Efficient Non-Interactive Secure Computation

Yuval Ishai[1]*, Eyal Kushilevitz[1]**, Rafail Ostrovsky[2]***, Manoj Prabhakaran[3]†, and Amit Sahai[2]‡

[1] Dept. of Computer Science, Technion, Haifa, Israel.
[2] University of California, Los Angeles.
[3] University of Illinois, Urbana-Champaign.

Abstract. Suppose that a receiver R wishes to publish an encryption of her secret input x so that every sender S, holding an input y, can reveal $f(x, y)$ to R by sending her a single message. This should be done while simultaneously protecting the secrecy of y against a corrupted R and preventing a corrupted S from having an unfair influence on the output of R beyond what is allowed by f.

When the parties are semi-honest, practical solutions can be based on Yao's garbled circuit technique. However, for the general problem when the parties, or even S alone, may be malicious, all known polynomial-time solutions are highly inefficient. This is due in part to the fact that known solutions make a *non-black-box* use of cryptographic primitives, e.g., for providing non-interactive zero-knowledge proofs of statements involving cryptographic computations on secrets.

Motivated by the above question, we consider the problem of secure two-party computation in a model that allows only parallel calls to an ideal oblivious transfer (OT) oracle with no additional interaction. We obtain the following results.

- **Feasibility.** We present the first general protocols in this model which only make a *black-box* use of a pseudorandom generator (PRG). All previous OT-based protocols either make a non-black-box use of cryptographic primitives or require multiple rounds of interaction.
- **Efficiency.** We also consider the question of minimizing the asymptotic number of PRG calls made by such protocols. We show that $\mathrm{polylog}(\kappa)$ calls are sufficient for each gate in a (large) boolean circuit computing f, where κ is a statistical security parameter guaranteeing at most $2^{-\kappa}$ simulation error of a malicious sender. Furthermore, the number of PRG calls per gate can be made *constant* by settling for a relaxed notion of security which allows a malicious S to arbitrarily correlate the event that R detects cheating with the input of R. This improves over the state of the art also for *interactive* constant-round black-box protocols, which required $\Omega(\kappa)$ PRG calls per gate, even with similar relaxations of the notion of security.

Combining the above results with 2-message (parallel) OT protocols in the CRS model, we get the first solutions to the initial motivating question which only make a black-box use of standard cryptographic primitives.

1 Introduction

This work is motivated by the following variant of the problem of computing on encrypted data [42, 43]. Suppose that a receiver R wishes to publish a semantically secure encryption of her secret input x so that any sender S, holding an input y, can reveal $f(x, y)$ to R by sending her a single message. (The message can be seen as an encryption of $f(x, y)$ that the receiver can decrypt.) We want this process to protect the secrecy of y against a corrupted R and, at the same time, prevent a corrupted S from having an unfair influence on the output of R beyond what is allowed by f. We refer to this flavor of computing on encrypted data as *non-interactive secure computation* (NISC).

As a concrete motivating scenario for NISC, consider a receiver Roberta who wishes to publish an encrypted version of her personal profile x on her public web page towards finding a suitable partner for dating. A solution

* 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

** Work done in part while visiting UCLA. Supported by ISF grant 1361/10 and BSF grant 2008411. eyalk@cs.technion.il

*** Research supported in part from NSF grants 0830803 and 0916574, BSF grant 2008411, and grants from Okawa Foundation, IBM, and Lockheed-Martin Corporation. rafail@cs.ucla.edu

† Supported by NSF grant CNS 07-47027. mmp@cs.uiuc.edu

‡ Research supported in part from NSF grants 0916574, 0830803, 0716389, and 0627781, BSF grant 2008411, a Google Research Award, a Xerox Foundation Grant, an equipment grant from Intel, and an Okawa Foundation Research Grant. sahai@cs.ucla.edu to our problem would allow an arbitrary sender Sam, with personal profile y, to send an email message to Roberta which reveals his verifiable contact information only if the two profiles match. (The matching criteria can either be determined by a public algorithm which is embedded into f, or alternatively specified in Roberta's secret profile.) In order to protect the secrecy of Roberta's profile x, its encryption should be semantically secure. In order to protect the secrecy of Sam's profile y, he should be ensured that no information is revealed to Roberta other than what is implied by the output of f. Finally, to help protect Roberta against eager senders who try to force a match, she should be ensured that every strategy of such a sender corresponds to some valid profile y.

Standard techniques for secure computation and computing on encrypted data perform quite well when the parties are guaranteed to be semi-honest. For instance, practical NISC protocols in this setting can be obtained by combining Yao's garbled circuit technique [44, 31] and any two-message oblivious transfer (OT) protocol [7, 14]. Low-communication (but at this point less practical) solutions can be obtained using homomorphic encryption for general functions [13] or for restricted classes of functions [30, 6, 22, 33].

For some of the above protocols, protecting S against a malicious R can come at a relatively low cost. In protocols based on Yao's construction this can be done (in the CRS model) by using efficient 2-message UC-secure OT protocols [39] (see also [11]). However, known techniques for protecting R against a malicious S either involve additional rounds of interaction [32] or are highly inefficient. For instance, this is the case if S is required to prove, using non-interactive zero-knowledge (NIZK), that he constructed a valid garbled circuit [7, 17]). Such proofs seem very costly even with the best known NIZK techniques. Moreover, even from a qualitative point of view, such NIZK-based solutions leave much to be desired in that they inherently require to make a *non-black-box* use of the underlying cryptographic primitives. For instance, while the semi-honest version of Yao's protocol can be implemented by making a black-box use of (parallel) 2-message OT and a pseudorandom generator (PRG), this is not the case for a NIZK-based variant which is secure against malicious senders.

The above state of affairs motivates the study of NISC protocols which only make a *black-box* use of standard cryptographic primitives. We further simplify the problem by allowing S and R to engage in *parallel* invocations of an ideal OT oracle, but without allowing any additional interaction.[4] We refer to such a protocol as a NISC protocol in the *OT-hybrid* model, or NISC/OT protocol for short. More formally, a NISC/OT protocol for $f(x, y)$ is a protocol which UC-securely realizes f using only parallel calls to an ideal OT oracle.

Our main motivation for considering this a good model for NISC is the aforementioned existence of efficient UC-secure 2-message implementations of OT in the CRS model. Indeed, using the UC composition theorem [8], NISC/OT protocols can be combined with such 2-message OT protocols to yield NISC protocols in the CRS model that have the same communication pattern as in the initial motivating example.

Additional advantages of OT-based protocols include the *generality* advantage of being able to realize OT in a variety of models and under a variety of standard assumptions, as well as efficiency advantages such as the possibility of precomputing the necessary OTs [4, 2] and the possibility to amortize the cost of this precomputation [3, 36, 18]. See [24] for further discussion.

We turn to the feasibility question of minimizing the assumptions on which NISC/OT protocols can be based. In the OT-hybrid model, any polynomial-time computable functionality can be efficiently realized *unconditionally* [16, 28, 24]. However, it is wide open whether the same is true for constant-round protocols. (This question is related to the corresponding question in the honest-majority MPC setting [5], which in turn is related to other wide open questions [19].) Given the lack of progress on this front, a second best alternative is to base general NISC/OT protocols on any one-way function, or equivalently a PRG. As noted above, Yao's protocol provides such a solution in the semi-honest model. Moreover, it is shown in [24] (see Appendix B of [23]) how to get a similar protocol in the malicious NISC/OT model; however, this protocol inherently makes a non-black-box use of the PRG. This motivates the following main question:

Are there NISC/OT protocols for general functions which only make a black-box use of a PRG?

A second goal of this work is to push the asymptotic efficiency limits of constant-round black-box protocols by minimizing the number of calls to the underlying cryptographic primitive. Existing constant-round black-box protocols in the OT-hybrid model (such as [34, 32] and their variants) require $\Omega(\kappa)$ calls to a PRG (or symmetric encryption)

[4] It is also useful to allow a message from the sender to the receiver which is independent of the receiver's OT choices; such a message can be realized in the pure parallel-OT hybrid model at the cost of one additional OT.

for each gate in the circuit, where κ is a statistical security parameter guaranteeing at most $2^{-\kappa}$ simulation error for a malicious sender.[5] This should be compared to the best protocols in the semi-honest model [44, 31] which require only $O(1)$ PRG calls per gate.

1.1 Our Results

We obtain the following main results.

- **Feasibility.** We present the first general NISC/OT protocols which only make a black-box use of a PRG. All previous protocols in the OT-hybrid model either make a non-black-box use of cryptographic primitives [24] or require multiple rounds of interaction (cf. [32]).
- **Efficiency.** We also consider the question of minimizing the asymptotic number of PRG calls made by such protocols. We show that $\mathrm{polylog}(\kappa)$ calls are sufficient for each gate in a (large) boolean circuit computing f, where κ is a statistical security parameter guaranteeing at most $2^{-\kappa}$ simulation error of a malicious sender.[6] Furthermore, the number of PRG calls per gate can be made *constant* by settling for a relaxed notion of security which allows a malicious S to arbitrarily correlate the event that R detects cheating with the input of R.

  This improves over the state of the art also for *interactive* constant-round black-box protocols, which required $\Omega(\kappa)$ PRG calls per gate, even with similar relaxations of the notion of security.

Combining the above results with 2-message (parallel) OT protocols in the CRS model, we get the first solutions to the initial motivating question which only make a black-box use of standard cryptographic primitives.

On re-using public keys. A standard security caveat that applies to many non-interactive protocols in the public key model (cf. [29, 27, 12, 9]) is that re-using the same receiver's public key for multiple sender messages may be problematic if the sender can learn the receiver's output on these messages. Indeed, the standard (UC-)security guarantee of our protocols only applies when an independent receiver message is used in each session. While the receiver's output does not reveal additional information about the receiver's input (other than what is allowed by f), it *may* reveal information about the secret randomness embedded in the public key, which may in turn compromise the receiver's security when leaking multiple outputs without refreshing the public key. Our protocols are indeed susceptible to this type of attacks.

We stress that re-using the same public key for multiple sender messages is always safe (in the sense of providing the natural "real-ideal" security guarantee) if the receiver refreshes the public key after revealing an output or using it in another protocol. This seems to be a very mild requirement in many practical scenarios in which sender messages are infrequent or can be aggregated before taking any action.

Similarly to [9], we can provide t-time reusable public keys (for which up to t outputs can be revealed before the key needs to be refreshed) at a much better cost than publishing t independent public keys. We note, however, that (non-black-box) NIZK-based NISC protocols are not susceptible at all to this type of attacks, and leave the possibility of obtaining a similar result using black-box constructions as an interesting open question.

On asymptotic vs. practical efficiency. As is usual in theoretical work in cryptography, we focus on optimizing asymptotic efficiency and do not try to optimize or even analyze the underlying hidden constants. Moreover, in doing so we focus on the typical case where the circuit size is much bigger than the input size which in turn is much bigger than the security parameter, and sometimes ignore low-order *additive* terms that depend on the smaller quantities. These optimization goals may conflict with practical efficiency. The question of optimizing NISC protocols towards practical implementations is left for future work.

[5] The "LEGO protocol" [38] reduces this overhead by a factor of $\log |C|$, where $|C|$ is the size of the circuit, at the expense of employing a homomorphic commitment primitive.

[6] The simulation error of the receiver is close to the distinguishing advantage of the PRG (as in Yao's original protocol) and can be made $2^{-\Omega(\kappa)}$ by choosing a PRG with similar strength.

1.2 Overview of Techniques

At a high level, our NISC/OT protocols are obtained using the following modular steps:

1. Statistically secure NISC/OT protocols for $\mathbf{NC}^0$ functions. Here we can rely on a previous protocol from [24] (see Appendix B of [23]). We also present an asymptotic efficiency improvement by applying "MPC in the head" techniques in the spirit of [20]. This is presented in Section 3.
2. Computationally secure NISC protocols for *general* functions in the $\mathbf{NC}^0$-hybrid model (allowing the parties a single call to an ideal $\mathbf{NC}^0$ functionality). Here we combine a particular implementation of Yao's garbled circuit construction with the use of unconditional one-time MACs to guarantee that a malicious sender can either deliver a correct output to the receiver or the receiver detects cheating and aborts. However, these protocols allow a malicious sender to correlate the event of the receiver aborting with the receiver's input. We present two variants of the protocol: the first (Section 5) allows arbitrary correlations with the receiver's inputs, and is the most efficient protocol we obtain in this work. The second variant (Section 6) is slightly less efficient but allows only correlations that can be expressed as disjunctions of circuit wires and their negations.
3. Finally, we present (in Section 7) an efficient general reduction of full security to security with the latter type of "correlated abort". The idea is to transform the original circuit into a new, randomized, circuit in which disjunctions of wires or their negations provide essentially no information about the input. A special case of this transformation is implicit in [25]. We reduce the general case to honest-majority MPC in the semi-honest model and instantiate it using a recent efficient protocol from [10].

We also present (in Section 4) a direct ad-hoc construction of NISC protocols in the $\mathbf{NC}^0$-hybrid model, which is asymptotically less efficient but is somewhat simpler than that obtained by combining steps 2 and 3 above.

2 Preliminaries

Below we define a non-interactive secure computation scheme (NISC). NISC may involve a trusted setup, an ideal implementation of some (non-reactive) functionality $\mathcal{H}$. We shall refer to such an NISC scheme as NISC/$\mathcal{H}$.

An NISC/$\mathcal{H}$ scheme for a function $f : X \times Y \to Z$ is a 2-party protocol between Receiver and Sender, of the following format:

- Receiver gets an input $x \in X$ and Sender gets an input $y \in Y$.
- The two parties invoke an instance of $\mathcal{H}$ with inputs of their choice, and Receiver obtains outputs from $\mathcal{H}$.
- Sender may send an additional message to Receiver.
- Receiver carries out a local computation and outputs $f(x, y)$ or an error message.

The correctness and secrecy requirements of an NISC scheme can be specified in terms of UC security. We shall denote the security parameter by κ and require that for a protocol to be considered secure, the simulation error be $2^{-\Omega(\kappa)}$. An NISC/$\mathcal{H}$ scheme for f is required to be a UC-secure realization of the following functionality $\mathcal{F}_f$ in the $\mathcal{H}$-hybrid model.

- $\mathcal{F}_f$ accepts $x \in X$ from Receiver and $y \in Y$ from Sender, and outputs $f(x, y)$ to Receiver and an empty output to Sender. If y is a special input error, then the output to Receiver is error.

In particular, we will be interested in NISC/OT schemes, where OT stands for a functionality that provides (parallel) access to multiple instances of $\binom{2}{1}$ Oblivious Transfer. In this case, the additional message from the sender to the receiver can be implemented using a single additional OT call.

We define a relaxed notion of security which is useful in the context of NISC, but may also be of broader interest.

Security with input-dependent abort. Given an SFE functionality $\mathcal{F}$, we define a functionality $\mathcal{F}^\dagger$ which behaves as follows: first $\mathcal{F}^\dagger$ accepts a description of a predicate ϕ from the adversary (e.g., in the form of a PPT algorithm); after receiving inputs from all the parties, $\mathcal{F}^\dagger$ computes the outputs to the parties as $\mathcal{F}$ does; but before delivering the outputs to the parties, $\mathcal{F}^\dagger$ runs ϕ with all the inputs; if ϕ outputs abort, then $\mathcal{F}^\dagger$ replaces the output to the honest parties by the message abort. Otherwise $\mathcal{F}^\dagger$ delivers the output from $\mathcal{F}$ to all the parties.

Though we defined security with input-dependent abort as a general security notion, we shall exclusively focus on 2-party functionalities $\mathcal{F}_f$ as defined above.

Security with wire-disjunction triggered abort. For a 2-party SFE functionality $\mathcal{F}$ as above, outputting $f(x, y)$ to only Receiver, we define a functionality $\mathcal{F}^{\ddagger}$ which is a restriction of $\mathcal{F}^{\dagger}$ in which the algorithm ϕ for determining aborting is restricted to be of the following form: ϕ includes a set W of pairs of the form (w, b), where w is a wire in a fixed circuit C for computing the function f, and $b \in \{0, 1\}$; $\phi(x, y) = \mathsf{abort}$ if and only if there exists a pair $(w, b) \in W$ such that when C is evaluated on (x, y), the wire w takes the value b. We will also consider the stronger notion of *input-disjunction triggered abort* where the disjunction can only involve input wires.

Protocol making a black-box use of a PRG. We are interested in NISC/OT schemes that do not rely on any cryptographic assumption other than the security of a PRG. Further, the scheme should be able to use any PRG provided to it, in a black-box fashion. Formally, we consider *fully black-box reductions* [41] from NISC/OT to PRG.

Towards a more concrete measure of efficiency, we require NISC/OT protocols to be $2^{-\Omega(\kappa)}$ secure and measure complexity as a function of κ and the circuit size of f. Security against corrupted senders will be statistical. To achieve the above goal against a computationally bounded corrupted receiver, we need to use a PRG for which the advantage of any PPT adversary in distinguishing the output of the PRG from a random string (of the appropriate length) is $2^{-\Omega(\kappa)}$. To this end a PRG can have a longer *computational security parameter*, k, that defines the length of its seed (k is a function of κ, but for simplicity we denote it as a separate parameter). The PRGs considered below have input and output length $\Theta(k)$.

Efficiency: Communication and Cryptographic Overheads. The best known NISC/OT scheme secure against *passive* corruption is provided by Yao's garbled circuit construction (see below) and forms the benchmark for efficiency for us. There are three aspects in which a NISC/$\mathcal{H}$ scheme can be more expensive compared to the garbled circuit (over OT) scheme:

- *The complexity of $\mathcal{H}$.* For instance, if $\mathcal{H}$ is the parallel OT functionality OT, then the number of instances of $\binom{2}{1}$ OTs and the total length of the OT strings provide natural measures of complexity of $\mathcal{H}$. (Note that a NISC/$\mathcal{H}$ scheme invokes a single instance of $\mathcal{H}$.) If $\mathcal{H}$ is a more involved functionality, we shall be interested in complexity measures related to securely realizing $\mathcal{H}$.
- *Communication overhead.* We shall include in this any communication directly between the two parties and any communication between the parties and $\mathcal{H}$. We define the *communication overhead* of a NISC scheme as the ratio of total communication in the NISC scheme and the total communication in Yao's garbled circuit (over OT) scheme.
- *Cryptographic overhead.* Yao's garbled circuit scheme makes a black-box use of PRG. To evaluate a function that is computed by a circuit C, it uses $\Theta(|C|)$ calls to a PRG (with input and output lengths $\Theta(k)$). The ratio between the number of such calls made by a NISC scheme and Yao's garbled circuit scheme can be defined as the *cryptographic overhead* of the NISC scheme.

Garbled Circuit. There are two main variants of Yao's garbled circuit construction in the literature, one following the original construction of Yao [44, 31] and one following the presentation in [37, 1]. The former allows a negligible probability of error while evaluating the circuit (since a "wrong" key may decrypt a ciphertext encrypted using different key, without being detected), whereas the latter includes pointers with the keys indicating which ciphertexts they are intended for. In this work, we follow the latter presentation (with a few simple changes). Below we describe the version of garbled circuit we shall use.

Consistent with our use of garbled circuits, we shall refer to two parties Receiver (with input x) and Sender (with input y) who wish to evaluate a function represented as a boolean circuit C, with binary gates. Given such a circuit C and input y for Sender, the garbled circuit Y_C for C is constructed as follows. For each (non-output) wire w in C, two k-bit strings K_w^0 and K_w^1 are randomly chosen, as well as a random bit r_w. The random bit r_w is used to mask the values on the wires during evaluation of the garbled circuit (if w is an output wire, or an input wire belonging to Receiver, then we set $r_w = 0$). The garbled circuit Y_C consists of the following:

- For each gate g (with input wires u, v and output wire w), for each $a, b \in \{0, 1\}$, an encryption

$$\mathrm{Encr}^{g,a,b}_{K_u^{a\oplus r_u}, K_v^{b\oplus r_v}}(K_w^{c\oplus r_w}, c)$$

where $c = \tilde{F}_g(a, b) := F_g(a \oplus r_u, b \oplus r_v) \oplus r_w$ where u, v are the input wires to gate g and w its output wire,[7] and Encr is a symmetric-key encryption scheme with a mild one-time semantic security guarantee (see below).
- For each input-wire w for which the value is already fixed (i.e., w is an input wire that belongs to Sender), the pair $(K_w^{y_w}, y_w \oplus r_w)$, where y_w is the value of that input wire.

Note that if gate g has input wires u, v and output wire w, then for any values (α, β), given (K_u^x, a) and (K_v^y, b) where $a = \alpha \oplus r_u$ and $b = \beta \oplus r_v$, one can obtain (K_w^z, c), where $z = F_g(\alpha, \beta)$ and $c = z \oplus r_w$. Hence, given Y_C and $(K_w^{x_w}, x_w)$ for each input-wire w belonging to Receiver (note that for such w, $x_w \oplus r_w = x_w$), one can evaluate $C(x, y)$ (note that an output wire w has $r_w = 0$).

The privacy guarantee of a garbled circuit is that, given x and $f(x, y)$, it is possible to construct a simulation $(\tilde{Y_C}, \tilde{K}_1, \ldots, \tilde{K}_{|x|})$ which is computationally indistinguishable (with at most a distinguishing advantage $2^{-\Omega(\kappa)}$) from $(Y_C, K_1, \ldots, K_{|x|})$ where Y_C is the garbled circuit constructed for Sender's input y, and K_i are keys of the form $K_w^{x_w}$ where w are the input wires for Receiver.

For the above security guarantee to hold, encryption Encr only needs to be *one-time* semantically secure (even if one of the two keys is revealed) [1]. But it is convenient for us to restrict ourselves to a concrete instantiation of an encryption scheme.[8] A particular instance of an encryption scheme, that satisfies the requirements for a garbled circuit, can be defined in terms of black-box access to a pseudorandom generator or, more conveniently, in terms of a pseudorandom function. For each key K in $\{0, 1\}^k$, let the pseudorandom function initialized with the key K be denoted by $R_K : [|C|] \to \{0, 1\}^{k'}$ where $[|C|]$ is the set of indices for the gates in the circuit C and k' is the maximum length of the messages to be encrypted ($k + 1$ above, but longer in the variants used in some of our schemes).[9] Then the encryption Encr is defined as follows:

$$\mathrm{Encr}^t_{K_1, K_2}(m) = m \oplus R_{K_1}(t) \oplus R_{K_2}(t), \tag{1}$$

where $t = (g, a, b)$ is a "tag" that is used once with any key.

3 A Statistical NISC/OT Protocol for NC^0

A central building block of our protocols for general functions f is a statistically secure protocol for "simple" functions g in which each output bit depends on just a small number of input bits. We say that $g(x, y)$ is d-local if each of its output bits depends on at most d input bits. Towards an asymptotic measure of efficiency, we will (implicitly) consider an infinite family of finite functionalities $g_n : \{0, 1\}^{\alpha_n} \times \{0, 1\}^{\beta_n} \to \{0, 1\}^{\gamma_n}$. We say that such a family is in **NC**^{0} if there exists an absolute constant d such that each g_n is d-local. The precise locality d of the **NC**^{0} functions we will employ is small, and will be hidden in the asymptotic analysis.

Our general protocols for evaluating a function $f(x, y)$ will typically require the evaluation of **NC**^{0} functions $g(a, b)$ where the receiver's input a is short (comparable in length to x) but the sender's input b and the output are long (comparable to the circuit size of f). We would therefore like the number of OT invocations to be comparable to the receiver's input length α, and the total communication complexity (in the OT-hybrid model) to be as close as possible to the output length γ.

The semi-honest model. In the semi-honest model, there are several known techniques for obtaining *perfectly* secure protocols that meet these requirements (cf. [21] and references therein): in such protocols the number of OTs is exactly α and the total communication complexity is $O(\gamma)$ (with a hidden multiplicative constant depending at most exponentially on d). Our goal is to get similar efficiency in the malicious model without introducing additional interaction.

[7] In fact, g may have more than one output wire; in such case, they are all assigned the same keys K_w^0 and K_w^1.

[8] The construction and analysis in [1] admits any one-time semantically secure symmetric-key encryption scheme. Our construction here is somewhat more streamlined since we use a specific encryption scheme.

[9] Note that the domain of the inputs for PRF is small, and hence a PRG with an appropriately long output can be used to directly implement the PRF. In fact, the PRF will be invoked on only as many inputs as the maximum fan-out of the circuit, and needs to be defined only for the values on which it will be invoked. Nevertheless we shall use the PRF notation for convenience and conceptual clarity.

Previous results. A statistically secure NISC/OT protocol for $\mathbf{NC}^0$ functions in the malicious model is implicit in [28]. (Via known reductions, this can be extended to functions in low complexity classes such as $\mathbf{NC}^1$ with a polynomial complexity overhead.) A more efficient protocol was given in [24] (see Appendix B of [23]). The protocol from [24] can also provide computational security for general functions, but this requires a *non-black-box* use of a pseudorandom generator. From here on we focus on the case of $\mathbf{NC}^0$ functions.

The protocol of [24] is based on a reduction to *multi-party* computation (MPC) in the *semi-honest* model, in the spirit of the MPC-based zero-knowledge protocols of [20]. Instantiated with standard MPC protocols, and settling for a relaxed notion of security, discussed in Section 3.2 below, its communication complexity is $\Theta(\gamma \cdot \kappa)$, where γ is the output length of f and κ is a statistical security parameter guaranteeing simulation error of $2^{-\kappa}$. (Here and in the following we will assume that $\gamma \gg \alpha, \kappa$ and ignore low order terms in the efficiency analysis for simplicity.)

3.1 Overview of New Protocol

We present a different approach for NISC/OT that reduces the multiplicative overhead from $\Theta(\kappa)$ to $\mathrm{polylog}(\kappa)$. Our general approach employs perfectly secure MPC protocols for the *malicious* model. The efficiency improvement will be obtained by plugging in the recent perfectly secure protocol from [10].

Given an $\mathbf{NC}^0$ function $g(a, b)$, where $g : \{0,1\}^\alpha \times \{0,1\}^\beta \to \{0,1\}^\gamma$, our construction has a similar high level structure to that of [24, 23]:

1. Start with a perfectly secure NISC/OT protocol π for g in the *semi-honest* model in which the receiver uses its original α input bits a as the sequence of OT choices. Several such protocols with a constant overhead can be found in the literature (see [21] and references therein).
2. Use the sender's algorithm in π to define a "certified OT" functionality COT, which is similar to parallel OT except that it verifies that the α pairs of strings (together with an additional witness) provided by the sender satisfy a given global consistency predicate. If this verification fails, a special error message $\perp$ is delivered to the receiver. Concretely, we will employ a COT functionality in which the sender's witness includes its randomness and its input b, and the predicate verifies that the α pairs of strings are as prescribed by the protocol. (For efficiency reasons, it may be useful to include in the witness the values of intermediate wires in the sender's computation. This standard technique can be used to transform an arbitrary predicate into one in $\mathbf{NC}^0$.)
3. Take a *perfectly secure* MPC protocol Π_{COT} for a multi-party functionality corresponding to COT, and use it to obtain a statistically secure two-party NISC/OT protocol π_{COT} for COT. This is the main novel contribution of the current section, which will be described in detail below.
4. Use π_{COT} for obtaining an NISC/OT protocol π_g for g with security in the malicious model. This can be done in a straightforward way by using COT to emulate π while ensuring that the sender does not deviate from its prescribed algorithm. Note that the protocol makes a non-black-box use of π, and thus in our black-box setting we cannot apply it to protocols π which make use of cryptographic primitives.

3.2 Relaxing Security

A (standard) technical subtlety that we need to address is that our direct implementation of π_{COT} will not realize the functionality COT under the standard notion of security, but rather under a relaxed notion of security that we refer to as security with "input-value disjunction (IVD) abort". This is similar to the notion of security with wire-value disjunction (WVD) abort from Section 6, except that here the disjunctive predicate applies only to input values. That is, the ideal functionality is augmented by allowing a malicious sender to specify a disjunctive predicate in the receiver's input bits (such as $x_2 \vee \bar{x}_4 \vee x_7$) which makes the functionality deliver $\perp$ if the receiver's input satisfies the predicate. (Otherwise the output of the original functionality is delivered.)

A standard method for upgrading security with IVD-abort into full security is by letting the receiver "secret-share" its input (cf. [28, 32]). Concretely, the receiver encodes x into a longer input x' in a way that ensures that every disjunctive predicate in x' is either satisfied with overwhelming probability, or alternatively is completely independent of x. The original functionality g is then augmented to a functionality h that first decodes the original input and then computes g. (To prevent cheating by a malicious receiver, the decoding function should output a valid input x for any string x'.)

One can now apply any protocol π_h for h which is secure with IVD-abort in order to obtain a fully secure protocol for the original functionality g. We note that the functionality h will not be in $\mathbf{NC}^0$; thus, the overhead for realizing it unconditionally (even in the semi-honest model) will be too big for our purposes. Instead, we apply the security boosting reduction only at higher level protocols which offer computational security and rely on Yao's garbled circuit construction. For such protocols, we only pay an *additive* price comparable to the circuit *size* of the decoder, which we can make linear in the input length.

We finally suggest a concrete method to encode x into x' as above. A simple method suggested in [28, 32] is to let x' be an additive sharing of x into $\kappa + 1$ shares (over $\mathcal{F}_2^\alpha$). This has the disadvantage of increasing the length of x by a factor of κ, which we would like to avoid. Better alternatives were suggested in the literature (see, e.g., [40]) but these still increase the input length by a constant factor and significantly increase the circuit size. Instead, we suggest the following encoding method. Let $G : \{0,1\}^\delta \to \{0,1\}^\alpha$ be a κ-wise independent generator. That is, for a random r, the bits of $G(r)$ are κ-wise independent. Then the encoding is defined by $Enc(x) = (r_1, \dots, r_{\kappa+1}, x \oplus G(r_1 \oplus \cdots \oplus r_{\kappa+1}))$ where the r_i are uniformly random strings of length δ. The corresponding decoder is defined by $Dec(r_1, \dots, r_{\kappa+1}, z) = z \oplus G(r_1 \oplus \cdots \oplus r_{\kappa+1})$.

The following lemma is straightforward.

Lemma 1. *For every disjunctive predicate $P(x')$, the following holds: (1) If P involves at most κ literals, then $\Pr[P(Enc(x)) = 1]$ is completely independent of x. (2) Otherwise, $\Pr[P(Enc(x)) = 1] \geq 1 - 2^{-\kappa}$.*

We note that efficient implementations of G can be based on expander graphs [35]. In particular, for any constant $0 < c < 1$ there is an $\mathbf{NC}^0$ implementation of G (with circuit size $O(\alpha)$) where $\delta = \alpha^c + poly(\kappa)$. Thus, in the typical case where $\alpha \gg \kappa$, the encoding size is $\alpha + o(\alpha)$.

The following corollary shows that, from an asymptotic point of view, boosting security with IVD-abort into full security comes essentially for free both in terms of the circuit size and the receiver's input length.

Corollary 1. *Let $f(x, y)$ be a functionality with circuit size s and receiver input size $\alpha = |x|$. Then, there exists a functionality $h(x', y)$ and a linear-time computable encoding function Enc such that:*

- *A fully secure protocol π_f for f can be obtained from any protocol π_h for h which is secure with IVD-abort by letting the parties in π_f run π_h with inputs $x' = Enc(x)$ and y.*
- *The circuit size of h is $s + O(\alpha) + \mathrm{poly}(\kappa)$.*
- *The receiver's input length in h is $\alpha + o(\alpha) + \mathrm{poly}(\kappa)$.*

3.3 Realizing COT via Robust MPC

It remains to describe an efficient protocol π_{COT} for COT which is secure with IVD-abort. In this section, we reduce this task to perfectly robust MPC in the presence of an honest majority.

We consider an MPC network which involves a sender S, n servers P_i, and 2α receivers $R_{i,b}$, $1 \leq i \leq \alpha$, $b \in \{0,1\}$; for simplicity, we assume that receivers do not send, but only receive messages in the protocol. (We will later set $n = O(\kappa\alpha)$.) All parties are connected via secure point-to-point channels as well as a common broadcast medium. Define the following multi-party version of COT: the sender's input consists of α pairs of strings $(y_{i,0}, y_{i,1})$ and a witness w. The other players have no input. The output of receiver $R_{i,b}$ is $\perp$ if $P(\{y_{i,b}\}, w) = 0$, and otherwise it is $y_{i,b}$.

Now, assume we are given an MPC protocol Π_{COT} that realizes this multiparty COT functionality and provides the following security guarantees. The adversary may attack up to $t = \Omega(n)$ of the servers, as well as any number of the other players (sender and receivers). For such an adversary, the protocol provides perfect correctness and, moreover, if the adversary is semi-honest we are also guaranteed *privacy*. Such a protocol, with the desired complexity, appears in [10]. We now use Π_{COT} to construct a COT protocol π_{COT} as follows.

1. Sender runs the MPC protocol Π_{COT} "in his head" (a-la [20]), where its input (α pairs of strings $(y_{i,0}, y_{i,1})$ and a witness w) serve as inputs for the sender S of Π_{COT}. It creates strings $V_1, \dots, V_n$ with the views of the n servers in this run, as well as $V_{1,0}, V_{1,1}, \dots, V_{\alpha,0}, V_{\alpha,1}$ with the views of the 2α receivers.

2. Let u be an integer such that $1/u \in [t/2n, t/4n]$. The sender and the receiver apply one parallel call to an OT in which the receiver selects, for each $i \in [n]$, a view V_{i,b_i} (where the n selection bits $b_i \in \{0,1\}$ are the COT-receiver input) as well as each of the n server views with probability $1/u$. [10]
3. Receiver checks for inconsistencies among the views that it read (namely, for each pair of views V_A, V_B, corresponding to players A, B, all messages from A to B reported in V_B should be consistent with what an honest A computes in Π_{COT} based on V_A). If any such inconsistency is found or if any of the α selected receiver views has a $\perp$ output, then the receiver outputs $\perp$; otherwise, the receiver outputs the output of the α selected receivers.

To analyze the protocol above, first note that if both Sender and Receiver are honest then the output of protocol π_{COT} is always correct (in particular, because so is Π_{COT}).

Next, consider the case of a dishonest Receiver (and honest Sender). Since, we use ideal OTs the Receiver can only choose it's selection bits which will yield exactly one of $V_{i,0}, V_{i,1}$ (for each $i \in [n]$) and each of the n server views with probability $1/u$. By the choice of u, the expected number of server views that the receiver will obtain, denoted ℓ, is $n/u \leq t/2$ and, moreover, only with a negligible probability $\ell > t$. Whenever $\ell \leq t$, the privacy property of Π_{COT} assures that from (semi-honest) views of ℓ servers and any number of receivers, no additional information (other than what the output of those receivers contain) is learned about the input of the Sender.

Finally, consider the case of a dishonest Sender (and honest Receiver). The COT simulator, given the Sender's view (in particular, the views of the MPC players), constructs the inconsistency graph G, whose nodes are the MPC players and an edge between nodes A, B whenever the corresponding views are inconsistent. In addition, G' is the sub-graph induced by the n nodes corresponding to the servers. The simulator starts by running a polynomial-time 2-approximation (deterministic) algorithm for finding minimal vertex-cover in the graph G'; i.e, the algorithm outputs a vertex cover B whose size is not more than twice the size of a minimal vertex-cover B^*. Consider two case, according to the size of B.

Case 1: $|B| > t$. In this case the simulator outputs $\perp$ with probability 1; we argue that in the real COT protocol, the receiver outputs $\perp$ with probability negligibly less than 1. This is because $|B^*| \geq |B|/2 > t/2$ and so there must be a matching in G' of size larger than $t/4$ (the size of a minimal vertex-cover of a graph is at most twice the size of a maximal matching). This, together with the choice $t = \Omega(n)$, implies that the probability that the ℓ servers picked by the Receiver do not contain an edge of G' is $2^{-\Theta(n)}$. In all other cases, the Receiver outputs $\perp$. (A similar argument was made in [20]; for more details, see there.)

Case 2: $|B| \leq t$. In this case, the COT simulator passes the views of the MPC sender and of all servers in B to the MPC simulator. The MPC simulator extracts an effective sender input (i.e., α pairs of strings and a witness w). If this input does not satisfy the predicate P then output $\perp$ (by the perfect correctness of Π_{COT}, on such input π_{COT} always outputs $\perp$ as well). It remains to deal with the case where the predicate does hold. For this, the COT simulator picks each server with probability $1/u$ (as does the honest receiver in π_{COT}) and if there is any inconsistency among the set T of selected views then the receiver outputs $\perp$; otherwise, the simulator also compares the view of each of the 2α receivers with each of the servers in T. It prepares a disjunctive predicate, P_d, consisting of the literals corresponding to receivers which have at least one such inconsistency (i.e., the predicate is satisfied exactly if the Receiver will select any of the problematic views; in both cases this leads to a $\perp$ output). It sends to the functionality the input extracted by the simulator along with the predicate P_d.

To conclude, let us summarize the complexity of our construction and compare it with the one in [23, Appendix B] (essentially the two constructions are incomparable with advantages depending on the spectrum of parameters).

Theorem 1. *The above protocol is a secure protocol* with IVD abort *for computing any* $\mathbf{NC}^0$ *function* $g(a,b)$*, where* $g : \{0,1\}^\alpha \times \{0,1\}^\beta \to \{0,1\}^\gamma$*. Its communication complexity is polylog*$(\kappa) \cdot \gamma +$poly(α, κ)*. (Recall that* $n = O(\kappa\alpha)$*.) The number of OT calls is* $O(\alpha\kappa)$*.*

Theorem 2. *[23] There exists a secure protocol* with IVD abort *for computing any* $\mathbf{NC}^0$ *function* $g(a,b)$*, where* $g : \{0,1\}^\alpha \times \{0,1\}^\beta \to \{0,1\}^\gamma$ *whose communication complexity is* $O(\kappa\gamma)$ *and number of OT calls is* $O(\alpha + \kappa)$*.*

[10] This is based on [24] which can be done non-interactively in our model. The observation is that it is known that $\binom{u}{1}$-OT non-interactively reduces to $u-1$ instances of $\binom{2}{1}$-OT. Now, given $\binom{u}{1}$-OT, a string can be transferred with probability $1/u$ simply by letting the sender put the string in a random location i of a u-entry array, and send to the receiver (independently of the receiver's selection) an additional message with i. Also note, that with our choice of parameters $u = O(1)$.

4 A Direct Protocol for NISC/NC^0

Our first construction follows a cut-and-choose approach in the spirit of previous constant-round protocols making black-box access to cryptographic primitives [34, 32]. The price we pay for this relatively simple solution is $O(\kappa)$ cryptographic and communication overheads. In particular, we show the following.

Theorem 3. *For any function $f : X \times Y \to Z$ that has a polynomial sized circuit C with n input wires for the first input, there exists an* **NC**^{0} *functionality $\mathcal{H}_C$ with $O(\kappa k|C|)$-bit long output and $n + O(\kappa)$-bit input from Receiver, such that there is an NISC/$\mathcal{H}_C$ scheme for $\mathcal{F}_C$ that makes a black-box use of a PRG, invoking the PRG $O(\kappa|C|)$ times, and with $O(\kappa k|C|)$ total communication. (Recall that κ is a statistical security parameter and k is a computational one.)*

We shall defer the proof of this theorem to Section 8, where a more general result is presented (see Theorem 6).

5 A Lean NISC/NC^0 Protocol with Input-Dependent Abort

In this section, we present a NISC scheme for $\mathcal{F}_C^\dagger$, which allows input-dependent abort. This scheme is very efficient: the communication overhead over the garbled circuit scheme is (a small) constant and the cryptographic overhead is just 1 (allowing the PRGs to output a slightly longer string). We shall present the scheme first as a NISC/$\mathcal{H}_C$ scheme, for an **NC**^{0} functionality $\mathcal{H}_C$, and then apply the result of Section 3 to obtain an NISC/OT scheme.

Theorem 4. *For any function $f : X \times Y \to Z$ that has a polynomial sized circuit C with n input wires for the first input, there exists an* **NC**^{0} *functionality $\mathcal{H}_C$ with $O(\kappa|C|)$-bit long output and $n + O(\kappa)$-bit input from Receiver, such that there is an NISC/$\mathcal{H}_C$ scheme for $\mathcal{F}_C^\dagger$ that makes a black-box use of a PRG, invoking the PRG $O(|C|)$ times, and with $O(k|C|)$ total communication.*

PROOF SKETCH: Appendix A contains the details of the proof. At a high-level, this scheme allows Receiver to verify that each pointer bit uncovered in the garbled circuit is correct as follows: each pointer bit is tagged using a MAC (with a key provided by Receiver). However since this bit should be kept secret until the corresponding entry in the garbled circuit is decrypted, a share of the tag is kept encrypted with the pointer bit, and the other share is provided to Receiver. Sender, who does not know the MAC key, can create the one share that he must encrypt, and an **NC**^{0} functionality takes the MAC key from Receiver, computes the MAC tag and hands over the other share to Receiver. Input dependent abort is obtained since, intuitively, the sender can only use wrong MACs in some entries which will make the Receiver abort in case those entries are decrypted. □

6 NISC/NC^0 with Wire-Disjunction Triggered Abort

We extend the scheme in Section 5, to achieve the stronger security guarantee of security with wire-disjunction triggered abort. Similar to the previous scheme, this scheme ensures (partial) correctness of the garbled circuit via an **NC**^{0} functionality which provides the share of a MAC to Receiver. However, the MAC is not just on a single pointer bit, but also on the key stored in the garbled circuit entry. This scheme has some features of the scheme in Section 4 in that Sender provides a table of purported outputs from a PRF, some of which will be verified by Receiver during decoding. However, this construction avoids the $O(\kappa)$ overhead, at the expense of settling for security with *wire-disjunction triggered abort*.

This construction involves a statistically secure, one-time MAC for k bit messages. It will be important for us to implement this MAC scheme using **NC**^{0} circuits. This can be done following [21], if the message is first encoded appropriately. Since the encoding itself is not an **NC**^{0} functionality, we require Sender to provide the encoding, along with values of all the wires in a circuit that computes the encoding. Then an **NC**^{0} circuit can verify this encoding, and in parallel create the MAC tag.

In Appendix B we prove the following theorem.

Theorem 5. *For any function $f : X \times Y \to Z$ that has a polynomial sized circuit C with n input wires for the first input, there exists an* **NC**^{0} *functionality $\mathcal{H}_C$ with $O(k|C|)$-bit long output and $n + O(\kappa)$-bit input from Receiver, such that there is an NISC/$\mathcal{H}_C$ scheme for $\mathcal{F}_C^{\ddagger}$ that makes a black-box use of a PRG, invoking the PRG $O(|C|)$ times, and with $O(k|C|)$ total communication.*

Compared to Theorem 4, this construction is asymptotically less efficient, since the output of $\mathcal{H}_C$ is longer ($O(k|C|)$ instead of $O(\kappa|C|)$, as $\mathcal{H}_C$ will now be required to deliver the entire garbled srcuit to Receiver).

7 From Security with WDT-Abort to Full Security

In this section, we discuss general methods for converting any NISC scheme satisfying security with *wire disjunction triggered* (WDT) abort into an NISC with full security, based on semi-honest secure MPC protocols. Our transformation formalizes and generalizes such a transformation that was given in the work of [25, 26] (and our intuition below follows their intuition) in the context of constructing stateful hardware circuits that remain private even when an adversary can tamper with the values on wires. We note that the construction of [25] also had to deal with multiple other issues that do not concern us, which added complexity to their solution. Outlined below, our solution can be seen as a simplification of their construction.

The benefit of the transformation outlined in this section over the simple majority-based approach discussed earlier is the potential for greater efficiency. We will first formalize the encoding notion that we use to deal with WDT attacks, then we present an outline of our general transformation, and then show how to invoke this transformation using known semi-honest MPC protocols from the literature to obtain higher levels of efficiency.

Our transformation is captured by means of a new encoding, that we define below. The details of realizing this transformation are presented in the full version.

Definition 1. ***(WDT-resilient encoding)*** *A randomized circuit family C' together with an efficient randomized encoding algorithm family Enc and an efficient deterministic decoding algorithm family Dec is a* WDT-resilient encoding *of a circuit C that takes two inputs if the following properties hold:*[11]

(Correctness) For all (x, y) in the domain of C, we have that

$$\Pr[Dec(C'(Enc(x), y)) = C(x, y)] = 1$$

(Malicious Receiver Security) There exists a randomized efficient machine $RecSim$ such that for every x' in the range of Enc (but not necessarily in the image of Enc), there exists x in the domain of Enc such that for every y such that (x, y) is in the domain of C, the output distribution of $RecSim(x', C(x, y))$ is identical to the distribution $C'(x', y)$.

(WDT-Malicious Sender Security) For any set S of wires in C' or their negations, let $Disj_S[C'(Enc(x), y)]$ to be the event that the disjunction of the values specified by S, when the input of C' is $(Enc(x), y)$, is satisfied. The probability space is over the random gates of C' and the randomness used by Enc.
For any such S and for all x_1, x_2, and y such that (x_1, y) and (x_2, y) are in the domain of C, we have:

$$|\Pr[Disj_S[C'(Enc(x_1), y)]] - \Pr[Disj_S[C'(Enc(x_2), y)]]| = 2^{-\Omega(\kappa)}.$$

8 Public-Code NISC

So far we only considered NISC schemes which rely on an OT oracle that gets inputs from both the sender and the receiver. As discussed in the introduction, this can be combined with a 2-message implementation of OT to get a protocol which does not require any active action from the receiver except publishing an encryption of her input.

[11] The entire tuple (C', Enc, Dec) is parameterized by a statistical security parameter 1^κ, which is omitted here for simplicity of notation. Note also that this definition defines the notion of a WDT-resilient encoding. In applications, we will require that there is an efficient deterministic procedure that takes as input C and 1^κ and outputs a tuple (C', Enc, Dec) from such a family.

In this section we discuss this variant of NISC, called Public-Code NISC or PC-NISC for short. In more detail, this flavor of NISC allows Receiver to publish an encoding of her input x, and later let one or more Senders compute on the encoding of x using their private inputs y, and send it back to her; she can decode this message and recover the value $f(x, y)$ (and nothing more). There could be a setup like a common reference string (CRS), or correlated random variables.

Formally, a PC-NISC scheme for a function $f : X \times Y \to Z$ consists of the following four PPT algorithms.

- Setup: takes only the security parameter as an input and outputs a pair of strings (σ^R, σ^S). These strings are meant to be given to the two parties (Receiver and Sender, respectively).
- Encode: takes an input $x \in X$, and a setup string σ^R, and outputs a string c encoding x (or possibly an error message, if σ^R appears malformed).
- $\mathsf{Compute}$: takes an encoding c, an input $y \in Y$ and a setup string σ^S and outputs an "encoded output" (or an error message if c appears malformed).
- Decode: takes an encoded output and a setup string σ^R, and outputs $z \in Z$ (or an error message if the encoded output appears malformed).

Ideally, in a PC-NISC scheme, a single published encoding can be used by Receiver to carry out multiple computations. To define the security of a PC-NISC scheme, below we define the functionality $\mathcal{F}_f^{(T)}$, which allows T invocations before letting a corrupt Sender manipulate the outcome.

- $\mathcal{F}_f^{(T)}$ accepts an input x from Receiver.
- Then in each round, it accepts an input y from Sender. and outputs $f(x, y)$ to Receiver (and an empty output to Sender). If y is a special command error, the output to Receiver is error.
- There is a bound T on the number of inputs $\mathcal{F}_f^{(T)}$ accepts from corrupt Senders before correctness is compromised. More formally, a corrupt Sender is allowed to include with its input a command (cheat, ψ) where ψ is an arbitrary PPT algorithm, and after T such rounds, in each subsequent such round, $\mathcal{F}_f^{(T)}$ outputs $\psi(x)$ to Receiver.

Now, given a PC-NISC scheme Σ consider the 2-party protocol Π_Σ (in a $\mathcal{F}_{\Sigma.\mathsf{Setup}}$-hybrid model, which simply makes a fresh pair (σ^R, σ^S) available to the two parties) in which Receiver, on input x, sends $c := \Sigma.\mathsf{Encode}(x, \sigma^R)$ to Sender; on receiving an input y reactively from the environment, Sender sends $u = \Sigma.\mathsf{Compute}(c, y, \sigma^S)$ to Receiver, and Receiver outputs $\Sigma.\mathsf{Decode}(u)$. We say that Σ is a secure PC-NISC scheme if the protocol $\Pi_\Sigma^{\mathcal{F}_{\Sigma.\mathsf{Setup}}}$ is a UC secure realization of the functionality $\mathcal{F}_f^{(T)}$.

We shall be interested in NISC schemes for $\mathcal{F}_f^{(T)}$, where $T = \Omega(\kappa)$.

Defining PC-NISC/$\mathcal{H}$. The goal of PC-NISC was to avoid the live availability of Receiver, when Sender is executing the scheme. However it is still possible to consider such a scheme in an $\mathcal{H}$-hybrid model, if the functionality $\mathcal{H}$ itself allows Receiver to send an input, and subsequently have multiple rounds of independent interactions with Sender, delivering a separate output to Receiver in each round. We shall use this convention as an intermediate step in achieving PC-NISC/OT and PC-NISC/CRS schemes, which can be specified in the plain model (i.e., without reference to a hybrid-model) in terms of the Setup algorithm. In PC-NISC/CRS, Setup sets $\sigma^R = \sigma^S$ to be a randomly generated string, according to some probability distribution that will be specified by the scheme.

In PC-NISC/OTvar, Setup outputs several instances of correlated random variables: in each instance, Receiver gets two random bits (a_0, a_1) and Sender gets random bits (b_0, b_1) such that $a_0 b_0 = a_1 \oplus b_1$.[12] They can be readily used in a PC-NISC scheme Σ_0 for evaluating the OT function, in which Receiver has a choice bit c, Sender has two inputs x_0 and x_1, and Receiver obtains x_c. Hence a NISC/OT scheme for a function f can be easily turned into a PC-NISC/OTvar scheme for f *if the number of sessions to be supported* $T = 1$: the Encode and $\mathsf{Compute}$ algorithms will incorporate $\Sigma_0.\mathsf{Encode}$ and $\Sigma_0.\mathsf{Compute}$; further, $\mathsf{Compute}$ will include the message sent by Sender in the NISC/OT scheme; Decode involves first applying $\Sigma_0.\mathsf{Decode}$ to obtain the outcome of OT, before carrying out the local computation of the NISC/OT scheme.

The main challenge in constructing a PC-NISC scheme, beyond that already present in constructing NISC schemes, is to be able to support a larger number of computations for the same input encoding.

[12] There are several equivalent formulations of such a pair of correlated random variables.

First, we observe that the NISC/OT scheme for $\mathbf{NC}^0$ functionalities from Section 3 can be extended into a PC-NISC/OTvar supporting T adding a $poly(\kappa, T)$ amount to communication and cryptographic complexities. This is done by increasing the number of servers in the underlying MPC used in this scheme.

In the full version we prove the feasibility result below, analogous to – indeed extending – Theorem 3.

Theorem 6. *For any function $f : X \times Y \to Z$ that has a polynomial sized circuit C with n input wires for the first input, there exists an $\mathbf{NC}^0$ functionality $\mathcal{H}_C^{(T)}$ with $O(\kappa k|C|)$-bit long output and $n + O(\kappa)$-bit input from Receiver, supporting T computations, such that there is a NISC/$\mathcal{H}_C^{(T)}$ scheme for $\mathcal{F}_f^{(T)}$ that makes a black-box use of a PRG, invoking the PRG $O((\kappa + T)|C|)$ times, and with $O((\kappa + T)k|C|)$ total communication.*

Note that the above NISC scheme is already for $\mathcal{F}_f^{(T)}$, and can be translated to a PC-NISC scheme for f supporting T executions, as described earlier. Thus, given this scheme, we can combine it with a PC-NISC/OTvar for $\mathcal{H}_C^{(T)}$ (also described above) to obtain a PC-NISC/OTvar for $\mathcal{F}_f^{(T)}$. A proof of Theorem 6 is given in the full version.

References

1. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. In *IEEE Conference on Computational Complexity*, pages 260–274. IEEE Computer Society, 2005.
2. Donald Beaver. Precomputing oblivious transfer. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 97–109. Springer, 1995.
3. Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proc.* 28*th STOC*, pages 479–488. ACM, 1996.
4. Donald Beaver and Shafi Goldwasser. Multiparty computation with faulty majority. In *CRYPTO*, pages 589–590, 1989.
5. Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, pages 503–513. ACM, 1990.
6. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.
7. Christian Cachin, Jan Camenisch, Joe Kilian, and Joy Müller. One-round secure computation and secure autonomous mobile agents. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 512–523. Springer, 2000.
8. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Electronic Colloquium on Computational Complexity (ECCC) TR01-016, 2001. Previous version "A unified framework for analyzing security of protocols" availabe at the ECCC archive TR01-016. Extended abstract in FOCS 2001.
9. Kai-Min Chung, Yael Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In Tal Rabin, editor, *CRYPTO '10*, volume 6223 of *Lecture Notes in Computer Science*, pages 483–501. Springer, 2010.
10. Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 445–465. Springer, 2010.
11. Ivan Damgård, Jesper Buus Nielsen, and Claudio Orlandi. Essentially optimal universally composable oblivious transfer. In Pil Joong Lee and Jung Hee Cheon, editors, *ICISC*, Lecture Notes in Computer Science. Springer, 2008.
12. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO '10*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2010.
13. Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178. ACM, 2009.
14. Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. *i*-hop homomorphic encryption and rerandomizable yao circuits. In Tal Rabin, editor, *CRYPTO '10*, volume 6223 of *Lecture Notes in Computer Science*, pages 155–172. Springer, 2010.
15. Oded Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.
16. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play ANY mental game. In ACM, editor, *Proc.* 19*th STOC*, pages 218–229. ACM, 1987. See [15, Chap. 7] for more details.
17. Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 111–129. Springer, 2007.
18. Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO '03*, volume 2729 of *Lecture Notes in Computer Science*, pages 145–161. Springer, 2003.

19. Yuval Ishai and Eyal Kushilevitz. On the hardness of information-theoretic multiparty computation. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 439–455. Springer, 2004.
20. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30. ACM, 2007.
21. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *STOC*, pages 433–442. ACM, 2008.
22. Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594. Springer, 2007.
23. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. Preliminary full version on `http://www.cs.uiuc.edu/~mmp/`.
24. Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO '08*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008.
25. Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 308–327. Springer, 2006.
26. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO '03*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.
27. Yael Tauman Kalai and Ran Raz. Succinct non-interactive zero-knowledge proofs with preprocessing for logsnp. In *FOCS*, pages 355–366. IEEE, 2006.
28. Joe Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31. ACM, 1988.
29. Joe Kilian, Silvio Micali, and Rafail Ostrovsky. Minimum resource zero-knowledge proofs (extended abstract). In *FOCS*, pages 474–479. IEEE, 1989.
30. Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373. IEEE, 1997.
31. Yehuda Lindell and Benny Pinkas. A proof of yao's protocol for secure two-party computation. *Electronic Colloquium on Computational Complexity (ECCC)*, (063), 2004.
32. Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, *EUROCRYPT*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78. Springer, 2007.
33. Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. Additively homomorphic encryption with -operand multiplications. In Tal Rabin, editor, *CRYPTO '10*, volume 6223 of *Lecture Notes in Computer Science*, pages 138–154. Springer, 2010.
34. Payman Mohassel and Matthew K. Franklin. Efficiency tradeoffs for malicious two-party computation. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 458–473. Springer, 2006.
35. Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On epsilon-biased generators in nc^0. *Random Struct. Algorithms*, 29(1):56–81, 2006.
36. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.
37. Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *ACM Conference on Electronic Commerce*, pages 129–139, 1999.
38. Jesper Buus Nielsen and Claudio Orlandi. Lego for two-party secure computation. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 368–386. Springer, 2009.
39. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO '08*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.
40. Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. Secure two-party computation is practical. In Mitsuru Matsui, editor, *ASIACRYPT '09*, volume 5912 of *Lecture Notes in Computer Science*, pages 250–267. Springer, 2009.
41. Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC '04*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.
42. Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of secure computation (Workshop, Georgia Inst. Tech., Atlanta, Ga., 1977)*, pages 169–179. Academic, New York, 1978.
43. Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC^1. In *FOCS*, pages 554–567, 1999.
44. Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proc.* 27*th FOCS*, pages 162–167. IEEE, 1986.

A Proof of Theorem 4

In this section, we present a NISC scheme for $\mathcal{F}_C^{\dagger}$, which allows input-dependent abort. This scheme is very efficient: the communication overhead over the garbled circuit scheme is (a small) constant and the cryptographic overhead is just 1 (allowing the PRGs to output a slightly longer string). We shall present the scheme first as a NISC/$\mathcal{H}_C$ scheme, for an $\mathbf{NC}^0$ functionality $\mathcal{H}_C$, and then apply the result of Section 3 to obtain an NISC/OT scheme.

Theorem 7 (Theorem 4 (restated)). *For any function $f : X \times Y \to Z$ that has a polynomial sized circuit C with n input wires for the first input, there exists an $\mathbf{NC}^0$ functionality $\mathcal{H}_C$ with $O(\kappa|C|)$-bit long output and $n + O(\kappa)$-bit input from Receiver, such that there is an NISC/$\mathcal{H}_C$ scheme for $\mathcal{F}_C^{\dagger}$ that makes a black-box use of a PRG, invoking the PRG $O(|C|)$ times, and with $O(k|C|)$ total communication.*

PROOF SKETCH: First we present the $\mathbf{NC}^0$ functionality $\mathcal{H}_C$.

- Sender's inputs consist of the following:
  - an input y for C;
  - for each wire w, keys K_w^0 and K_w^1 and wire mask r_w for a garbled circuit construction Y_C;
  - a κ-bit string $\tau_{g,a,b}^S$ for each entry (g, a, b) of the garbled circuit Y_C.
- Receiver's inputs are an input x for C and a pair of κ long bit-strings τ_0 and τ_1.
- The functionality outputs to Receiver the following pieces of information:
  - For each input-wire w in Y_C that belongs to Receiver, $K_w^{x_w}$ (for such w, we have $r_w = 0$).
  - For each entry in the garbled circuit, indexed by (g, a, b), a string $\tau_{g,a,b}^R := \tau_{g,a,b}^S \oplus \tau_{z \oplus r_w}$, where $z = F_g(a \oplus r_u, b \oplus r_v)$, where u and v are the input wires to g.

Now we define a scheme Σ for the SFE functionality for evaluating C as follows:

- Receiver and Sender invoke $\mathcal{H}_C$. Receiver's inputs are (x, τ_0, τ_1) where τ_0 and τ_1 are chosen at random; Sender's input are y and randomly chosen values for the construction of the garbled circuit and random values for $\tau_{g,a,b}^S$ for each entry (g, a, b) of the garbled circuit. Receiver obtains as output the keys $K_w^{x_w}$ for each input-wire w that belongs to her, and for each tuple (g, a, b), a string $\tau_{g,a,b}^R$.
- Sender sends Y_C^* which is constructed similar to Y_C, but modified as follows: the ciphertext indexed by (g, a, b) where g is a gate (with input wires u, v and output wire w) and (a, b) a pair of bits indicating the masked values of the input wires is $\mathrm{Encr}^{g,a,b}_{K_u^{a \oplus r_u}, K_v^{b \oplus r_v}}(K_w^{c \oplus r_w}, c, \tau_{g,a,b}^S)$ where $c = \tilde{F}_g(a, b) := F_g(a \oplus r_u, b \oplus r_v) \oplus r_w$ (i.e., $\tau_{g,a,b}^S$ is added to the entry for (g, a, b) before encrypting it).
- Receiver uses the keys $K_w^{x_w}$ to carry out the evaluation of the garbled circuit Y_C^*. In the process, for each gate g, she obtains (purported values of) $\tau_{g,a,b}^S$ for some pair of bits (a, b). For each such tuple (g, a, b), she checks that $\tau_{g,a,b}^S \oplus \tau_{g,a,b}^R$ is indeed equal to τ_c, where c is the pointer bit obtained by decrypting the entry for that tuple in the garbled circuit. If the check fails, then she outputs error. Otherwise, she outputs the value obtained by evaluating the garbled circuit.

Security against corrupted Receiver immediately reduces to the computational privacy of the garbled circuit. Security against corrupted Sender is based on the following observation. Since Sender cannot guess $\tau_0 \oplus \tau_1$ with more than $2^{-\Omega(\kappa)}$ probability, he has no more chance of preparing a ciphertext that will decrypt to $\tau_{g,a,b}^S \oplus \tau_0 \oplus \tau_1$. Hence, we can ignore the event that Receiver accepts a masked value that is incorrect according to the values submitted to $\mathcal{H}_C$. Thus (with probability $1 - 2^{-\Omega(\kappa)}$), Receiver obtains the output $f(x, y)$ according to the values submitted to $\mathcal{H}_C$, or aborts. The probability of aborting can indeed depend on the input x, but since the functionality $\mathcal{F}_C^{\dagger}$ allows input-dependent abort, this can be easily simulated. Specifically, the simulator submits the algorithm ϕ to $\mathcal{F}_C^{\dagger}$, which honestly runs $\mathcal{H}_C$ and Receiver's code (using Receiver's input x, and with access to the PRG) on the messages from Sender, and if Receiver aborts, outputs abort. □

B Proof of Theorem 5

This construction involves a statistically secure, one-time MAC for k-bit messages. It will be important for us to implement this MAC scheme using $\mathbf{NC}^0$ circuits. This can be done following [21], if the message is first encoded appropriately. Since the encoding itself is not an $\mathbf{NC}^0$ functionality, we require Sender to provide the encoding, along with values of all the wires in a circuit that computes the encoding. Then, an $\mathbf{NC}^0$ circuit can verify this encoding and, in parallel, create the MAC tag.

Theorem 8 (Theorem 5 (restated)). *For any function $f : X \times Y \to Z$ that has a polynomial sized circuit C with n input wires for the first input, there exists an $\mathbf{NC}^0$ functionality $\mathcal{H}_C$ with $O(k|C|)$-bit long output and $n + O(\kappa)$-bit input from Receiver, such that there is an NISC/$\mathcal{H}_C$ scheme for $\mathcal{F}_C^{\ddagger}$ that makes a black-box use of a PRG, invoking the PRG $O(|C|)$ times, and with $O(k|C|)$ total communication.*

Compared to Theorem 4, this construction is asymptotically less efficient since the output of $\mathcal{H}_C$ is longer ($O(k|C|)$ instead of $O(\kappa|C|)$, as $\mathcal{H}_C$ will now be required to deliver the entire garbled srcuit to Receiver).

PROOF SKETCH: First, we define an $\mathbf{NC}^0$ functionality $\mathcal{H}_C$, as follows:

- Receiver's inputs are an input x to C and a key SK for the MAC.
- Sender's inputs consist of an input y for C, and the following information for the construction of a garbled circuit Y_C (using a PRF R) with input y:
  - for each wire w in C, keys K_w^0, K_w^1 and mask r_w;
  - an auxliary information table which contains entries of the form $(g, u, b : z)$ for each gate g, each input-wire u to gate g, and each value $b \in \{0, 1\}$, where (purportedly) $z = R_{K_u^{b \oplus r_u}}(g)$. (Also, appropriate encodings of collections of such z values, to allow applying an $\mathbf{NC}^0$ MAC; see below.)
  - a κ-bit string $\tau_{w,b}^S$ for each wire w and each $b \in \{0, 1\}$.
- The functionality outputs to Receiver the following pieces of information:
  - the garbled circuit Y_C, using the values provided by Sender (without querying the PRF);
  - for each input-wire w in Y_C that belongs to Receiver, $K_w^{x_w}$.
  - for each wire w and each $b \in \{0, 1\}$, a string $\tau_{w,b}^R := \tau_{w,b}^S \oplus \mathsf{MAC}_{SK}(z_1, \ldots, z_t)$, such that the auxiliary information table has entries $(g_i, w, b : z_i)$ where $g_1, \ldots, g_t$ are all the gates (in some canonical order) that the wire w fans-out to.[13]

We define an NISC/$\mathcal{H}_C$ scheme Σ for the functionality $\mathcal{F}_C^{\ddagger}$ as follows.

- Receiver and Sender invoke $\mathcal{H}_C$. Receiver's inputs are x and a freshly generated key SK for the MAC scheme. Sender's inputs are y and freshly generated keys and masks for the garbled circuit, an auxiliary information table as expected by $\mathcal{H}_C$ (with values filled in correctly from the PRF R), and random strings $\tau_{w,b}^S$ for each wire w and $b \in \{0, 1\}$.
  Receiver obtains as output a garbled circuit Y_C, key $K_w^{x_w}$ for each input-wire w that belongs to Receiver, and $\tau_{w,b}^R$ for each wire w and $b \in \{0, 1\}$.
- Sender sends $\alpha_{(w,b)} := \tau_{w,b}^S \oplus R_{K_w^{b \oplus r_w}}(0)$ (where 0 is an index not used for any gate), for all wires w and $b \in \{0, 1\}$.
- Receiver decodes the garbled circuit Y_C and, in the process, recovers (a purported value of) the key $K_w^{b \oplus r_w}$ for each wire w and a value b. It uses this key to recover (a purported value of) $\tau_{w,b}^S$ from $\alpha_{(w,b)}$ and combining it with $\tau_{w,b}^R$ reconstructs (a purported value of) $\mathsf{MAC}_{SK}(z_1, \ldots, z_t)$ where $z_i = R_K(g_i)$, with K being the purported value of $K_w^{b \oplus r_w}$ that was recovered. If the MAC tag does not verify for any (w, b) then it outputs abort. Otherwise, it outputs the value that was obtained from evaluating the garbled circuit.

[13] As mentioned above Sender's input also includes an appropriate encoding of $(z_1, \ldots, z_t)$ (along with an $\mathbf{NC}^0$ verifiable proof of correctness of the encoding) so that it can be used in an $\mathbf{NC}^0$ MAC scheme [21].

Security for the case of a corrupted Receiver easily reduces to the privacy of the garbled circuit construction. For the case of a corrupted Sender, we consider a simulator which simulates $\mathcal{H}_C$ to Sender, and interacts with $\mathcal{F}_f^{\ddagger}$. If the (simulated) $\mathcal{H}_C$ outputs error, or if the output encoding from Sender does not include $\alpha_{(w,b)}$ for all wires w and $b \in \{0,1\}$, then the simulator sends error to $\mathcal{F}_f^{(T)}$. Otherwise, it defines a set W to include all (w,b) such that either there is an entry $(g_i, w, b : z_i)$ in the auxiliary information table such that $z_i \neq R_{K_w^{b \oplus r_w}}(g_i)$, or $\alpha_{(w,b)} \neq \tau_{w,b}^S \oplus R_{K_w^{b \oplus r_w}}(0)$. It will send this set to $\mathcal{F}_f^{\ddagger}$ as the set of wire-values that trigger abort.

To see that this simulation is good, firstly, note that if on evaluating the circuit C with inputs (x,y), for every wire w, the value it takes b is such that $(w,b) \notin W$, then in the real execution there will be no abort. Further, the outcome will indeed be $f(x,y)$. On the other hand, if the evaluation of C with inputs (x,y) leads to a value b for some wire w such that $(w,b) \in W$, we need to argue that the real execution leads to abort. Consider a topologically first gate in the circuit whose output-wire w is such a wire. Note that this means that Receiver correctly recovers $K_w^{b \oplus r_w}$. Now suppose that Receiver's check verifies for (w,b). Then, $\mathsf{MAC}_{SK}(z_1^*, \ldots, z_t^*) = \mathsf{MAC}_{SK}(z_1, \ldots, z_t) \oplus \tau_{w,b}^S \oplus \alpha_{(w,b)} \oplus R_{K_w^{b \oplus r_w}}(0)$, where $z_i^* := R_{K_w^{b \oplus r_w}}(g_i)$, z_i is the entry for (g_i, w, b) in the auxiliary information table, and $\alpha_{(w,b)}$ is part of the output encoding provided by Sender. If, for all entries $(g_i, w, b : z_i)$ in the auxiliary information table submitted by Sender, $z_i = z_i^*$, then Receiver's check will verify only if $\alpha_{(w,b)} \oplus R_{K_w^{b \oplus r_w}}(0) \oplus \tau_{w,b}^R = \mathsf{MAC}_{SK}(z_1, \ldots, z_t)$, and therefore $\alpha_{(w,b)} = \tau_{w,b}^S \oplus R_{K_w^{b \oplus r_w}}(0)$, contradicting the assumption that $(w,b) \in W$. On the other hand, if for some i, $z_i \neq z_i^*$, then Sender has managed to compute $\mathsf{MAC}_{SK}(z_1^*, \ldots, z_t^*) \oplus \mathsf{MAC}_{SK}(z_1, \ldots, z_t)$ (without seeing a MAC tag or SK), contradicting the one-time security of the MAC. □

C Proof of Theorem 6

Theorem 9 (Theorem 6 (restated)). *For any function $f : X \times Y \rightarrow Z$ that has a polynomial sized circuit C with n input wires for the first input, there exists an $\mathbf{NC}^0$ functionality $\mathcal{H}_C^{(T)}$ with $O(\kappa k|C|)$-bit long output and $n + O(\kappa)$-bit input from Receiver, supporting T computations, such that there is a NISC/$\mathcal{H}_C^{(T)}$ scheme for $\mathcal{F}_f^{(T)}$ that makes a black-box use of a PRG, invoking the PRG $O((\kappa + T)|C|)$ times, and with $O((\kappa + T)k|C|)$ total communication.*

PROOF SKETCH: For the sake of convenience, we define an intermediate functionality that we will realize using a simple NISC scheme, and show how to amplify it to an NISC scheme with full-fledged security.

For any 2-party function f with output for only one party (Receiver), we define (a multi-round) functionality $\mathcal{L}_f$ as follows:

- When initialized, $\mathcal{L}_f$ accepts an input x from Receiver. Also, it picks a random vector $L \leftarrow \{1,2\}^\ell$; if Receiver is corrupt, then L is provided by Receiver.
- At each round, $\mathcal{L}_f$ behaves as follows:
  - It accepts y from Sender. In addition, if Sender is corrupt, it can send $(\mathsf{cheat}, \hat{L}, \psi)$ to $\mathcal{L}_f$, where $\hat{L} \in \{0,1,2,3\}^\ell$, and ψ is a PPT function (represented as, say a circuit) used below.
  - $\mathcal{L}_f$ gives an empty output to Sender and outputs a vector $(z_1, \ldots, z_\ell)$ to Receiver, defined as follows. If Sender has not sent the cheat command, $z_i = f(x,y)$ for all i. Else, for each i, if $\hat{L}_i = 0$ then $z_i = f(x,y)$, but if $\hat{L}_i = 3 - L_i$ then, $z_i = \psi(i,x)$, and otherwise (i.e., if $\hat{L}_i \in \{L_i, 3\}$) $z_i = \mathsf{error}$.

An NISC scheme Σ for $\mathcal{L}_f$ can be easily modified to be a fully secure NISC scheme Σ^* for computing f, in a black-box manner, as follows. Below we let $\ell \geq 2(\kappa + T)$ where T is the bound on the number of computations a single encoding from Σ^* can be used before correctness is compromised.

- Σ^* has identical Setup, Encode and $\mathsf{Compute}$ algorithms as Σ.
- $\Sigma^*.\mathsf{Decode}$ runs $\Sigma.\mathsf{Decode}$ to obtain $(z_1, \ldots, z_\ell)$; if for any i, $z_i = \mathsf{error}$, then it outputs error. Otherwise, it checks if more than $\ell/2$ of z_i have the same value, and if so, outputs that value; if no such value exists, it outputs error.

To prove that the above NISC scheme is indeed secure, firstly, note that the use of Σ can be replaced by access to $\mathcal{L}_f$. We consider the cases where each of Receiver and Sender is corrupt.

For the first case (corrupt Receiver), the security is immediate: a simulator can simply provide the vector $(z_1, \ldots, z_\ell)$, with $z_i = f(x, y)$ for all i (where x was extracted from the call to $\mathcal{L}_,$ and $f(x, y)$ is obtained in each round from $\mathcal{F}_f^{(T)}$.

To argue the security for the second case (corrupt Sender), we consider a series of hybrids. Hybrid 0 is the same as the real protocol execution (in the $\mathcal{L}_f$-hybrid model). In Hybrid 1, this real execution is carried out by a simulator who is given Receiver's inputs, and who (faithfully) executes $\mathcal{L}_f$, and is also allowed to specify the output for Receiver. However, we shall describe this execution slightly differently, by setting values of L progressively. Initially all L_i are left undetermined. In a round being simulated, on receiving input y and possibly $(\mathsf{cheat}, \hat{L}, \psi)$ from Sender to $\mathcal{L}_f$, the simulator behaves as follows:

- If there is no cheat command, or if there is a cheat command, but $\hat{L}_i = 0$ for all i, then set the output for Receiver as $f(x, y)$ (and do not change L).
- Else, if $\hat{L}_i = 3$ for any i, or if $\hat{L}_i = L_i$ for any i for which L_i has already been set, then set Receiver's output to error.
- Else, let m be the number of indices i such that L_i is not set, and $\hat{L}_i \in \{1, 2\}$. Then:
  - With probability 2^{-m} do the following: for all m such indices i, set $L_i = 3 - \hat{L}_i$. In this case, set z_i, for $i = 1$ to ℓ, as $f(x, y)$ or $\psi(i, x)$, depending on L_i and $\hat{L}_i$ as in the definition of $\mathcal{L}_f$, and set Receiver's output to be the majority among z_i, or if no majority exists, then error.
  - With probability $1 - 2^{-m}$, pick at random one of the m indices i, and set $L_i = \hat{L}_i$. In this case, set Receiver's output to be error.

It is easy to see that the environment's view of Hybrid 1 is identical to that of Hybrid 0 (the real execution). In particular, note that the probability of error occuring because of some z_i being equal to error is correctly simulated.

In Hybrid 2, the simulator does not use Receiver's inputs, nor set the output values for Receiver, but instead interacts with Receiver via $\mathcal{F}_f^{(T)}$. This simulator is specified as follows, in terms of its differences from the previous one:

- For the cases for which the Hybrid 1 simulator set Receiver's output to $f(x, y)$, send y to $\mathcal{F}_f^{(T)}$. For the cases for which it set Receiver's output to error (excluding the case which depended on finding the majority of z_i values).
- For the case for which the Hybrid 1 simulator based Receiver's output on the majority of z_i values, instead do the following:
  - If L is set in less than $\ell/2$ positions, send y to $\mathcal{F}_f^{(T)}$.
  - If L is set in $\ell/2$ or more positions, but fewer than T rounds have passed, then abort the simulation. On the other hand, if L is set in $\ell/2$ or more positions, and at least T rounds have passed, then set each of the undetermined position in L to a random element from $\{1, 2\}$. In each subsequent round send (cheat, ψ^*) to $\mathcal{F}_f^{(T)}$ where $\psi^*(x)$ computes z_i, for $i = 1$ to ℓ, as $\psi(i, x)$ of $f(x, y)$ depending on L_i and $\hat{L}_i$ as in the definition of $\mathcal{L}_f$, and outputs the majority of such output values, if it exists, or error otherwise.

Note that the probability that this simulation aborts, i.e., the probability thatbefore T inputs, at least $\ell/2$ positions in L will be set, is at most $2^{-(\ell/2-T)}$. With $\ell \geq 2(\kappa + T)$, this is at most $2^{-\kappa}$, and so we shall condition on this not happening (in either Hybrid 1 or Hybrid 2). Conditioned on this, the two simulations are identical, since when L is set in less than $\ell/2$ positions, the outcome computed as majority of z_i is indeed the same as $f(x, y)$.

Securely Realizing $\mathcal{L}_f$. We describe an NISC scheme which shall access a PRF $R' : \{0,1\}^{k/2} \times [|C|] \to \{0,1\}^{k+1}$. The construction involves a garbled circuit as described earlier, using the particular Encr scheme in Equation (1), with the PRF R defined as follows: $R_{K_0 K_1}(g) = R'_{K_0}(g) \oplus R'_{K_1}(g)$ (note that the key for R' is half of the key for R; w.l.o.g we assume that k is even).

First we define an $\mathbf{NC}^0$ functionality $\mathcal{H}_C^{(T)}$.

- From Receiver, the function takes as input x (which has a bit for each input-wire w that belongs to Receiver), and a vector $L \in \{1, 2\}^\ell$ (indicating her choice of which auxiliary information table described she wants to check).

- In each round, from Sender it takes as input y for the circuit C, and the following information for constructing ℓ garbled circuits Y_C^i ($i = 1$ to ℓ) for C with input y, based on the PRF R'. For each i:
  - the keys and masks for all the wires in Y_C^i. (To avoid clutter we denote them as K_w^0, K_w^1 and r_w, without the index i).
  - two tables A_1^i and A_2^i, with the following auxiliary information: A_1^i consists of tuples of the form $(g, u, b : K, z)$ for each gate g, each input-wire u to gate g, and each value $b \in \{0, 1\}$, where K is the first half of $K_u^{b \oplus r_u}$ (in the i^{th} garbled circuit), and $z = R'_K(g)$; similarly, A_2^i has tuples of the form $(g, u, b : K, z)$, where K is the second half of $K_u^{b \oplus r_u}$.
- In each round, the output of $\mathcal{H}_C^{(T)}$ (only for Receiver) consists of the garbled circuits Y_C^i themselves based on the information provided by Sender (without querying the PRF); further Receiver gets $A_{L_i}^i$, and for each input-wire w, the key $K_w^{x_w}$ in each garbled circuit, where x_w is the input bit specified by Receiver for the wire w.
- After T rounds, $\mathcal{H}_C^{(T)}$ allows Sender to control its output.

Note that $\mathcal{H}_C^{(T)}$ does not check if the values in the auxiliary information table are correct, as this will involve non-black-box access to the PRG.

Now we define a NISC/$\mathcal{H}_C^{(T)}$ scheme Σ for the functionality $\mathcal{L}_f$ (where the circuit C computes the function f), which uses the PRF R' in a blackbox manner.

- In each round Receiver and Sender invoke $\mathcal{H}_C^{(T)}$. Receiver's inputs (only in the first round) are x and a randomly chosen value for $L \in \{1, 2\}^\ell$. Sender constructs ℓ garbled circuits Y_C^i independently for C and input y, and uses the resulting information (as specified in the definition of $\mathcal{H}_C^{(T)}$) to derive its input. Receiver obtains as output the ℓ garbled circuits, with keys $K_w^{x_w}$ for each input wire she owns, and $A_{L_i}^i$, for each i.
- Next she locally checks that the auxiliary information tables obtained are consistent with R'. If these checks for a garbled circuit i do not verify, it sets $z_i = \mathsf{error}$. Otherwise, for each i, she carries out the garbled circuit evaluation to obtain a value that she sets as z_i. Her output is $(z_1, \ldots, z_\ell)$.

We shall reduce the security of the above scheme to the security of the garbled circuit construction. Security for the case when the Receiver is corrupt can be shown using the following simulation:

- First, the simulator obtains the vector $L \in \{1, 2\}^\ell$ that the Receiver feeds into the (simulated) functionality $\mathcal{H}_C^{(T)}$. For simplicity of description we shall assume that $L = (2, \ldots, 2)$.
- Then for each round, it simulates the output of $\mathcal{H}_C^{(T)}$ as follows.
  - Consider a modified garbled circuit defined as follows: the keys are k bits long, but the second half of the key is all 0's. Encr uses R' with only the first half of each key. This Encr function meets the security conditions required for the security of a garbled circuit. Hence, given an output value, the entire garbled circuit can be simulated. So first, for each i, such a garbled circuit is simulated for the output $f(x, y)$.
  - Then, the other halfs of the wire-keys (namely, the halfs revealed by the auxiliary information A_2^i) are picked randomly, and the simulated garbled circuit is modified by XORing to the appropriate location in each ciphertext the appropriate half-key and XORing the entire ciphertext with the output of the PRF using the appropriate half-key.

For the case when Sender is corrupt, the simulation is simpler. One can simulate $\mathcal{H}_C^{(T)}$ without access to x (and with random L), except for forcing the output for Receiver. However, if a garbled circuit Y_C^i is constructed correctly, or if it has an error that is caught by $\mathcal{H}_C^{(T)}$, then the output z_i can be forced by sending y and setting $\hat{L}_i = 0$ or $\hat{L}_i = 3$ respectively. In case an error in Y_C^i is not caught by $\mathcal{H}_C^{(T)}$ and the auxiliary information $A_{L_i}^i$, then it must be an error in the table $A_{3-L_i}^i$. So, in case an error is present in A_1^i or A_2^i only, then the simulator sets $\hat{L}_i = 1$ or $\hat{L}_i = 2$ respectively. If an error is present in both A_1^i and A_2^i, then the simulator sets $\hat{L}_i = 3$. For i such that $\hat{L}_i = 1$ or 2, $\psi(i, x)$ is defined so that the output $z_i = \psi(i, x)$ is as produced by Receiver with input x. □

D From security with WDT-abort to full security

In this section, we discuss general methods for converting any NISC scheme satisfying security with *wire disjunction triggered* (WDT) abort into an NISC with full security, based on semi-honest secure MPC protocols. Our transformation formalizes and generalizes such a transformation that was given in [25, 26] (and our intuition below follows their intuition), in the context of constructing stateful hardware circuits that remain private even when an adversary can tamper with the values of wires. We note that the construction of [25] also had to deal with multiple other issues that do not concern us, which added complexity to their solution. Outlined below, our solution can be seen as a simplification of their construction.

The benefit of the transformation outlined in this section, over the simple majority-based approach discussed earlier, is the potential for greater efficiency. We will first formalize the encoding notion that we use to deal with WDT attacks, then we present an outline of our general transformation, and then show how to invoke this transformation using known semi-honest MPC protocols from the literature to obtain higher levels of efficiency.

Our transformation is captured by means of a new encoding, that we now define:

Definition 2. ***(WDT-resilient encoding)*** *A randomized circuit family C' together with an efficient randomized encoding algorithm family Enc and an efficient deterministic decoding algorithm family Dec is a* WDT-resilient encoding *of a circuit C that takes two inputs if the following properties hold:*[14]

(Correctness) For all (x, y) in the domain of C, we have that

$$\Pr[Dec(C'(Enc(x), y)) = C(x, y)] = 1$$

(Malicious Receiver Security) There exists a randomized efficient machine $RecSim$ such that for every x' in the range of Enc (but not necessarily in the image of Enc), there exists x in the domain of Enc such that for every y such that (x, y) is in the domain of C, the output distribution of $RecSim(x', C(x, y))$ is identical to the distribution $C'(x', y)$.

(WDT-Malicious Sender Security) For any set S of wires in C' or their negations, let $Disj_S[C'(Enc(x), y)]$ to be the event that the disjunction of the values of the wires (or negations) specified by S, when the input of C' is $(Enc(x), y)$, is satisfied. The probability space is over the random gates of C' and the randomness used by Enc. For any such S, and for all x_1, x_2, and y such that (x_1, y) and (x_2, y) are in the domain of C, we have:

$$|\Pr[Disj_S[C'(Enc(x_1), y)]] - \Pr[Disj_S[C'(Enc(x_2), y)]]| = 2^{-\Omega(\kappa)}.$$

Outline of Transformation. Recall that our goal is to provide security against a malicious sender who can specify an abort condition as a disjunction of wire values over the circuit. We want to make such aborts independent of the receiver's actual input. At a high level, the main idea is to transform the given circuit into an equivalent randomized circuit that has a kind of "t-wise independence" property on the values of all its wires. If this were really possible, it would solve the problem of wire-value disjunction abort, since one of two cases would occur:

1. If the adversary specifies a disjunction of $k \leq t$ wires, then by t-wise independence, the probability that the disjunction would be satisfied would be $1 - 2^{-k}$, independent of the receiver's input.
2. On the other hand, if the adversary specifies a disjunction of $k > t$ wires, then by t-wise independence, the probability that this disjunction would be satisfied would be at least $1 - 2^{-t}$, which is overwhelmingly close to 1 if t is a statistical security parameter. So the adversary would cause an abort with overwhelming probability, and it wouldn't matter what the receiver's input is.

The problem is that such t-wise independence is obviously impossible for wires within circuits since, for example, the wires incident to any particular gate could not satisfy it. So instead we aim for a relaxed condition that we call *operative t-wise independence*, where all wires in the circuit can be "colored" one of several different colors with the following properties:

[14] The entire tuple (C', Enc, Dec) is parameterized by a statistical security parameter 1^κ, which is omitted here for simplicity of notation. Note also that this definition defines the notion of a WDT-resilient encoding. In applications, we will require that there is an efficient deterministic procedure that takes as input C and 1^κ and outputs a tuple (C', Enc, Dec) from such a family.

- For any set S of wires, where there are most t different colors represented in S, it is possible to simulate the values of all wires in S independently of the receiver's input.
  This property would allow us to preserve analysis (1) above, albeit replacing the probability $1-2^{-k}$ above with some other value, but still a value that would depend *only* on the actual wires chosen by the adversary, and not the receiver's input.
- Every individual wire w has at least some constant probability c of being both 0 or 1, even when conditioned on the values of all wires that have a different color than w.
  This property would allow us to preserve analysis (2) above, albeit replacing $1-2^{-t}$ with $1-c^{-t}$ where c is the constant probability mentioned here.

Achieving the first condition of operative t-wise independence is made possible by "implementing" a t-private MPC protocol (with security only required against semi-honest parties) within a circuit [26], where the parties from the protocol will correspond to the colors outlined above: more precisely, if a wire corresponds to a value that is computed by a party P_i in the MPC protocol, this wire is colored with color i. The simulation requirement of the MPC protocol now directly implies the first condition of operative t-wise independence. We shall call the result of this step the "MPC circuit".

To achieve the second condition, we further "re-randomize" every gate of the MPC circuit to ensure that there is always new entropy in every wire corresponding to a new color [25, 26]. This step can be done in a constant-overhead manner by, for example, applying a variant of the 1-private semi-honest GMW [16] protocol to each gate of the MPC circuit.

An explicit implementation of this second transformation is as follows: for each player P_i we define a pair of mask bits r_1, r_2. Every wire assigned to player P_i (and thus colored with color i) will be split into two wires, one masked by r_1 and one masked by r_2 (the same applies to the input and output wires). Now each internal NAND gate is replaced by a gadget which maps $(a+r_1, a+r_2, b+r_1, b+r_2)$ to $(ab+1+r_1, ab+1+r_2)$. The only requirement from such a gadget is that each internal gate computes a function of two bits only, and that each wire should be "unpredictable" in the sense that for any fixed a, b, the value of each wire is sensitive to the choice of r_1, r_2.

Concretely, corresponding to a NAND gate in the MPC circuit, let $v_0 = a + r_1$ and $v_1 = b + r_2$ be the masked inputs. Now, we compute:

$$\begin{aligned}
&v_2 = v_0 v_1 = ab + ar_2 + br_1 + r_1 r_2 \text{ (the last term ensures unpredictability)}\\
&v_3 = v_0 r_2 = ar_2 + r_1 r_2\\
&v_4 = v_1 r_1 = br_1 + r_1 r_2\\
&v_5 = v_2 + r_2 = ab + ar_2 + br_1 + r_2 + r_1 r_2\\
&v_6 = v_5 + v_3 = ab + br_1 + r_2\\
&v_7 = v_6 + v_4 = ab + r_2 + r_1 r_2\\
&v_8 = r_1 r_2 + 1\\
&v_9 = v_7 + v_8 = ab + 1 + r_2 \text{ (second masked output)}
\end{aligned}$$

One can generate $ab + 1 + r_1$ (first masked output) similarly.

We note that the encoding and decoding of the receiver's input (the Enc and Dec required by the definition above) would split the receiver's input into shares that would be effective inputs to each of the MPC parties above. This is accomplished by using standard Reed-Solomon encodings, for example, to achieve t-private secret sharing. Note that this can be further optimized by using share packing to achieve a polylogarithmic overhead.

Results. We will obtain our desired result based on the existence of highly efficient MPC protocols, as captured by the following theorem from [10]:

Theorem 10 ([10]). *Let C be any circuit taking m inputs. Then there exists a perfectly secure t-private MPC protocol for C over m parties such that the total computation of all parties in the protocol is bounded by $|C| \cdot poly(\log|C|, \log m) + depth(C) \cdot poly(m, \log|C|)$, where $t = \Theta(m)$.*

Based on the outline above and following the proofs in [25, 26] we have the following theorem:

Theorem 11. *For any two-input circuit C, there exists a WDT-encoding of C such that $|C'| = |C| \cdot poly(\log|C|, \log\kappa) + depth(C) \cdot poly(\kappa, \log|C|)$, where κ is a statistical security parameter.*

Combining this with the results of previous sections, we obtain the following theorem:

Theorem 12. *For any function $f : X \times Y \to Z$ that has a polynomial sized circuit C, there is an NISC/OT scheme for $\mathcal{F}_f$ that makes a black-box use of a PRG, invoking the PRG $|C| \cdot poly(\log|C|, \log\kappa) + depth(C) \cdot poly(\kappa, \log|C|)$ times.*