7, 25], and some new tools that may be of independent interest.

Among the new general-interest tools is a so-called "Statistical-to-Perfect Lemma," showing roughly the following. Given a 2-party functionality $\mathcal{F}_f$ for securely evaluating a function f and $0 \leq \delta \leq 1$, we define $\widetilde{\mathcal{F}}_f^{(\delta)}$ to be an "δ-faulty" version of $\mathcal{F}_f$ that with probability δ allows the adversary to learn the inputs and have full control over the outputs but otherwise behaves normally. The lemma says that *any* ϵ-secure protocol for $\mathcal{F}$ in a $\mathcal{G}$-hybrid model (i.e., using oracle access to $\mathcal{G}$) *perfectly* realizes the functionality $\widetilde{\mathcal{F}}_f^{(\delta)}$ in the $\mathcal{G}$-hybrid model, where δ tends to 0 with ϵ (but inherently grows with the size of the input domain). The above lemma allows one to take an *arbitrary* (and possibly inefficient) protocol for OT from a noisy channel, such as the one from [10], and use it with a sufficiently large security parameter to get a perfectly secure implementation of $\widetilde{\mathcal{F}}_{\mathsf{OT}}^{(\delta)}$, for an arbitrarily small constant $\delta > 0$, while communicating just a constant number of bits (depending on δ) over the channel.

This calls for the use of *OT combiners* [22, 21, 32], which combine n OT implementation candidates of which some small fraction may be faulty into $m < n$ good instances of OT. A similar high level approach was used in [21] to solve our main question in the semi-honest model. While in the semi-honest model there are constant-rate combiners (tolerating a constant fraction of faulty candidates with $m = \Omega(n)$) that make only a single use of each OT candidate [21], known constant-rate OT combiners in the malicious model require a large number of calls to each candidate, making them insufficient for our purposes. Instead, we take the following alternative approach.

1. We give a direct construction of a constant-rate protocol for *string-OT* from a BSC. (As discussed above, such a result was only known for the easier cases of an erasure channel or in the semi-honest

[3]The protocol also involves a similar amount of communication over a noiseless channel. This additional communication can be implemented using the BSC with a constant rate.

model.) The protocol employs previous protocols for OT from BSC [10], the completeness of OT for secure two-party computation [27, 25], techniques from secure multiparty computation (including the use of algebraic-geometric multiplicative secret sharing [7, 26]), and privacy amplification techniques [3, 2]. Its analysis relies on the Statistical-to-Perfect lemma discussed above.

2. We extend the IPS protocol compiler [25] to apply also when the so-called "inner protocol" can employ a BSC channel. The main difficulty is that even when being forced to reveal their secrets, parties can use the uncertainty of the channel to lie without taking the risk of being caught. We address this difficulty in a natural way by employing statistical tests to ensure that *significant* deviations are being caught with high probability. The extended protocol compiler requires the inner protocol to satisfy an intermediate notion of security, referred to as "error-tolerance," that is stronger than security in the semi-honest model and weaker than security in the malicious model.

3. We instantiate the ingredients required by the extended compiler from Step 2 as follows. The so-called "watchlists" are implemented using string-OTs obtained via the protocol described in Step 1 above. The outer protocol is an efficient honest-majority MPC protocol for n instances of OT (see [24], building on [13, 7]). The error-tolerant inner protocol is based on an error-tolerant constant-rate OT combiner from [21].

1.3 Related Work

There is a very large body of related work on cryptography from noisy channels that was not accounted for in the above survey, and even here we can only give a very partial account. For the question of basing other cryptographic primitives (such as key agreement and commitment) on noisy channels see [2, 3, 29, 14, 37, 38] and references therein. The question of characterizing the types of channels on which OT can be based was studied in [28, 11, 14, 12, 38]. A general approach for converting feasibility results for OT from noisy channels into constant-rate protocols in the *semi-honest* model was given in [26]. Our work introduces a similar conversion technique that can be applied in the malicious model.

2 Preliminaries

Some of our results and analysis (in particular Theorem 1) apply to general 2-party secure function evaluation (SFE) functionalities. Such a functionality is characterized by a pair of functions $f = (f_A, f_B)$, $f_A : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}_A$ and $f_B : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}_B$ for (often finite) domains $\mathcal{X}$, $\mathcal{Y}$ and ranges $\mathcal{Z}_A, \mathcal{Z}_B$. We will refer to such an f as a 2-pary function. We associate a functionality $\mathcal{F}_f$ with a 2-party function f, which behaves as follows: $\mathcal{F}_f$ waits for inputs from both parties, and computes the respective outputs. Then if either party is corrupted, it sends the corresponding output to that party. Then it waits for an instruction from the adversary to release the output(s) to the uncorrupted party (or parties). We shall refer to such a functionality $\mathcal{F}_f$ as a 2-party SFE functionality.

The two main functionalities in this work are $\mathcal{F}_{\mathsf{BSC}}$ and $\mathcal{F}_{\mathsf{OT}}$. The $\mathcal{F}_{\mathsf{BSC}}$ functionality (BSC stands for Binary Symmetric Channel) takes as input a bit x from one of the parties (Alice), and outputs a single bit z to the other party (Bob) such that $\Pr[x \neq z] = p$ for some fixed constant probability strictly less than half. (Note that this is a randomized functionality.) $\mathcal{F}_{\mathsf{OT}}$ is an SFE functionality, associated with a function defined by $f_B(x_0, x_1; b) = x_b$ where x_0, x_1, b are single bits each; for $\mathcal{F}_{\mathsf{OT}}$ f_A is a constant function. The functionality $\mathcal{F}_{\mathsf{string\text{-}OT}}$ is similar to $\mathcal{F}_{\mathsf{OT}}$, but the inputs from Alice x_0, x_1 are longer strings.

For every 2-party SFE functionality $\mathcal{F}_f$, we define a weakened variant $\widetilde{\mathcal{F}}_f^{(p)}$ where $0 \leq p \leq 1$ is a constant error probability in the following sense. When invoked, an instance of $\widetilde{\mathcal{F}}_f^{(p)}$ would first generate a random bit which is 1 with probability p. Note that the bit is sampled before receiving inputs from any party. If the bit is 0, then the functionality behaves exactly as $\mathcal{F}_f$. Otherwise, if the bit is 1, then the functionality yields itself to adversarial control: i.e., the input(s) it receives are passed on to the adversary, and the adversary specifies the outputs to be sent (and when they should be sent) to the honest party (parties). In this case, even if neither party interacting with the functionality is corrupt, the functionality will allow the adversary to control it.

The main security definition we use is of *statistical* Universally Composable (UC) security [6]. The level of security – called statistical error – is indicated by the maximum distinguishing advantage between the real execution of the protocol and a simulated execution involving the ideal functionality that any environment can get (the distinguishing advantage being the difference in probabilities of the environment outputting 1 when interacting with the two systems). We require that the statistical error goes down as $2^{-\Omega(k)}$, where k is the security parameter. The computational complexity of the protocols should be polynomial in k and the input size. For intermediate constructions (and in Theorem 1) we consider *perfect* security as well.

We say that a protocol Π is in the $\mathcal{G}$-hybrid model if the parties can initiate and interact with (any number of) instances of the ideal functionality $\mathcal{G}$. Our goal is to give a "constant-rate" protocol for $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model. A protocol Π in the $\mathcal{G}$-hybrid model is said to be a constant-rate protocol for a functionality $\mathcal{F}$, if the total communication in Π (including communication with instances of $\mathcal{G}$) is $O(\ell) + \mathrm{poly}(k)$, where ℓ is the total communication with $\mathcal{F}$. We will be interested in realizing *parallel* instances of a target functionality, given the number of instances as a parameter (during run-time). More formally we can define a functionality $\mathcal{F}^*$ which takes ℓ as an initial input from one of the parties, and then implements ℓ parallel copies of $\mathcal{F}$. Note that when $\mathcal{F}$ and $\mathcal{G}$ are finite functionalities (i.e., with the total communication with a single instance upperbounded by a constant, as is the case for $\mathcal{F}_{\mathsf{OT}}$ and $\mathcal{F}_{\mathsf{BSC}}$), to securely realize $\mathcal{F}^*$, a constant-rate protocol Π will instantiate only $O(\ell) + \mathrm{poly}(k)$ instances of $\mathcal{G}$. (For simplicity, we shall refer to Π as a protocol for $\mathcal{F}$, rather than $\mathcal{F}^*$.)

An arithmetic encoding scheme. Our protocol (particularly, the sub-protocol in Section 4.1) relies on an efficient secret-sharing scheme that supports entrywise addition and multiplication of shared vectors. Following the terminology of [8], we refer to such a scheme as an *arithmetic encoding scheme*. Our abstraction captures the useful features of algebraic-geometric secret-sharing, introduced in [7] (see [26, 8] for related abstractions).

Our notion of arithmetic encoding is parameterized by a tuple $(\mathbb{F}, \rho, \delta, \delta')$ and is defined by three efficient algorithms (Encode, Encode', Decode'). Here $\mathbb{F}$ is a constant-size finite field, ρ, δ, δ' are positive constants less than 1, and the three algorithms satisfy the following properties.

- Encode and Encode' define constant-rate, *probabilistic* encodings of vectors over $\mathbb{F}$. More precisely, for every integer $m > 0$, there is an n, with $m > \rho n$, such that Encode and Encode' probabilistically map vectors in $\mathbb{F}^m$ to $\mathbb{F}^n$. Further, Encode and Encode' are linear: i.e., each entry of $\mathsf{Encode}(x)$ (respectively, $\mathsf{Encode}'(x)$) is a linear function of the entries of x and a set of independent random elements.

- The joint distribution of any $\lfloor \delta n \rfloor$ entries of the output of $\mathsf{Encode}(x)$ is independent of the input x.

- Decode' is an efficient δ-error-correcting decoder for Encode'. More precisely, we require that if y has Hamming distance at most δn from a vector in the support of $\mathsf{Encode}'(x)$, then $\mathsf{Decode}'(y) = x$.

- We require the following "homomorphic" properties. For any X, Y, X', Y' in the support of $\mathsf{Encode}(x)$, $\mathsf{Encode}(y)$, $\mathsf{Encode}'(x')$, $\mathsf{Encode}'(y')$, respectively:
  - $X * Y$ is in the support of $\mathsf{Encode}'(x * y)$
  - $X' + Y'$ is in the support of $\mathsf{Encode}'(x' + y')$

  where $*$ and $+$ denote entrywise multiplication and addition over $\mathbb{F}$ respectively.

- We require Encode' to be sufficiently randomizing. Note that $\mathsf{Encode}'(x)$ is uniform over an affine subspace of $\mathbb{F}^n$ whose dimension is at most $n - m$. We require that this dimension be at least $n - m(1 + \delta')$.

An arithmetic encoding scheme with the above properties can be obtained from the classes of algebraic geometric codes used in [7]. See Appendix A for details.

3 Statistical Security to Perfect Security

A crucial ingredient in our constructions and analysis is the ability to consider a weakly secure protocol to be a perfectly secure protocol for a weaker variant of the functionality. More precisely, we show the following.

Theorem 1 *Let $f : \mathcal{X} \times Y \rightarrow \mathcal{Z}_A \times \mathcal{Z}_B$ be a 2-party function, and $\mathcal{F}_f$ the secure function evaluation functionality for f. Suppose $\mathcal{G}$ is a 2-party functionality and Π is a D-round protocol such that Π UC securely realizes $\mathcal{F}_f$ in the $\mathcal{G}$-hybrid model, with a statistical security error of ϵ. Then Π UC securely realizes $\widetilde{\mathcal{F}}_f^{(p)}$ in the $\mathcal{G}$-hybrid model with perfect security, where $p = D|\mathcal{X}||\mathcal{Y}|\epsilon$.*

Above, if Π is only standalone-secure for $\mathcal{F}_f$, then the same conclusion holds for standalone security of Π for $\widetilde{\mathcal{F}}_f^{(p)}$.

This result gives a powerful composition theorem when multiple instances of the protocol Π are used together. Note that by UC security, it is indeed the case that if k copies of Π are run, one could instead consider k copies of $\mathcal{F}$, with a statistical security error bounded by $k\epsilon$. However, if ϵ is not negligible, say $\epsilon > 1/k$, then this bound gives us no useful security guarantee. What the above result does is to give a strong security guarantee for the case when ϵ is non-negligible, or even when it is a constant. It says that when k copies of Π are run, it roughly yields $(1 - p)k$ copies of $\mathcal{F}$ (mixed with about pk copies under adversarial control). In fact, it is further guaranteed that which copies will be corrupted is not under adversarial control.

In Appendix B.1 we show that it is unavoidable that p is bigger than ϵ by a factor that grows linearly with the domain size of the function.

We give a high-level idea of how we prove the above theorem. Given the systems corresponding to REAL and IDEAL executions, the overall approach is to decompose each of the REAL and IDEAL systems into two parts – $\text{REAL}_0, \text{REAL}_1$ and $\text{IDEAL}_0, \text{IDEAL}_1$ – so that REAL_0 and IDEAL_0 are identical and carry much of the "mass" of the systems; then we construct a new ideal system by combining IDEAL_0 and REAL_1, to get a system that is identical to the REAL system. Here, a combination of two systems means that with a *fixed* probability one of the two systems is chosen (corresponding to whether $\widetilde{\mathcal{F}}_f^{(p)}$ lets the adversary control it or not, corresponding to choosing IDEAL_1 and IDEAL_0 respectively): in particular, the simulator in the new system is not allowed to influence this choice. Further – and this is the main difficulty in the proof – we need to ensure that IDEAL_0 can be implemented by a simulator interacting with $\mathcal{F}_f$ (without access to an honest party's inputs or outputs); to implement REAL_1 the simulator may control the functionality as well.

Note that Theorem 1 is related to Lemma 5 in [30]. The main difference are the abovementioned restrictions on the system IDEAL_0 which require extra care in our proof.

Splitting the systems in this manner needs to be carefully defined. We carry this out in Appendix B. Here we illustrate this by a toy example, to give a sense of how the simulator for perfect security is derived from the simulator for statistical security. The protocol we consider is for a degenerate 2-party function f which provides a constant output to both parties. Further, it takes a fixed input from Alice ($|\mathcal{X}| = 1$) and takes a bit from Bob ($\mathcal{Y} = \{0, 1\}$). The protocol Π for our example consists of a single message z from Bob to Alice, which is equal to y with probability $\frac{1}{2}$ and $\perp$ otherwise. We shall consider the case when Alice is corrupt and Bob is honest. Further we need to consider only a "dummy adversary" who simply allows the environment to play the role of Alice in the (real) protocol. The simulator simulates a message from Bob, which is equal to $\perp$ with probability $\frac{1}{2}$, and a uniformly chosen bit otherwise. It is easy to see that this simulation is good up to a statistical distance of $\frac{1}{4}$.

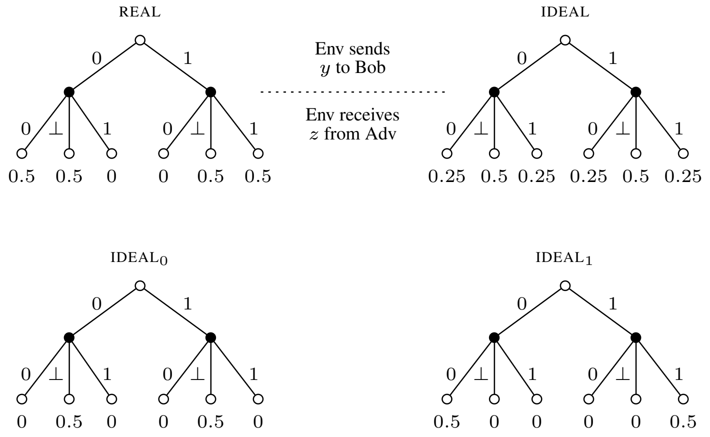

Figure 1: An example to illustrate Theorem 1. The protocol used in the example (in which Bob sends a single message to Alice — please see text) is depicted as the interaction of a system REAL with the environment. The original simulated system is IDEAL. The modified simulation (for a functionality that yields to the adversary with probability 0.5) is obtained as the combination $\text{IDEAL}_0 + \text{IDEAL}_1$ which is exactly the same as the REAL system.

In Figure 3 we illustrate this example using what we call *interaction trees*, which capture an execution involving a system (REAL or IDEAL) and an environment. The edges from the top node in these trees correspond to the two possible inputs that the environment can give to Bob ($y = 0$ and $y = 1$). The edges out of the black nodes correspond to corrupt Alice reporting the (only) message it receives from Bob in the protocol: this can be one of 0, 1, or $\perp$. The leaves correspond to complete transcripts. The probabilities of the system reaching a leaf, provided the environment sends the messages (in our case, just y) that lead to that leaf is considered the "weight" assigned to that leaf by the system.

The top-left figure corresponds to the real execution of the protocol, and the top-right corresponds to the ideal execution. Note that in the simulation, the behavior of the simulator is independent of the input y. Then we obtain a "partial system" (with total weight only 0.5, for each value of y), IDEAL_0 by comparing the REAL and IDEAL systems. In this example, IDEAL_0 is obtained by retaining in each leaf the minimum of the weights assigned by the two systems, on that leaf, but for any choice of y. We will use the ideal functionality $\widetilde{\mathcal{F}}_f^{(p)}$, with $p = \frac{1}{2}$, since that is the weight not retained by IDEAL_0.

IDEAL_1 is obtained by "subtracting" IDEAL_0 from REAL, so that the combination of IDEAL_0 and IDEAL_1

is indeed REAL. In doing this we needed to ensure that weights induced by IDEAL_0 are no more than what REAL assigns (so that the system $\text{REAL} - \text{IDEAL}_0$ does not have negative weights). Also we needed to ensure that IDEAL_0 can be implemented by a simulator which does not have access to y. Note that to implement IDEAL_1, the simulator will need to know y.

In going from this toy example to the general case poses several issues. Here the simulator for IDEAL_0 was determined without considering the interaction between the simulator and the functionality. (Indeed, there was little interaction between the two.) In general we cannot afford to do this. To properly take into account how the simulator's behavior depends on what it learns from the functionality, we consider a separate interaction which the simulator is the system and it interacts with an "enhanced environment" consisting of the original environment and the functionality. But the original statistical security guarantee is only against normal environments (and indeed, does not make sense against enhanced environments, since in the real execution there is no ideal functionality present). This requires us to relate the behavior of the enhanced environment to the behavior of the environment in the ideal world.

The final proof uses several carefully defined quantities for the three systems (the real and ideal executions, and the simulator system), and shows how one can define IDEAL_0 which can be implemented without using y, ensures that it can be extended to a perfect simulation (i.e., that the remainder of the simulation is a non-negative system), while retaining as much weight as possible (to keep p low as promised in Theorem 1).

4 A Constant-Rate OT Protocol

In this section we present our constant-rate protocol for $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model. The construction follows the paradigm of the IPS compiler [25] of combining an outer protocol secure in the honest-majority setting, with an inner protocol secure in the passive corruption (semi-honest) setting, using "watchlists" implemented using string-OTs. For this we need to instantiate these components in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model, and also extend the IPS compiler so that it admits an inner protocol in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model. We outline these steps below, and present the details in the subsequent sections.

- In order to construct the inner protocol, we will need a constant-rate OT protocol using $\mathcal{F}_{\mathsf{BSC}}$, that is secure against adaptive *passive* corruption. However, since monitoring the use of a $\mathcal{F}_{\mathsf{BSC}}$ functionality (which inherently allows errors) is harder than monitoring the use of the $\mathcal{F}_{\mathsf{OT}}$ functionality we will need a somewhat stronger security guarantee from this protocol (namely, passive security should hold even when a small constant fraction of the $\mathcal{F}_{\mathsf{BSC}}$ instances can be corrupted). We shall formalize this notion of "error-tolerance" and observe that a protocol in [21] already has the requisite properties (Lemma 2).

- The next step is to construct a constant-rate *string-OT* protocol from $\mathcal{F}_{\mathsf{BSC}}$, with security against active corruption (Lemma 1). The protocol implements a single instance of string-OT (i.e., takes only one choice bit as input from the receiver), and the rate refers to the ratio of the length of the string to the number of instances of $\mathcal{F}_{\mathsf{BSC}}$ used. This crucially relies on Theorem 1 which states that a weakly secure protocol for a functionality is a perfectly secure protocol for a weak version of the same functionality.

- The final step involves an extension of the IPS compiler [25] wherein the "inner-protocol" is in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model (rather than in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model) and enjoys error-tolerance (Lemma 3).

The extended IPS compiler from above is used to combine an appropriate constant-rate[4] outer protocol for $\mathcal{F}_{\mathsf{OT}}$ (based on [13, 7], as used in [25]) with an error-tolerant inner protocol obtained from the first step, using watchlists implemented using string-OTs from the second step.

To implement n instances of $\mathcal{F}_{\mathsf{OT}}$, the resulting compiled protocol will invoke the string-OT protocols $O(k)$ times with $O(n/k)$ long strings. Since these string-OTs are implemented using the constant-rate protocol from the second step, the compiled protocol uses a total of $O(n)$ instances of $\mathcal{F}_{\mathsf{BSC}}$ for the watchlists.

Similarly the compiled protocol invokes k instances of the inner-protocol (for a functionality defined by the outer protocol). Originally, each instance of this inner-protocol can be implemented using $O(n/k)$ instances of $\mathcal{F}_{\mathsf{OT}}$, and is passive-secure in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model. We shall replace the $\mathcal{F}_{\mathsf{OT}}$ instances used by the inner protocol with the constant-rate error-tolerant protocol from the first step. This results in an error-tolerant inner protocol in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model (for the same functionality as the original inner-protocol), which uses $O(n/k)$ instances of $\mathcal{F}_{\mathsf{BSC}}$. Thus overall, for the inner-protocol instances too, the compiled protocol uses $O(n)$ instances of $\mathcal{F}_{\mathsf{BSC}}$.

In the following sub-sections we describe how the above three steps are carried out, and what precise security guarantees they provide. Then, by following the above sketched construction we obtain our main result.

Theorem 2 *There exists a UC-secure constant-rate protocol for $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model. That is, there is a protocol that securely realizes n parallel, independent instances of $\mathcal{F}_{\mathsf{OT}}$ with statistical error 2^{-k}, with $O(n) + poly(k)$ bits of communication (including communication over $\mathcal{F}_{\mathsf{BSC}}$).*

An important corollary of implementing oblivious transfer is that it can be used to implement arbitrary function evaluation, quite efficiently [25]. Thus combined with the main result of [25] we have the following.

Corollary 3 *For any two party function f that can be computed using a boolean circuit of size s, there is a UC-secure protocol for $\mathcal{F}_f$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model, with $O(s) + \mathrm{poly}(k)$ bits of communication.*

4.1 A Constant-Rate String-OT Protocol

We denote by $\mathcal{F}_{\mathsf{string\text{-}OT}[\ell]}$ a string-OT functionality for which the sender's inputs are two strings from $\{0,1\}^\ell$. In this section we prove the following.

Lemma 1 *There exists a protocol which securely realizes a single instance of $\mathcal{F}_{\mathsf{string\text{-}OT}[\ell]}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model, with total communication of $O(\ell) + \mathrm{poly}(k)$ bits.*

This constant-rate protocol for $\mathcal{F}_{\mathsf{string\text{-}OT}}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model is constructed in three steps. The construction relies on an intermediate functionality, namely $\mathcal{F}_{\mathsf{OLE}}$ (or more precisely, $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$). The $\mathcal{F}_{\mathsf{OLE}}$ functionality (OLE stands for Oblivious Linear function Evaluation) over the field $\mathbb{F}$ evaluates the following function: it takes $p, r \in \mathbb{F}$ from Alice and $q \in \mathbb{F}$ from Bob and outputs $pq + r$ to Bob (and sends an empty output to Alice). $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$ is the error-prone version of $\mathcal{F}_{\mathsf{OLE}}$ as defined in Section 2. For simplicity we omit here the error parameter, which will be chosen as a sufficiently small constant.

Our protocol for $\mathcal{F}_{\mathsf{string\text{-}OT}}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model is constructed by composing the following protocols:

1. $\mathcal{F}_{\mathsf{string\text{-}OT}}$ protocol in the $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$-hybrid model, using a constant-rate protocol that relies on a constant-rate arithmetic encoding scheme as defined in Section 2.

[4]Here the constant-rate refers to the total communication in the protocol, and the total computation of all the servers per instance of $\mathcal{F}_{\mathsf{OT}}$ produced. More precisely, regarding the computational complexity of the servers, we are interested in the complexity of a passive-secure protocol for implementing the server computations, and it is only the so-called "type II" computations of the servers that contribute to this. See [25] for details.

2. $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$ protocol in the $\widetilde{\mathcal{F}}_{\mathsf{OT}}$-hybrid model, and

3. $\widetilde{\mathcal{F}}_{\mathsf{OT}}$ protocol in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model.

The second step is obtained by applying Theorem 1 to any OT-based protocol for $\mathcal{F}_{\mathsf{OLE}}$ (e.g., [27, 25]), where the latter is invoked with a sufficiently large (but *constant*) security parameter. The third step is obtained by similarly applying Theorem 1 to any protocol for $\mathcal{F}_{\mathsf{OT}}$ from $\mathcal{F}_{\mathsf{BSC}}$ (e.g., [10]). See Appendix C for further details on the last two steps. In the rest of this section we focus on the first step.

Reducing $\mathcal{F}_{\mathsf{string\text{-}OT}}$ to $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$. We describe an OT protocol for ℓ-bit strings. This construction uses an arithmetic encoding scheme as defined in Section 2, with parameters $0 < \rho, \delta, \delta' < 1$ and a constant-size field $\mathbb{F}$. We shall also use a strong randomness extractor Ext in our construction – a family of 2-universal hash functions suffices. During the analysis we shall specify the relation between the various parameters. The protocol is in the $\widetilde{\mathcal{F}}^{(\phi)}_{\mathsf{OLE}}$-hybrid where $\phi \leq \delta/2$.

To realize a string OT functionality in which the sender's input are two ℓ-bit strings, we shall employ the encoding from the arithmetic encoding family, with $m = O(\ell)$ (i.e., Encode and Encode' map strings from $\mathbb{F}^m$ to strings in $\mathbb{F}^n$). For the construction to have constant rate, we shall require that $m = O(\ell)$. We shall specify how exactly m and n are related to ℓ and other parameters later.

- Alice's input is (s_0, s_1) where $s_0, s_1 \in \{0,1\}^\ell$, and Bob's input is $b \in \{0,1\}$.
- Alice randomly draws x_0, x_1 from $\mathbb{F}^m$, and defines A and X'_0 as follows:
  - $A = \mathsf{Encode}(x_1 - x_0)$. In fact, we will simplify the protocol a bit by requiring that A is obtained by applying Encode to $x_1 - x_0$ with a fixed randomness (say the all-0 string).
  - Let $X'_0 = \mathsf{Encode}'(x_0)$. We will write $X'_0 = W + Z$ where W is obtained by applying Encode' to x_0 with a fixed randomness, and $Z = \mathsf{Encode}'(0^m)$ (randomly encoded).
- Bob lets $B = \mathsf{Encode}(b^m)$ (the bit b is identified with 0 or 1 in $\mathbb{F}$; b^m stands for a vector in $\mathbb{F}^m$ with all co-ordinates being b).
- They invoke n instances of the $\widetilde{\mathcal{F}}^{(\phi)}_{\mathsf{OLE}}$ functionality sequentially, as follows. For each $i \in [n]$, Alice inputs $(p_i, r_i) = (A^{(i)}, {X'_0}^{(i)})$ and Bob inputs $q_i = B^{(i)}$ to an instance of $\widetilde{\mathcal{F}}^{(\phi)}_{\mathsf{OLE}}$, and Bob gets the output $y_i = p_i q_i + r_i$. ($A^{(i)}$ stands for the ith coordinate of the vector A.) The vector $y \in \mathbb{F}^n$ that Bob gets from this is a (possibly) noisy version of $A * B + X'_0$, which in turn is in the support of $\mathsf{Encode}'(x_b)$. Bob sets $x_b = \mathsf{Decode}'(y)$.
- Alice picks two seeds h_0, h_1 for Ext and lets $w_0 = s_0 \oplus \mathsf{Ext}(x_0; h_0)$ and $w_1 = s_1 \oplus \mathsf{Ext}(x_1; h_1)$. (The parameters of Ext are chosen as mentioned above.) She sends (h_0, h_1, w_0, w_1) to Bob.
- Bob obtains $s_b = w_b \oplus \mathsf{Ext}(x_b; h_b)$.

It is easy to see that the above protocol has a constant rate (since $\ell = \Omega(m)$). To prove the UC security of the protocol, we need to consider the case where both parties are honest, as well as when one of the parties is corrupt. Let $d = \lfloor \delta n \rfloor$. Note that, except with negligible probability, at most $2\phi n < d$ out of n instances of $\widetilde{\mathcal{F}}^{(\phi)}_{\mathsf{OLE}}$ will let the adversary control them. In the simulation for all the cases (both parties are honest, or exactly one party is corrupt), the simulator starts off by faithfully simulating whether each instance of $\widetilde{\mathcal{F}}^{(\phi)}_{\mathsf{OLE}}$ lets the adversary control it or not. If more than d instances yield to adversarial control, the simulation aborts; as in the real execution, this happens with negligible probability. In the rest of the analysis, we condition on this not happening in the real execution as well as in the simulation.

Security when neither party is corrupt. In this case the environment sees only the input-output behavior of the protocol (real or simulated), the view of the adversary in the (d or fewer) corrupted instances of $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$ and the message (h_0, h_1, w_0, w_1) from Alice to Bob.[5] In the simulation (i.e., ideal execution of $\mathcal{F}_{\mathsf{string\text{-}OT}}$), Bob's output is always s_b when Alice's inputs are (s_0, s_1) and Bob's input is b (as the ideal $\mathcal{F}_{\mathsf{string\text{-}OT}}$ functionality receives its inputs from the honest parties). In the real execution of the protocol (conditioned on less than d instances of $\widetilde{\mathcal{F}}^{(\phi)}_{\mathsf{OLE}}$ being under adversarial control) the vector y received by Bob has a Hamming distance of less than d from a vector in the support of $\mathsf{Encode}'(x_b)$. So, by the error-correcting guarantee of Decode', Bob recovers x_b, and hence outputs s_b correctly. The view of the adversary from the at most d coordinates of A, X'_0 and B can be perfectly simulated. So can h_0 and h_1 be. Since x_0, x_1 are random (given the above simulated variables), the outputs of the extractor are negligibly far from uniformly random, and so (w_0, w_1) can be simulated (up to negligible statistical error) independent of (s_0, s_1) by picking uniformly random strings.

Security against corrupt Alice. Here the simulation proceeds in two steps. First, Alice's view is completely straight-line simulated (if well-formed messages are not received from Alice in any round, then Bob aborting the protocol can be simulated). Next Bob's view is sampled for the two cases $b = 0$ and $b = 1$, conditioned on Alice's view, from which Bob's outputs for each case, denoted s_0 and s_1, respectively, are obtained. To complete the simulation the simulator sends (s_0, s_1) as the input to the ideal $\mathcal{F}_{\mathsf{string\text{-}OT}}$ functionality. Details of the two steps follow.

Conditioned on $\widetilde{\mathcal{F}}^{(\phi)}_{\mathsf{OLE}}$ yielding to the adversary at most d times, Alice sees at most d coordinates of the encoding of B and this can be perfectly simulated since they are independent of Bob's input. So first the simulator will sample the (at most) d coordinates for B and hands them over to the Alice as the message from the instances of $\widetilde{\mathcal{F}}^{(\phi)}_{\mathsf{OLE}}$ controlled by her. Then it receives from Alice the output for Bob from these instances. Further, the simulator receives all but d coordinates of (A, X'_0) from Alice as inputs to the instances of $\widetilde{\mathcal{F}}^{(\phi)}_{\mathsf{OLE}}$ not controlled by her. In the next round, the simulator receives (h_0, h_1, w_0, w_1) from Alice. This completes the first part of the simulation.

For the next part, the simulator picks $B_0 = \mathsf{Encode}(0^m)$ and $B_1 = \mathsf{Encode}(1^m)$ randomly, conditioned to match the d coordinates of B that were already simulated. Then, it computes s_0 as what Bob would compute in the protocol if it uses $B = B_0$ and receives the messages implied by what Alice sent to the simulator in the first part. Similarly, it computes s_1 if Bob used $B = B_1$. Then the simulator will send (s_0, s_1) to the functionality.

Given our conditioning the real and simulated executions on there being no more than d instances of $\widetilde{\mathcal{F}}^{(\phi)}_{\mathsf{OLE}}$ under adversarial control, this is a perfect simulation. Thus over all, this gives a statistically good simulation.

[5]If the plain communication channels between the two parties are private, then the view of the environment does not include the message from Alice to Bob. Further, in this case one could redefine $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$ so that when both parties are corrupt, it will require the adversary to submit an algorithm for controlling the corrupt instances, rather than getting access to the inputs sent to the functionality. It can be seen that in Theorem 1 we can use such a definition of $\widetilde{\mathcal{F}}^{(p)}_{f}$. Then the view of the environment consists only of the input and output of the protocol.

Security against corrupt Bob. If Bob is corrupt, then he may not input a valid B in the range of $\mathsf{Encode}(0^m)$ or $\mathsf{Encode}(1^m)$. Nevertheless, we shall see that by using appropriate encodings and the extractor, there is a bit b such that Bob learns a negligible amount of information about s_{1-b}.

Note that in each instance of $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$ that gets corrupted Bob may learn the corresponding coordinates of both A and X'_0, and in the other instances he learns one linear equation over the corresponding coordinates of A and X'_0. Since both Encode and Encode' are linear, this means that Bob learns a system of (at most) $n+d$ linear equations over x_0, x_1 and the random elements used by Alice in forming X'_0. Alice's secrets that will be used subsequently are the two vectors x_0, x_1 which are only m coordinates long, so it is non-trivial to ensure that the information that Bob learns (which is more than n field elements, and typically $n > 2m$) does not contain both x_0 and x_1. This is ensured by the blinding vector Z used in the encoding of X'_0: intuitively, Z encodes at least $t' := n - m(1+\delta')$ random field elements which are present in all but d of the equations Bob learns, and so effectively Bob learns at most as much information about (x_0, x_1) as from $n - t' + d = m(1+\delta') + d$ linear equations. We shall require that $m(1+\delta') + d$ is significantly smaller than $2m$ to ensure that Bob does not learn both x_0 and x_1 completely.

To formalize the above intuition, we shall consider the dimension of the solution space for (the $2m$-dimensional vector) $X := (x_0, x_1)$ given the linear equations that Bob obtains. Below, we shall use dim to stand for the dimension of the solution space for the corresponding variables, conditioned on a set of linear equations. Let U denote the linear equations that Bob sees after the invocations of $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$ in the protocol (but before the last message in the protocol is sent). Also, let R denote the random vector used in encoding A; also by abuse of notation, we will denote by R a set of $\dim(R)$ linear equations that gives all of R. We are interested in $\dim(X|U)$ (for any possible value of U). Then, we have the following:

$$\dim(X|U) \geq \dim(X|U,R) = \dim(XZR|U,R).$$

The first inequality is simply because additional equations cannot increase the solution space, and the second equality is because, for each value of X, Z and R are uniquely determined given R, U. Also,

$$\dim(XZR|U,R) \geq \dim(X) + \dim(Z) - (n+d)$$

since $\dim(XZR) = \dim(X) + \dim(Z) + \dim(R)$ and U, R is in the form of at most $n + d + \dim(R)$ linear equations over $\mathbb{F}$. Also, $\dim(X) + \dim(Z) \geq 2m + n - m(1+\delta') = n + m(1-\delta')$. Let

$$\alpha := m(1-\delta') - d.$$

Then, $\dim(X|U) \geq \alpha$.

Then, for each value of U, there is at least one value $c \in \{0,1\}$ such that $\dim(x_c|U) \geq \alpha/2$. Note that such a c can be efficiently computed by solving for x_0 and x_1 from the equations defined by U.

Now we are ready to describe the simulator. The simulator starts off by perfectly simulating Alice's execution and the instances of $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$, till the end of all the n invocations of $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$. At this point the simulator computes the bit c as above (so that $\dim(x_c|U) \geq \alpha/2$). Then it sends $b = 1 - c$ to the ideal $\mathcal{F}_{\mathsf{string\text{-}OT}}$ functionality, and gets one of Alice's inputs, s_b. It picks a random string $w \leftarrow \{0,1\}^\ell$ in place of w_c and continues the simulation. That is, the simulator lets $w_c = w$ and $w_b = s_b \oplus \mathsf{Ext}(x_b; h_b)$.

Note that the environment's views in the real protocol and the simulation are identically distributed before Alice sends the last message. At this point, the distribution of x_c given all the information that the environment has till then is uniform over a subspace of $\mathbb{F}^m$ of dimension at least $\alpha/2$. After this, in the real protocol, the environment (which knows s_0, s_1) gets $L_c := (h_c, \mathsf{Ext}(x_c; h_c))$ and $L_b := (h_b, \mathsf{Ext}(x_b; h_b))$. In the simulation, $\mathsf{Ext}(x_c; h_c)$ is replaced by a random string ($w \oplus s_c$, where w is random). We would like to argue that L_c is statistically almost independent of x_c given U and L_b, so that the simulation is good. Since the output from the extractor is not necessarily linear, now on we consider the average min-entropy (i.e., the logarithm of predictability) [15] of random variables rather than the dimension of their support. Since $\dim(x_c|U) \geq \alpha/2$ we have $\tilde{H}_\infty(x_c|U) \geq \alpha/2 \cdot \log|\mathbb{F}|$, and hence $\tilde{H}_\infty(x_c|Uh_b) \geq \alpha/2 \cdot \log|\mathbb{F}|$ (h_b being independent of (x_c, U)).

Then, if $\mathsf{Ext}(x_b; h_b)$ is ℓ bits long, we have $\tilde{H}_\infty(x_c|UL_b) \geq \alpha/2 \log|\mathbb{F}| - \ell$ (see [15]). We shall let $\ell := 1/2\,(/\alpha/2 \cdot \log|\mathbb{F}| - \Theta(k))$ so that $\tilde{H}_\infty(x_c|UL_b) \geq (2\ell + \Theta(k)) - \ell = \ell + \Theta(k)$. Then, we set our extractor family to be a 2-universal hash function family mapping $\mathbb{F}^m$ to ℓ-bit strings. This ensures that $\mathsf{Ext}(x_c; h_c)$, conditioned on the environment's view U before the last message, and h_c, L_b, s_0, s_1, is close (up to a statistical distance of $2^{-\Omega(k)}$ is) to being uniformly random. Thus the simulator substituting $\mathsf{Ext}(x_c; h_c)$ by a random string does not change the environment's view but by a negligible amount.

4.2 Error-Tolerant Protocol for $\mathcal{F}_{\mathsf{OT}}$ over $\mathcal{F}_{\mathsf{BSC}}$

Error-tolerance. We say that a protocol π is an *error-tolerant* protocol for $\mathcal{F}$ in the $\mathcal{G}$-hybrid model if it is secure against adaptive passive corruption, and in addition tolerates active corruption of a small constant fraction of $\mathcal{G}$ instances that it invokes. More formally, we can define a modified functionality $\mathcal{G}'$, which behaves exactly as $\mathcal{G}$ until a new command $\mathsf{corrupt}$ is received as input from the adversary; if this command is received, then this instance of $\mathcal{G}$ will yield to adversarial control – i.e., send its current view to the adversary, forward immediately any subsequent message that it receives, and send messages to other parties in the protocol as instructed by the adversary. π is called a ϵ_0-error-tolerant protocol for $\mathcal{F}$ in the $\mathcal{G}$-hybrid model if π is a secure protocol for $\mathcal{F}$ in the $\mathcal{G}'$-hybrid model against adaptive passive corruption, against the class of adversaries who send out the $\mathsf{corrupt}$ command to at most $\epsilon_0 T$ of $\mathcal{G}'$ instances, where T is (an upperbound on) the total number of $\mathcal{G}$ instances invoked by π. We will call π simply an error-tolerant protocol if it is ϵ_0-error-tolerant for any constant $\epsilon_0 > 0$.

As described above in the inner protocol in our construction, we will require a constant-rate error-tolerant protocol for $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model.

We observe that such a protocol is implicit in a result in [21]. They present a constant-rate OT protocol in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model which is secure against adaptive passive adversaries. This construction starts with a simple passive-secure constant-rate protocol Φ for $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model, with a small but constant probability of error, and then uses a constant-rate *combiner* to reduce the error to negligible. This combiner uses each candidate $\mathcal{F}_{\mathsf{OT}}$ instance once, and (passive-securely) realizes a constant fraction of $\mathcal{F}_{\mathsf{OT}}$ instances. As mentioned in [21], the "error-tolerant" version of their combiner allows a small fraction of the candidate $\mathcal{F}_{\mathsf{OT}}$ instances to be actively and adaptively corrupted, though requires the parties themselves to follow the combiner's protocol honestly. The combiner corresponds to a constant-rate protocol for $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model with error tolerance as we have defined above. By composing this protocol with Φ, we get a constant-rate protocol for $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model, with the property that if a small constant fraction of the instances of $\mathcal{F}_{\mathsf{BSC}}$ are corrupted (resulting in the corruption of a small fraction of $\mathcal{F}_{\mathsf{OT}}$ instances used by the combiner protocol), security remains intact.

Lemma 2 *[21] There is a constant-rate, error-tolerant protocol for $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model.*

4.3 An Extension to the IPS Compiler

IPS compiler requires a semi-honest inner protocol over $\mathcal{F}_{\mathsf{OT}}$. We need to extend this compiler to work with inner protocols over $\mathcal{F}_{\mathsf{BSC}}$. The IPS compiler depends on being able to monitor the use of $\mathcal{F}_{\mathsf{OT}}$ channels with a good probability of catching errors; however, one cannot monitor the $\mathcal{F}_{\mathsf{BSC}}$ functionality at the same level. Hence we shall depend on the slightly stronger error-tolerance guarantee of the inner protocol. Here we shall limit ourselves to the 2-party setting (since we are interested in a 2-party functionality, namely $\mathcal{F}_{\mathsf{OT}}$).

Below we state the extension of the IPS compiler (with the new elements underlined). The proof sketch is given in Appendix D.

Lemma 3 *Suppose Π is a protocol among $n = \Theta(k)$ servers and 2 clients, for a 2-party functionality $\mathcal{F}$ between the clients, with UC-security against adaptive, active corruption of $\Omega(n)$ servers and adaptive, active corruption of (any number of) clients. Suppose $\rho^{\mathcal{F}_{\mathsf{BSC}}}$ is a 2-party protocol in the $\underline{\mathcal{F}_{\mathsf{BSC}}\text{-hybrid model}}$, that realizes the functionality of each server in the protocol Π, with $\underline{\text{error tolerance}}$. Then there is a 2-party protocol (compiled protocol) for the functionality $\mathcal{F}$ in the $\underline{(\mathcal{F}_{\mathsf{BSC}}, \mathcal{F}_{\mathsf{string\text{-}OT}})\text{-hybrid model}}$, with UC-security against adaptive, active adversaries. Further, if the (insecure) protocol obtained by directly implementing each server of Π using $\rho^{\mathcal{F}_{\mathsf{BSC}}}$ has constant rate, then the compiled protocol has constant rate too.*

Putting things together. The final protocol is obtained from Lemma 3 by using the following outer and inner protocols. The outer protocol is the one used in Section 5.1 of [25] (based on [13, 7]) applied to the functionality which realizes n instances of OT. The inner protocol is the standard OT-based implementation of the GMW protocol in the semi-honest OT-hybrid model [19], except that the OT instances consumed by this protocol are implemented using the error-tolerant protocol of Lemma 2. The watchlists are implemented using the protocol of Lemma 1.

References

[1] R. Ahlswede and I. Csiszar. On Oblivious Transfer Capacity. In *ISIT '07*, pages 2061-2064.

[2] C. H. Bennett, G. Brassard, C. Crépeau, and U. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41:1915-1923, 1995.

[3] C. H. Bennett, G. Brassard, and J.-M. Robert. Privacy Amplification by Public Discussion. *SIAM J. Comput.* 17(2): 210-229, 1988.

[4] G. Brassard, C. Crépeau, and J.-M. Robert. All-or-nothing disclosure of secrets. In *Crypto '86, LNCS*, vol. 263, pp. 234-238, 1987.

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

[6] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS '01*, pages 136-145.

[7] H. Chen and R. Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *CRYPTO '06*, pages 521–536.

[8] I. Cascudo, R. Cramer, and C. Xing. The Torsion-Limit for Algebraic Function Fields and Its Application to Arithmetic Secret Sharing. In *Crypto '11*.

[9] C. Crépeau. Efficient cryptographic protocols based on noisy channels. In *EUROCRYPT '97*, pages 306–317.

[10] C. Crépeau and J. Kilian. Achieving oblivious transfer using weakened security assumptions. In *FOCS '88*, pages 42–52.

[11] C. Crépeau, K. Morozov, and S. Wolf. Efficient Unconditional Oblivious Transfer from Almost Any Noisy Channel. In *SCN '04*, pages 47-59.

[12] I. Damgård, S. Fehr, K. Morozov, and L. Salvail. Unfair Noisy Channels and Oblivious Transfer. In *TCC '04*, pages 355-373.

[13] I. Damgård and Y. Ishai. Scalable Secure Multiparty Computation. In *Proc. Crypto 2006*, pages 501-520, 2006.

[14] I. Damgård, J. Kilian, and L. Salvail. On the (Im)possibility of Basing Oblivious Transfer and Bit Commitment on Weakened Security Assumptions. EUROCRYPT 1999: 56-73

[15] 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.* 38(1): 97-139, 2008.

[16] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[17] A. Garcia and H. Stichtenoth. On the asymptotic behavior of some towers of function fields over finite fields. *Journal of Number Theory*, 61(2):248-273, 1996.

[18] P. Gemmell and M. Sudan. Highly Resilient Correctors for Polynomials. *Information Processing Letters*, 43(4):169-174, 1992.

[19] O. Goldreich. **Foundations of Cryptography** - *Volume 2.* Cambridge University Press, 2004.

[20] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87*, pages 218–229.

[21] D. Harnik, Y. Ishai, E. Kushilevitz, and J. B. Nielsen. OT-Combiners via Secure Computation. In *TCC '08*, pages 393-411.

[22] D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On tolerant combiners for oblivious transfer and other primitives. In *EUROCRYPT '05*, pages 96–113.

[23] H. Imai, K. Morozov, and A. Nascimento. Efficient Oblivious Transfer Protocols Achieving a Non-Zero Rate from Any Non-Trivial Noisy Correlation. In *ICITS '07*.

[24] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In *39th STOC*, pp. 21–30, 2007.

[25] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding Cryptography on Oblivious Transfer - Efficiently. In *CRYPTO '08*, pages 572-591.

[26] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Extracting Correlations. FOCS 2009: 261-270

[27] J. Kilian. Founding cryptography on oblivious transfer. In *STOC '88*, pages 20–31.

[28] J. Kilian. More general completeness theorems for secure two-party computation. In *STOC '00*, pages 316-324.

[29] U. Maurer. Perfect Cryptographic Security from Partially Independent Channels. In *STOC '91*, pages 561-571.

[30] U. Maurer, K. Pietrzak and R. Renner. Indistinguishability Amplification. *CRYPTO '07*, pages 130-149.

[31] A. Nascimento and A. Winter. On the Oblivious Transfer Capacity of Noisy Correlations. *ISIT '06*, pages 1871-1875.

[32] B. Przydatek and J. Wullschleger. Error-Tolerant Combiners for Oblivious Primitives. In *ICALP '08*, pages 461-472.

[33] M.O. Rabin. How to exchange secrets by oblivious transfer. TR-81, Harvard, 1981.

[34] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal,* vol. 27, pp. 379-423 and 623-656, July and October, 1948.

[35] S. Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, 1983.

[36] Severin Winkler, Juerg Wullschleger. On the Efficiency of Classical and Quantum Oblivious Transfer Reductions. CRYPTO 2010: 707-723

[37] Andreas Winter, Anderson C. A. Nascimento, Hideki Imai: Commitment Capacity of Discrete Memoryless Channels. IMA Int. Conf. 2003: 35-51

[38] J. Wullschleger. Oblivious Transfer from Weak Noisy Channels. In *TCC '09*, pages 332-349.

[39] A. D. Wyner. The wire-tap channel. *Bell Cyst. Tech. J.*, vol. 54, pp. 1355-1387, 1975.

[40] A. C. Yao. How to generate and exchange secrets. In *FOCS '86*, pages 162–167.

A Arithmetic Encoding from MPC-Friendly Codes

In this section, we briefly sketch how an implementation of our notion of an arithmetic encoding scheme $(\mathsf{Encode}, \mathsf{Encode}', \mathsf{Decode}')$ (as defined in Section 2) follows from the literature.

Below we recall (verbatim) the notion of MPC-friendly codes from [26], which *almost* have all the properties we need.

Claim 1 ([26], implicit in [7]) *There exists a finite field $\mathbb{F}$ and an efficiently constructible family of linear error-correcting codes $C_K : \mathbb{F}^K \to \mathbb{F}^{N_K}$ with the following properties: (1) $N_K = O(K)$; (2) The dual distance of C_K is $\delta_K = \Omega(K)$; (3) The linear code C'_K spanned by all entrywise-products of pairs of codewords in C_K has dimension $K' = \Omega(K)$, minimal distance $\Delta_K = \Omega(K)$ and supports efficient decoding of up to $\mu_K = \Omega(K)$ errors. (The entrywise product of $(c_1, \dots, c_N)$ and $(c'_1, \dots, c'_N)$ is $(c_1c'_1, \dots, c_Nc'_N)$.)*

A family of codes C_K as above can be obtained from the construction of Garcia and Stichtenoth [17]. An efficient decoder for C'_K can be obtained via the Gemmel-Sudan generalization of the Welch-Berlekamp decoder for Reed-Solomon codes [18].

The one stronger property that we need here (beyond what was needed in [26]), in order to guarantee that Encode' generates the amount of entropy that we need, is a "near-MDS" property of the code C'_K. Specifically, it suffices to ensure that, for a small enough constant $\delta_0 > 0$, we have:

$$(N_K - K' - \Delta_K) < \delta_0 K,$$

where $K' = \dim(C'_K)$. Indeed, this follows immediately from the construction of [17], which in fact allows us to obtain $\delta_0 = o(1)$.

To define Encode and Encode' which take messages in $\mathbb{F}^m$ we consider the code C_K, where $K = O(m)$ and the constants ρ, δ, δ' can be chosen as functions of the parameters of C_K and C'_K as specified later. We will set the range of Encode and Encode' to be $\mathbb{F}^n$ where $n = N_K - m$.

- The algorithm Encode is defined as follows. For simplicity, we shall assume that the code C_K is *systematic* with the message appearing at the first K of the $n + m$ coordinates.[6] On input a message $x \in \mathbb{F}^m$ and randomness $r \in \mathbb{F}^{K-m}$, Encode outputs the last n coordinates of the codeword $C_K(x \circ r)$ (i.e., the codeword for the message obtained by concatenating x and r, with x in the systematic part of the codeword removed).

- The algorithm Encode' is defined in terms of Encode and the code C'_K as follows: on input $x \in \mathbb{F}^m$ and randomness $r' \in \mathbb{F}^{K'-m}$, Encode' outputs $\mathsf{Encode}(x; 0^{K-m}) * \mathsf{Encode}(1^m; 0^{K-m}) + Z(r')$ where $Z(r')$ stands for an element from the set $\{z | 0^m \circ z \in C'_K\}$, uniformly randomly selected using the random string r'.

- Decode' is defined naturally as follows: on input y, Decode' uses the decoding algorithm for C'_K to decode $0^m \circ y$ and output the first m coordinates of the resulting K' long message.

The parameters of the resulting arithmetic encoding scheme are as follows: we can choose $m = \lfloor 1/2 \min\{K', \mu_K, \delta_K\} \rfloor$. Then $m = \Omega(K)$ and since $n = N_K - m = O(K)$ (because $N_K = O(K)$), we have $\rho = m/n = \Omega(1)$. Let $\delta = \frac{1}{n}(\min\{\mu_K, \delta_K\} - m) = \Omega(1)$. Also let $\delta' = \lceil 1/m(n - K') \rceil$.

It is instructive to note that Reed-Solomon codes satisfy all the properties we need, except that Reed-Solomon codes would require the size of the field $\mathbb{F}$ to grow linearly with K. One could use Reed-Solomon codes in our constructions (instead of algebraic geometric codes) at the cost of a polylogarithmic deterioration of the parameters.

B Statistical-to-Perfect Lemma

In this section we prove Theorem 1. We start off by setting up several definitions that will aid us in our analysis.

Systems. We define a system in two parts.

- An *interaction tree* is a rooted tree with nodes labeled as system nodes or environment nodes and edges labeled with messages from a finite message space.

[6]If the code is not systematic we define Encode as follows: We have $m \leq \delta_K$, the dual distance of C_K (see choice of parameters below), and hence for any message in $\mathbb{F}^m$, there are codewords in C_K with the first m co-ordinates being equal to the message. This is an affine space of dimension $K - m$. $r \in \mathbb{F}^{K-m}$ is used to specify a unique element in this affine space, by means of a system of linear equations that map $\mathbb{F}^{K-m}$ to this affine space.

- A system over an interaction tree is defined by a set of probability distributions p_v, one for each system node v. The probability distribution p_v is over the edges coming out of the node v. For a system node v and its child w, let $p_v(w)$ denote the weight assigned to the edge (v, w) by the probability distribution p_v.

Note that there is no probability weight on the edges from an environment node. That is, the system nodes are probabilistic while the environment nodes are treated as non-deterministic. For ease of notation we shall define $p_v(w) = 1$ for each environment node v and its child w.

Given a system with an interaction tree $\mathcal{T}$, for each node u in $\mathcal{T}$ we define the weight of a node u, denoted by $\pi(u)$, as the probability that the interaction reaches u, conditioned on the environment's messages being consistent with u. More formally,

$$\pi(u) = \prod_{(v,w)\in\mathsf{path}(u)} p_v(w),$$

where $\mathsf{path}(u)$ stands for the edges in the directed path from the root to u.

Note that once a deterministic environment ENV is fixed (i.e., for each environment node, an outgoing edge is chosen), the probability of the interaction reaching a node u when the system interacts with that environment is either $\pi(u)$ or 0 (depending on whether the environment's messages are consistent with u or not). We shall denote the experiment where a system P (over an interaction tree $\mathcal{T}$) interacts with an environment ENV (also defined over $\mathcal{T}$) to reach a leaf w by $w \leftarrow \mathcal{T}(\text{P}, \text{ENV})$: this is a probabilistic traversal of the tree $\mathcal{T}$ starting from the root and ending at the leaf w.

In fact a system can be directly defined in terms of the weight function $\pi : \mathsf{Nodes}(\mathcal{T}) \to [0, 1]$ as long as the following conditions hold:

- $\pi(\text{root}) = 1$,
- for $u \in \mathsf{SysNodes}(\mathcal{T})$, $\pi(u) = \sum_{v\in\mathsf{children}(u)} \pi(v)$, and
- for $u \in \mathsf{EnvNodes}(\mathcal{T})$, for each $v \in \mathsf{children}(u)$, $\pi(u) = \pi(v)$.

Note that for a system node v with $\pi(v) \neq 0$, for each $w \in \mathsf{children}(v)$, we can define $p_v(w) = \pi(w)/\pi(v)$ (and can set p_v to be an arbitrary distribution if $\pi(v) = 0$). We will call a system a *partial system* if it is defined by a weight function as above, but with $0 \leq \pi(\text{root}) \leq 1$. (Alternately, a partial system is a normal system, but with the weight function modified at every node by multiplying with a fixed quantity $0 \leq p \leq 1$.)

Difference of two systems. Given two (partial) systems P and Q with the same interaction tree, we can define the partial system $\text{P} - \text{Q}$ by defining, for each node u, $\pi_{\text{P}-\text{Q}}(u) := \pi_{\text{P}}(u) - \pi_{\text{Q}}(u)$, provided it is non-negative for every node u. We observe that this condition is equivalent to $\pi_{\text{P}}(u) - \pi_{\text{Q}}(u)$ being non-negative for every leaf u. When this is the case we say that $\text{P} - \text{Q}$ is a legitimate partial system.

Distance between a pair of systems. Given two systems P and Q and a (deterministic) environment ENV that outputs a single bit $\text{ENV}(w)$ on reaching a leaf w of its interaction tree (P, Q and ENV all over the same interaction tree), let $\Delta_{\text{ENV}}(\text{P}, \text{Q}) = |\Pr[w \leftarrow \mathcal{T}(\text{P}, \text{ENV}) : \text{ENV}(w) = 1] - \Pr[w \leftarrow \mathcal{T}(\text{Q}, \text{ENV}) : \text{ENV}(w) = 1]|$. Then we define the distance between the two systems as

$$\Delta(\text{P}, \text{Q}) = \max_{\text{ENV}} \Delta_{\text{ENV}}(\text{P}, \text{Q}).$$

(Note that we could allow probabilistic environments too, but the maximum is achieved by a deterministic environment.)

We can define a weight function $\alpha_{\mathrm{P/Q}}$ for the leaves of the interaction tree of the two systems to obtain an alternate expression for $\Delta(\mathrm{P},\mathrm{Q})$.

$$\alpha_{\mathrm{P/Q}}(w) = \begin{cases} 1 & \text{if } \pi_{\mathrm{Q}}(w) = 0 \\ \frac{\min(\pi_{\mathrm{P}}(w),\pi_{\mathrm{Q}}(w))}{\pi_{\mathrm{Q}}(w)} & \text{otherwise} \end{cases}$$

Then, $\Delta(\mathrm{P},\mathrm{Q}) = \max_{\mathrm{ENV}} \mathrm{E}[w \leftarrow \mathcal{T}(\mathrm{Q},\mathrm{ENV}) : 1 - \alpha_{\mathrm{P/Q}}(w)]$.

Real and Ideal systems. We consider the real and ideal systems (denoted REAL and IDEAL) interacting with an external environment through the input/output interface, with one party being corrupt and one party being honest. This also subsumes the case when both parties are honest, because the two honest parties can be considered as one single party with inputs from $\mathcal{X} \times \mathcal{Y}$ and the adversary as having no inputs (i.e., a domain of size 1). So, w.l.o.g we shall consider the systems when Alice is corrupt, with the understanding that Bob could in fact stand for both Alice and Bob combined. (Incidentally, the quantity we will be interested in is the product of the sizes of the domains of the honest and corrupt parties, and this is $|\mathcal{X}||\mathcal{Y}|$ whether one or no party is corrupt.)

In the interaction tree for these systems, denoted by $\mathcal{T}_{\mathrm{I/O}}$, first the environment sends an input $y \in \mathcal{Y}$ to the honest party Bob, which we shall call the *leading message*. We shall write α for the function $\alpha_{\mathrm{REAL/IDEAL}}$, so that

$$\Delta(\mathrm{IDEAL},\mathrm{REAL}) = \max_{\mathrm{ENV}} \mathrm{E}[w \leftarrow \mathcal{T}_{\mathrm{I/O}}(\mathrm{IDEAL},\mathrm{ENV}) : 1 - \alpha(w)]. \tag{1}$$

Also note that for every $w \in \mathsf{Leaves}(\mathcal{T}_{\mathrm{I/O}})$, we have $\pi_{\mathrm{REAL}}(w) \geq \alpha(w)\pi_{\mathrm{IDEAL}}(w)$.

The simulator system. To prove our lemma, we need to convert the simulator that interacts with the original functionality $\mathcal{F}_f$ into a simulator that interacts with a modified functionality $\widetilde{\mathcal{F}}_f^{(p)}$ (see below). $\mathcal{F}_f$ first accepts an input $y \in \mathcal{Y}$ from the honest party, and then when the simulator sends an input $x \in \mathcal{X}$, it computes $f_A(x,y)$ and sends it to the simulator; subsequently, if and when instructed by the simulator, it releases the output $f_B(x,y)$ to the honest party as well.

For our construction and analysis, it helps to be able to consider the simulator as a system on its own, interacting with not only the original environment but also the functionality. Note that the interaction tree for this system is different from that of IDEAL in a couple of ways: the first message in which the environment sends $y \in Y$ to the system is no more present, and further there are new environment nodes corresponding to the functionality, and new edges corresponding to messages between the simulator and the functionality. We shall denote the interaction tree for this system by $\mathcal{T}_{\mathrm{SIM}}$. We shall refer to the environment for this interaction as an "enhanced environment," since it controls the functionality as well.

We define $[u]_y$ to be the view of the environment if it sends y as the leading message and the view of the enhanced environment in the interaction is u. More formally, $[u]_y$ is the unique leaf in $\mathcal{T}_{\mathrm{I/O}}$, reached by the path from the root starting with y as the leading message, with subsequent messages being those consistent with u and y. This path is obtained by omitting from $\mathsf{path}(u)$ all the messages involving the functionality, and adding the output message from Bob to the environment (equal to $f_B(x,y)$ where $x \in \mathcal{X}$ is the input the simulator sends to the functionality, and delivered at the round at which the simulator instructs the functionality to do so).

We say a leaf u in $\mathcal{T}_{\text{SIM}}$ is *f-consistent* with $y \in \mathcal{Y}$ (denoted by $u \vdash_f y$) if the functionality's responses in $\mathsf{path}(u)$ are consistent with $f_A(\cdot, y)$. We say a leaf u in $\mathcal{T}_{\text{SIM}}$ is *f-consistent* with a leaf w in $\mathcal{T}_{\text{I/O}}$ (denoted by $u \models_f w$), if the leading message of w is y such that $w = [u]_y$ and $u \vdash_f y$.

Note that a leaf u of $\mathcal{T}_{\text{SIM}}$ may not be consistent with any leaf w of $\mathcal{T}_{\text{I/O}}$, or maybe consistent with multiple leaves with different leading messages. However, u can be consistent with at most one leaf of $\mathcal{T}_{\text{I/O}}$ with a given leading message y, namely $w = [u]_y$. Also, we point out that

$$\pi_{\text{IDEAL}}(w) = \sum_{\substack{u \in \mathsf{Leaves}(\mathcal{T}_{\text{SIM}}) \\ u \models_f w}} \pi_{\text{SIM}}(u) \tag{2}$$

since both are equal to the probability of the environment's view being w, in an execution involving the simulator, the functionality $\mathcal{F}_f$ and any environment that respects w (that is, as long as the system does not deviate from w in its interaction, the environment does not either).

For each leaf u in $\mathcal{T}_{\text{SIM}}$, we define a quantity β as follows:

$$\beta(u) := \min_{\substack{w \in \mathsf{Leaves}(\mathcal{T}_{\text{I/O}}) \\ u \models_f w}} \alpha(w) = \min_{\substack{y \in \mathcal{Y} \\ u \vdash_f y}} \alpha([u]_y)$$

(where that the equality follows from the definition of $u \models_f w$).

The new simulator. Finally we are ready to define the construction of the new simulator that interacts with the functionality $\widetilde{\mathcal{F}}_f^{(p)}$. We shall also specify the probability p with which $\widetilde{\mathcal{F}}_f^{(p)}$ yields control to the adversary. We define a partial system for part of the new simulator when the functionality $\widetilde{\mathcal{F}}_f^{(p)}$ does not yield control. (The rest of the simulator, for when $\widetilde{\mathcal{F}}_f^{(p)}$ does yield control, will be defined so as to complete a perfect simulation.)

Over the nodes of $\mathcal{T}_{\text{SIM}}$, we define two functions σ and σ^* as follows:

$$\sigma(u) = \begin{cases} \pi_{\text{SIM}}(u)\beta(u) & \text{if } u \text{ is a leaf,} \\ \sum_{v \in \mathsf{children}(u)} \sigma(v) & \text{if } u \text{ is a non-leaf system node,} \\ \min_{v \in \mathsf{children}(u)} \sigma(v) & \text{if } u \text{ is a non-leaf (enhanced) environment node.} \end{cases}$$

$$\sigma^*(u) = \begin{cases} \sigma(u) & \text{if } u \text{ is the root,} \\ \sigma^*(\mathsf{parent}(u)) & \text{if } \mathsf{parent}(u) \text{ is an (enhanced) environment node,} \\ \sigma(u)\frac{\sigma^*(\mathsf{parent}(u))}{\sigma(\mathsf{parent}(u))} & \text{if } \mathsf{parent}(u) \text{ is a system node.} \end{cases}$$

It is easy to see that for every leaf (in fact, every node) u of $\mathcal{T}_{\text{SIM}}$,

$$0 \le \sigma^*(u) \le \sigma(u) \le \pi_{\text{SIM}}(u). \tag{3}$$

Let SIM_0 denote the partial system obtained by letting π_{SIM_0} to be σ^*. Let IDEAL_0 be the system obtained by composing SIM_0 with $\mathcal{F}_f$. Then, as in the case of Eq (2), we have

$$\pi_{\text{IDEAL}_0}(w) = \sum_{\substack{u \in \mathsf{Leaves}(\mathcal{T}_{\text{SIM}}) \\ u \models_f w}} \pi_{\text{SIM}_0}(u) = \sum_{\substack{u \in \mathsf{Leaves}(\mathcal{T}_{\text{SIM}}) \\ u \models_f w}} \sigma^*(u). \tag{4}$$

We shall show that the partial system $\text{IDEAL}_1 := \text{REAL} - \text{IDEAL}_0$ is legitimate. Also, we define $p = 1 - \sigma(\text{root})$. Then the new simulator works as follows: if $\widetilde{\mathcal{F}}_f^{(p)}$ does not surrender, which happens with probability equal to $1-p$ then it runs SIM_0 (scaled up to weight 1) against $\widetilde{\mathcal{F}}_f^{(p)}$; since $\widetilde{\mathcal{F}}_f^{(p)}$ behaves as $\mathcal{F}_f$ in this case, the resulting partial system is exactly IDEAL_0 (which has weight $1-p$). If $\widetilde{\mathcal{F}}_f^{(p)}$ surrenders (with probability p), then it allows the simulator to control it completely, and the simulator implements the partial system IDEAL_1 (which has weight p). Taken together, the two partial systems give a system that is identical to REAL.

Claim 2 $\text{IDEAL}_1 := \text{REAL}-\text{IDEAL}_0$ *is a legitimate partial system. That is,* $\pi_{\text{REAL}}-\pi_{\text{IDEAL}_0}$ *is a non-negative weight function.*

PROOF: As mentioned before, it is enough to prove the non-negativity condition for the leaves of the interaction tree $\mathcal{T}_{\text{I/O}}$. For any leaf w of $\mathcal{T}_{\text{I/O}}$, we have:

$$\pi_{\text{IDEAL}_0}(w) = \sum_{\substack{u\in\text{Leaves}(\mathcal{T}_{\text{SIM}})\\ u\models_f w}} \sigma^*(u) \le \sum_{\substack{u\in\text{Leaves}(\mathcal{T}_{\text{SIM}})\\ u\models_f w}} \pi_{\text{SIM}}(u)\alpha(w) = \alpha(w)\pi_{\text{IDEAL}}(w) \le \pi_{\text{REAL}}(w),$$

where the first inequality follows from the fact that $\sigma^*(u) \le \sigma(u)$, and since u is a leaf, $\sigma(u) = \pi_{\text{SIM}}(u)\beta(u) \le \pi_{\text{SIM}}(u)\alpha(w)$ for every u such that $u \models_f w$. □

Next we bound $p = 1 - \sigma(\text{root})$. Recall that an enhanced environment interacts with a simulator and controls the functionality as well. $\sigma(\text{root})$ is defined so that it is exactly the minimum expected value for β that an enhanced environment can achieve. In other words,

$$\sigma(\text{root}) = \min_{\text{ENV}^\dagger\in\mathcal{E}^\dagger} \mathrm{E}\left[u \leftarrow \mathcal{T}_{\text{SIM}}(\text{SIM}, \text{ENV}^\dagger) : \beta(u)\right].$$

Below we shall consider an enhanced environment *maximizing* $1-\beta(u)$, so that the expected maximum can be bounded by the expected sum, which will let us use linearity of expectation to analyze it.

$$1 - \beta(u) = \max_{\substack{y\in\mathcal{Y}\\ u\vdash_f y}} (1 - \alpha([u]_y)) \le \sum_{y\in\mathcal{Y}} (1 - \alpha([u]_y)) \cdot \iota[u \vdash_f y],$$

where the indicator variable $\iota[u \vdash_f y] = 1$ if $u \vdash_f y$ and 0 otherwise. Then

$$\begin{aligned} p &\le \max_{\text{ENV}^\dagger\in\mathcal{E}^\dagger} \sum_{y\in\mathcal{Y}} \mathrm{E}\left[u \leftarrow \mathcal{T}_{\text{SIM}}(\text{SIM}, \text{ENV}^\dagger) : (1 - \alpha([u]_y)) \cdot \iota[u \vdash_f y]\right] \\ &\le \sum_{y\in\mathcal{Y}} \max_{\text{ENV}^\dagger\in\mathcal{E}^\dagger} \mathrm{E}\left[u \leftarrow \mathcal{T}_{\text{SIM}}(\text{SIM}, \text{ENV}^\dagger) : (1 - \alpha([u]_y)) \cdot \iota[u \vdash_f y]\right]. \end{aligned}$$

We define an "honest enhanced environment" for a simulator to first pick a y and run the functionality faithfully, with the input from the honest party (Bob) being y. We will denote the class of honest enhanced environments that uses y as the input from Bob by $\mathcal{E}_y^\ddagger$. Then we note that for every $y \in \mathcal{Y}$,

$$\max_{\text{ENV}^\dagger\in\mathcal{E}^\dagger} \mathrm{E}\left[u \leftarrow \mathcal{T}_{\text{SIM}}(\text{SIM}, \text{ENV}^\dagger) : (1 - \alpha([u]_y)) \cdot \iota[u \vdash_f y]\right] = \max_{\text{ENV}^\ddagger\in\mathcal{E}_y^\ddagger} \mathrm{E}\left[u \leftarrow \mathcal{T}_{\text{SIM}}(\text{SIM}, \text{ENV}^\ddagger) : 1 - \alpha([u]_y)\right]$$

because $\text{ENV}^{\dagger}$ cannot increase the expected value of $(1-\alpha([u]_y)) \cdot \iota[u \vdash_f y]$ by deviating from the correct functionality. Thus we get,

$$p \leq \sum_{y \in \mathcal{Y}} \max_{\text{ENV}^{\ddagger} \in \mathcal{E}_y^{\ddagger}} \mathrm{E}\left[u \leftarrow \mathcal{T}_{\text{SIM}}(\text{SIM}, \text{ENV}^{\ddagger}) : 1 - \alpha([u]_y)\right] \tag{5}$$

To bound the right-hand side we need to relate it to the maximum expected value achieved by a normal (unenhanced) environment. This we do next.

Claim 3 *If the depth of the interaction tree $\mathcal{T}_{\text{SIM}}$ is at most D,*[7] *then for all $y \in \mathcal{Y}$,*

$$\max_{\text{ENV}^{\ddagger} \in \mathcal{E}_y^{\ddagger}} \mathrm{E}\left[u \leftarrow \mathcal{T}_{\text{SIM}}(\text{SIM}, \text{ENV}^{\ddagger}) : 1 - \alpha([u]_y)\right] \leq D|\mathcal{X}| \max_{\text{ENV} \in \mathcal{E}} \mathrm{E}\left[w \leftarrow \mathcal{T}_{\text{I/O}}(\text{IDEAL}, \text{ENV}) : 1 - \alpha(w)\right]$$

PROOF: For any $\text{ENV}^{\ddagger} \in \mathcal{E}_y^{\ddagger}$, consider an environment $\text{ENV} \in \mathcal{E}$ that starts behaving as $\text{ENV}^{\ddagger}$, by sending y to the IDEAL system. But ENV may not be able to proceed as $\text{ENV}^{\ddagger}$ does, since it does not see the messages between the simulator and the functionality that $\text{ENV}^{\ddagger}$ can base its behavior on. There are three messages between the simulator and the functionality which can happen in any round of the interaction: the simulator can send an input to the functionality, receive a response, and subsequently instruct the functionality to release the output to Bob.

For this ENV randomly guesses $i \in [D]$ and $x \in \mathcal{X}$, and carries on the execution of $\text{ENV}^{\ddagger}$ assuming that exactly at the i^{th} round, the simulator queries the functionality with input x. Under this assumption, the response from the functionality can be calculated as $f_A(x, y)$. Further, the round at which the simulator instructs the functionality to release the output can be be inferred from the round at which ENV receives an output from Bob.

Now consider execution of $\mathcal{T}_{\text{I/O}}(\text{IDEAL}, \text{ENV})$. With probability $\frac{1}{D|\mathcal{X}|}$ the guess made by ENV matches the choice made by the simulator in the IDEAL system. Conditioned on such a match, the executions $\mathcal{T}_{\text{I/O}}(\text{IDEAL}, \text{ENV})$ (augmented with the guessed messages) are identically distributed as the executions $\mathcal{T}_{\text{SIM}}(\text{SIM}, \text{ENV}^{\ddagger})$, and in particular the probability of being consistent with a leaf w of $\mathcal{T}_{\text{I/O}}$ is the same in both. Then, $\mathrm{E}\left[w \leftarrow \mathcal{T}_{\text{I/O}}(\text{IDEAL}, \text{ENV}) | \text{ Guess correct } : 1 - \alpha(w)\right] = \mathrm{E}\left[u \leftarrow \mathcal{T}_{\text{SIM}}(\text{SIM}, \text{ENV}^{\ddagger}) : 1 - \alpha([u]_y)\right]$. This, combined with the probability of the guess being correct, proves the claim. □

By Eq. (5) and the above claim we have that

$$p \leq D|\mathcal{X}||\mathcal{Y}| \max_{\text{ENV} \in \mathcal{E}} \mathrm{E}\left[w \leftarrow \mathcal{T}_{\text{I/O}}(\text{IDEAL}, \text{ENV}) : 1 - \alpha(w)\right] = D|\mathcal{X}||\mathcal{Y}|\Delta(\text{REAL}, \text{IDEAL}), \tag{6}$$

where D is upperbounded by the number of rounds in the protocol defining the interaction tree of the system REAL.

B.1 Tightness of the Result

We show that it is unavoidable that p is bigger than ϵ by a factor dependent on the domain size of the function.

Consider a simple function f with $|\mathcal{X}| = 1$ (no input for Alice) and $|\mathcal{Y}| = n$, and $|\mathcal{Z}_A| = |\mathcal{Z}_B| = 1$ (i.e., no outputs for either party). A protocol for this function in which Bob, on input y, randomly picks $y' \leftarrow \mathcal{Y} \backslash y$ and sends y' to Alice is a secure realization of $\mathcal{F}_f$ with a statistical error $\epsilon = 1/n$ (because when Alice is

[7]It is enough that any root-to-leaf path in $\mathcal{T}_{\text{SIM}}$ has at most D simulator nodes that have a functionality node as a child.

corrupt, we have a simulator that picks a uniformly random y'). However, this protocol is not a perfectly secure realization of $\widetilde{\mathcal{F}}_f^{(p)}$ for any $p < 1$: when $\widetilde{\mathcal{F}}_f^{(p)}$ does not yield to the simulator (which happens with probability $1 - p > 0$), the simulator gets no information about y, but for perfect simulation, the simulator must send a $y' \neq y$ with probability 1, which is impossible when y is chosen at random.

C Steps in Constructing a String OT protocol

In this section we sketch the last two of the three steps in the proof of Lemma 1.

C.1 Reducing $\widetilde{\mathcal{F}}_{\mathsf{OLE}}$ to $\widetilde{\mathcal{F}}_{\mathsf{OT}}$

For a given small constant ϕ_1 we shall construct a (perfectly) secure protocol for $\widetilde{\mathcal{F}}_{\mathsf{OLE}}^{(\phi_1)}$ in the $\widetilde{\mathcal{F}}_{\mathsf{OT}}^{(\phi_0)}$-hybrid model, for a sufficiently small constant ϕ_0.

The main ingredient here is actually in the analysis, namely Theorem 1, which will then let us use existing low rate (i.e., sub-constant rate) protocols in the literature, but with constant security parameters. In more detail, there exist protocols for a single instance of $\mathcal{F}_{\mathsf{OLE}}$ in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model whose communication complexity grows with the security parameter [27, 25]. (The latter work gives a constant rate protocol asymptotically, but is not needed in this step.) We shall use such a protocol, denoted by Γ, but employing a constant security parameter. Details follow.

First we choose a constant c such that (1) for some D, by using a sufficiently large security parameter γ for Γ, it has at most D rounds and securely realizes $\mathcal{F}_{\mathsf{OLE}}$ with $\frac{\phi_1}{2c}$-statistical security, and (2) we have $c \geq D|\mathcal{X}||\mathcal{Y}|$, where $\mathcal{X}, \mathcal{Y}$ are the domains of the function associated with $\mathcal{F}_{\mathsf{OLE}}$. Note that it is possible to choose such a c, when the number of rounds D is a constant independent of the level of statistical security required from (as is the case with the protocols from the above works, for finite functionalities including $\mathcal{F}_{\mathsf{OLE}}$)[8] since $\mathcal{F}_{\mathsf{OLE}}$ has a finite domain (as $\mathbb{F}$ is a fixed finite field). Next, we choose ϕ_0 so that the total communication (including calls to $\mathcal{F}_{\mathsf{OT}}$) made by Γ when run with security parameter γ is at most $\frac{\phi_1}{2c\phi_0}$. Since γ, as well as c and ϕ_1, is a constant (independent of k), so is ϕ_0. Then by a union bound, using $\widetilde{\mathcal{F}}_{\mathsf{OT}}^{(\phi_0)}$ instead of $\mathcal{F}_{\mathsf{OT}}$ in Γ introduces an additional security gap of $\frac{\phi_1}{2c}$. Then we invoke Theorem 1 to conclude that this ϕ_1/c-secure protocol for $\mathcal{F}_{\mathsf{OLE}}$ is in fact a (perfectly) secure protocol for $\widetilde{\mathcal{F}}_{\mathsf{OLE}}^{(\phi_1)}$.

C.2 Reducing $\widetilde{\mathcal{F}}_{\mathsf{OT}}$ to $\mathcal{F}_{\mathsf{BSC}}$

Finally, given a small constant ϕ_0 we shall construct a (perfectly) secure protocol for $\widetilde{\mathcal{F}}_{\mathsf{OT}}^{(\phi_0)}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model. Again, this reduction will use existing protocols but with constant security parameter, and then apply Theorem 1 to obtain a $\widetilde{\mathcal{F}}_{\mathsf{OT}}$ protocol. In particular, the main protocol for $\mathcal{F}_{\mathsf{OT}}$ in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model in [10] is (UC) secure against active corruption. As in the previous step, we can run this protocol with the security parameter set to a sufficiently large constant and obtain a $\phi_0/2$-statistically secure protocol for $\mathcal{F}_{\mathsf{OT}}$. Then by Theorem 1, this is also a perfectly secure protocol for $\widetilde{\mathcal{F}}_{\mathsf{OT}}^{(\phi_0)}$.

[8] Even if we were to use a protocol for $\mathcal{F}_{\mathsf{OLE}}$ where the number of rounds D grows as $o(1/\epsilon)$ for achieving ϵ-security, we can find a c as required. Note that typically ϵ will be required to be negligible in the number of rounds, and every polynomial in D will be $o(1/\epsilon)$.

D Extending the IPS Compiler

To explain how we obtain Lemma 3, we briefly summarize the IPS compiler and point out where we differ from it. At a high-level, the compiled protocol works as follows:

- First, each honest client randomly chooses a set of k servers to put on its watchlist (which only that client knows). Then, for each server $\overline{P}_j$ the protocol will set up a "watchlist channel" $\mathbf{W}_j$ such that a client can send a message on $\mathbf{W}_j$ and if the other client has server $\overline{P}_j$ on its watchlist, then it will receive this message. We do not modify this step (except for using a different implementation of string-OTs).

- Then the clients start simulating a session of Π. The clients, in addition to playing the role of the respective clients in Π, will also use the inner protocol to implement the servers in Π. We shall denote by $\rho_j^{\mathcal{F}_{\mathsf{BSC}}}$ the j-th session of $\rho^{\mathcal{F}_{\mathsf{BSC}}}$, implementing the (reactive) functionality of the j-th server in Π, $\overline{P}_j$.

  Note that the inner protocol is in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model. We will require that the messages sent via the $\mathcal{F}_{\mathsf{BSC}}$ functionality at any point should be fresh uniform random bits. (This will be the case in our protocol, but in general can be easily enforced by first sending a random bit over the $\mathcal{F}_{\mathsf{BSC}}$ functionality and then (in the next round) sending the intended bit masked with the random bit.)

- Each client is required to report over $\mathbf{W}_j$ its inputs to the inner protocol session $\rho_j^{\mathcal{F}_{\mathsf{BSC}}}$, as well as every message that it receives within that session from the $\mathcal{F}_{\mathsf{BSC}}$ instances. We shall also require the parties to report their random tapes in this session over $\mathbf{W}_j$ (or use a coin-flipping into the well protocol over $\mathbf{W}_j$, if security depends on the corrupt party's random tape being honestly generated). This part is essentially the same as that in the original IPS compiler (for 2 clients), except that there the messages were received from $\mathcal{F}_{\mathsf{OT}}$ instances instead of $\mathcal{F}_{\mathsf{BSC}}$ instances.

- Each client is required to carry out consistency checks on the messages it can read from the various watchlists and the messages it receives in the rest of the protocol execution and to abort the execution of the entire (compiled) protocol if any check fails. *Our consistency checks are different from those in the IPS compiler since the inner protocol is no more in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model, but in the $\mathcal{F}_{\mathsf{BSC}}$-hybrid model.* We shall describe our consistency checks below.

Consistency checks on the watchlists. A corrupt party may report a wrong value for a bit received over a $\mathcal{F}_{\mathsf{BSC}}$ instance in a session $\rho_j^{\mathcal{F}_{\mathsf{BSC}}}$. The other party (who sent the bit) cannot easily detect this deviation since $\mathcal{F}_{\mathsf{BSC}}$ could legitimately have behaved differently. Indeed, the corrupt party could report one bit wrongly in *every* session (i.e., for each j, report one out of $O(n/k)$ bits incorrectly), and the resulting distribution of reports remains statistically indistinguishable from that for honest behavior.[9]

Nevertheless, we shall carry out a simple statistical consistency check that can, with high probability, catch large deviations in any session that is being watched, and leave it to the special-security guarantee of the inner protocol to correct small deviations (and, as in the IPS compiler, leave it to the outer protocol to correct deviations in the sessions that are not being watched).

More precisely, let t be an upperbound on the number of instances of $\mathcal{F}_{\mathsf{BSC}}$ invoked by one session of the inner protocol. For convenience, let 2ϵ be the special-security factor for the inner protocol over $\mathcal{F}_{\mathsf{BSC}}$; i.e.,

[9]This is in contrast to the setting of the IPS compiler, wherein if the corrupt party reports wrongly in $\Omega(k)$ $\mathcal{F}_{\mathsf{OT}}$ instances, it will be caught deviating with overwhelming probability, because even one inconsistency between the view of the sender and the reported view of the receiver proves corruption.

the protocol can tolerate active corruption of up to $2\epsilon t$ instances of $\mathcal{F}_{\textsf{BSC}}$ per session. Then, the consistency checks by Alice are as follows.

- For each inner protocol session that is on its watchlist, in every round Alice checks that the Hamming distance between the string of bits it each actually sent so far over $\mathcal{F}_{\textsf{BSC}}$ instances in that session, and the string of bits that Bob reported over the watchlist channel as he received by him is at most, say $pt + \frac{1}{2}(1-2p)\epsilon t$, where $p < \frac{1}{2}$ is the cross-over probability of the $\mathcal{F}_{\textsf{BSC}}$ functionality.
- Also, based on the reported information, Alice computes Bob's state and checks that the messages received from Bob (over the plain communication channels) are according to this state, and that the string of bits received over the $\mathcal{F}_{\textsf{BSC}}$ functionality has a Hamming distance of at most $pt + \frac{1}{2}(1-2p)\epsilon t$ from the string that Bob should have sent over the $\mathcal{F}_{\textsf{BSC}}$ functionality according to this state.

Note that the expected Hamming distance between the sent and received strings (after all t bits are sent) is pt and by Chernoff bounds, the probability of the Hamming distance exceeding this by $\frac{1}{2}(1-2p)\epsilon t$ is negligible (in t and hence in k) if the bits sent and reported as received are correct.

On the other hand, if Bob wrongly reports more than ϵt of the received bits, then the expected Hamming distance with what Alice sent[10] is more than $p(1-\epsilon)t + (1-p)\epsilon t = pt + (1-2p)\epsilon t$, and the probability that the Hamming distance will be at most $pt + \frac{1}{2}(1-2p)\epsilon t$ is again negligible by Chernoff bounds. Thus this consistency check will, with high probability, let Alice detect Bob altering more than ϵt of the bits received from $\mathcal{F}_{\textsf{BSC}}$ in any session on the watchlist. Also, if Bob behaves inconsistently with his reported view, then Alice will detect it with overwhelming probability unless the inconsistency is limited to altering at most ϵt bits that are input to the $\mathcal{F}_{\textsf{BSC}}$. Then, we consider a simulation such that in each inner session, the simulator runs the honest inner protocol for Bob, but corrupts up to $2\epsilon t$ instances of the $\mathcal{F}_{\textsf{BSC}}$ functionality used in that session (to alter the bits received or delivered by the $\mathcal{F}_{\textsf{BSC}}$ instances from this honest Bob, so that it matches what is sent by or reported as received by the actual Bob). This simulation remains perfect until Bob sends a message on the plain channel that is inconsistent with the message that the simulated honest Bob would send, or the simulation needs to corrupt over $2\epsilon t$ instances of $\mathcal{F}_{\textsf{BSC}}$; at this point the simulation of the inner protocol will be continued by corrupting the server in the outer protocol corresponding to this session. If the session is on Alice's watchlist, the probability that the simulation will reach this point without Alice having aborted the protocol is negligible. The rest of the analysis is identical to that in the case of the IPS compiler (i.e., that the probability that the simulator will need to corrupt more than a threshold number of servers in the outer protocol is negligible, and if it corrupts fewer than that many servers, the security guarantee of the outer protocol carries over to the compiled protocol).

[10]Here we use the fact that fresh random bits were sent over the $\mathcal{F}_{\textsf{BSC}}$ functionality, and the receiver needs to report what it received before learning anything more about these bits.