APPEARED IN STOC 2008: 433-442

Cryptography with Constant Computational Overhead

Yuval Ishai[*] [†] Eyal Kushilevitz[‡] [†] Rafail Ostrovsky[§] [†] Amit Sahai[¶] [†]

Abstract

Current constructions of cryptographic primitives typically involve a large multiplicative computational overhead that grows with the desired level of security. We explore the possibility of implementing basic cryptographic primitives, such as encryption, authentication, signatures, and secure two-party computation, while incurring only a *constant* computational overhead compared to insecure implementations of the same tasks. Here we make the usual security requirement that the advantage of any polynomial-time attacker must be negligible in the input length.

We obtain affirmative answers to this question for most central cryptographic primitives under plausible, albeit sometimes nonstandard, intractability assumptions.

- We start by showing that pairwise-independent hash functions can be computed by linear-size circuits, disproving a conjecture of Mansour, Nisan, and Tiwari (STOC 1990). This construction does not rely on any unproven assumptions and is of independent interest. Our hash functions can be used to construct message authentication schemes with constant overhead from any one-way function.
- Under an intractability assumption that generalizes a previous assumption of Alekhnovich (FOCS 2003), we get (public and private key) encryption schemes with constant overhead. Using an exponentially strong version of the previous assumption, we get signature schemes of similar complexity.
- Assuming the existence of pseudorandom generators in NC^0 with polynomial stretch together with the existence of an (arbitrary) oblivious transfer protocol, we get similar results for the seemingly very complex task of secure two-party computation.

  More concretely, we get general protocols for secure two-party computation in the semi-honest model in which the two parties can be implemented by circuits whose size is a constant multiple of the size s of the circuit to be evaluated. In the malicious model, we get protocols whose *communication complexity* is a constant multiple of s and whose computational complexity is slightly super-linear in s. For natural relaxations of security in the malicious model that are still meaningful in practice, we can also keep the computational complexity linear in s. These results extend to the case of a constant number of parties, where an arbitrary subset of the parties can be corrupted.

  Our protocols rely on non-black-box techniques, and suggest the intriguing possibility that the ultimate efficiency in this area of cryptography can be obtained via such techniques.

Categories and Subject Descriptors: F.2 [Theory of Computation]: Analysis of Algorithms and Problem Complexity

General Terms: Theory, Security.

Keywords: Constant computational overhead, Cryptography, Universal hashing.

[*]Computer Science Dept. Technion, Haifa, Israel and UCLA, Los Angeles, CA. Email: yuvali@cs.technion.ac.il Supported in part by BSF grant 2004361, ISF grant 1310/06, and NSF grants 0430254, 0456717, 0627781

[†]Research supported in part by Stealth Software Technologies.

[‡]Computer Science Dept. Technion, Haifa, Israel Email: eyalk@cs.technion.ac.il Supported in part by BSF grant 2002354 and ISF grant 1310/06.

[§]Department of Computer Science and Department of Mathematics, UCLA, Los Angeles, CA, Email: rafail@cs.ucla.edu Supported in part by BSF grant 2002354, IBM Faculty Award, Xerox Innovation Group Award, NSF grants 0430254, 0716835, 0716389 and U.C. MICRO grant.

[¶]Computer Science Dept. UCLA, Los Angeles, CA, Email: sahai@cs.ucla.edu Supported in part by BSF grant 2004361, NSF Cybertrust grants 0456717, 0627781, a Cyber-TA grant, an equipment grant from Intel, an Okawa Research award, and an Alfred P. Sloan Research Fellowship.

1 Introduction

Current constructions of cryptographic primitives involve a large multiplicative computational overhead that grows with the desired level of security. In this paper we consider the possibility of implementing cryptographic primitives with a *constant computational overhead* that does not grow with the level of security. This question is open even in the case of basic cryptographic primitives such as encryption, authentication or digital signatures, let alone in the case of more sophisticated tasks such as secure two-party computation.

Formulating the above question a bit more precisely, our goal is to obtain cryptographic protocols that can be implemented by circuits whose size is a constant multiple of the size of circuits that implement the given task with no security at all. In the case of the basic primitives mentioned above, the latter quantity is simply the length of the input to be encrypted, decrypted, authenticated, signed, or verified. In the case of secure computation, this quantity refers to the size of a circuit implementing the function that we want to compute securely. In all cases, we require the advantage of polynomial-time adversaries attacking the protocol to be negligible in the size of the inputs. Thus, security scales favorably with the input size. An affirmative answer to the above question can be interpreted as saying that, *in an amortized sense*, cryptographic security comes "essentially for free".

Our work was inspired by the recent line of work on cryptography in NC^0 by Applebaum et al. [3, 4, 5, 6] (see also [27, 19, 38]). In NC^0 circuits each bit of the output depends on just a constant number of input bits. In particular, the size of such circuits is linear in the length of the output. Unfortunately, in most previous constructions of cryptographic primitives in NC^0 (with one notable exception which will be discussed later) the length of the output is superlinear in the length of the input. Indeed, the general approach taken in [3, 4] for constructing NC^0 primitives is to start with a standard (non-NC^0) implementation of the primitive and then apply a transformation that reduces the circuit depth but can only increase its total size.

Cryptography in NC^0 also has some *inherent* limitations. For instance, it is impossible to get a pseudorandom generator (PRG) [12, 46] with output locality d that stretches an n-bit seed by more than n^d bits. More significantly, several important cryptographic tasks, such as decryption of ciphertexts, verification of signatures and MACs, and computing pseudorandom functions, are provably impossible to implement in NC^0. These limitations will not apply in our case, since we focus on linear circuit *size*.

The notable exception to the above state of affairs is a PRG with linear stretch in NC^0 (namely, ones that expand a seed of n bits to a pseudorandom string of length cn for some constant $c > 1$), which was constructed in [5] based on an intractability assumption of Alekhnovich [1]. Our initial observation is that (1) PRGs with linear-size circuits and linear stretch can be used to obtain PRGs of an arbitrary polynomial stretch whose circuit size is linear in the length of the output; and (2) the former PRGs are implied by linear-stretch PRGs in NC^0. In turn, using a standard hybrid encryption technique (cf. [29]), linear-size PRGs with a big (say, quadratic) stretch can be used to encrypt long messages while incurring only a constant amortized overhead over their size. Namely, first encrypt a short seed to a PRG (say, using an encryption scheme with quadratic circuit size), and then mask the message with the PRG applied to the seed.

The above observation suggests that cryptography with constant overhead may be feasible in the context of encryption, but leaves this possibility for other central primitives completely open. It is important to stress that all standard constructions of cryptographic primitives we are aware of fail to achieve the constant-overhead requirement. Indeed, those based on number-theoretic or algebraic assumptions require performing operations that are not known to be implementable in linear time, such as modular multiplication of integers whose size grows with the security parameter. Even the common *heuristics* for designing symmetric encryption schemes and hash functions, when generalized to provide an arbitrary level of security, fail to meet our goal.

1.1 Our Contribution

We obtain affirmative answers to the above question for most central cryptographic primitives under plausible, albeit sometimes nonstandard, intractability assumptions. In particular, we obtain results for the following primitives.

Universal Hashing. We start by presenting an *unconditional* construction of a family of pairwise-independent hash functions, mapping n bits to n bits, that can be computed by circuits of size $O(n)$. This refutes an 18-year-old conjecture of Mansour, Nisan, and Tiwari [37, Conjecture 15] that such functions require circuits of size $\Omega(n \log n)$. In addition to serving as a useful building block in subsequent cryptographic constructions, pairwise-independent (or universal [17]) hash functions have numerous other applications in algorithms and complexity theory. Our construction is conceptually simple and consists of the following three steps:

1. Expansion: The original input is expanded using a good linear-size encodable error correcting code. Here we use the codes of Guruswami and Indyk [31] that have a big relative distance (say 90%) while still requiring only a constant-size alphabet.

2. Randomization: An independent constant-size pairwise independent hash function is applied to each position (symbol) of the encoding. At the end of this step, the output on one fixed input has a lot of entropy (say 90%) even when conditioned on the output on another fixed input.

3. Shrinking: The output of the previous step is shrunk back to the original input size by applying a good deterministic extractor for bit-fixing sources. We observe that a linear-size implementation of such extractors can be obtained by transposing the encoding function of a good linear-size encodable error correcting code. Indeed, for linear mappings computed via linear operations (e.g., XOR gates) there is a circuit of the same size computing the transposed mapping [13] (essentially reversing the computation while interchanging XOR and fan-out operations). Thus, to implement this final step we can use the linear-size encodable binary codes of Spielman [45] (see also [20]).

Encryption. As discussed above, our initial observation is that the assumption of Alekhnovich [1] can be used to obtain constant-overhead PRGs and semantically secure (private or public key) encryption. Roughly speaking, Alekhnovich's assumption is related to the hardness of maximum-likelihood decoding for linear codes computed by NC^0 functions whose incidence graph is expanding. In our setting, this assumption can be relaxed in several ways. First, our constant-overhead implementation of extractors for bit-fixing sources allows us to rely on a non-explicit version of Alekhnovich's assumption by allowing probabilistic constructions of codes that fail with a non-negligible probability. We can also use arbitrary linear-size encodable codes rather than NC^0 codes.

MACs, PRFs, and signatures. The construction of pairwise independent hash functions described above immediately gives rise to unconditionally-secure *one-time* MACs (private-key signatures) in which the length of the key is linear in the message length. In fact, only the first two steps of the construction described above suffice for this purpose. One can also obtain a constant-overhead pseudorandom function (PRF) [28] that shrinks an n-bit input to an n^ϵ-bit output (for some constant $\epsilon > 0$) from any PRF, by first shrinking the input via a constant-overhead hash function and then applying the given PRF. This yields standard (computationally secure) MACs with constant overhead based on any one-way function. Combined with any constant-overhead encryption, these MACs can be used to implement constant-overhead CCA-secure encryption [21]. Finally, under an exponentially strong version of the assumption we use for encryption, we can combine the technique of Naor and Yung [41] with our hash function to yield universal one-way hash functions and signatures with constant overhead.

We stress that in contrast to the situation in the context of cryptography in NC^0, in the above applications we get efficient implementations for both sides: both for encrypting and decrypting, both for signing and verifying.

Secure computation. Assuming the existence of polynomial-stretch PRGs in NC^0 (namely ones that stretch n bits to n^c bits for some $c > 1$), together with the existence of an (arbitrary) oblivious transfer protocol, we get similar results for the seemingly very complex task of secure two-party computation [47, 30]. We point out constructions from the literature that can be conjectured to satisfy the former assumption; however, these constructions are not fully explicit (see Remark 5.7).

More concretely, under the above assumption we get general protocols for secure two-party computation in the semi-honest model (where the parties are "honest but curious") in which the two parties can be implemented by circuits whose size is a constant multiple of the size s of the circuit to be evaluated. Our construction employs a variant of Beaver's non-black-box technique for extending OTs [9], namely generating many OTs from few OTs, combined with polynomial-stretch PRGs in NC^0 and randomized encoding techniques (e.g., an information-theoretic version of Yao garbled circuit technique) [47, 35, 33, 3]. We note that Beaver's protocol can employ an arbitrary PRG, but it needs to perform cryptographic operations for every gate of the PRG. Thus, our result cannot be obtained by directly combining a linear-size PRG with Beaver's protocol and we need to rely on the NC^0 structure of the PRG.

In the malicious model, we get protocols whose *communication complexity* is a constant multiple of s and whose computational complexity is slightly super-linear in s. (In fact, an arbitrary super-linear function will do.) For natural relaxations of security in the malicious model that are still meaningful in practice, such as security against *covert* adversaries that are unwilling to be caught cheating [7], we can also keep the computational complexity linear in s. All the above results extend from two parties to any constant number of parties, where an arbitrary subset of the parties can be corrupted. We leave open the question of obtaining constant computational overhead for zero-knowledge proofs and secure two-party computation in the (strict) malicious model.

Our secure computation protocols suggest the intriguing possibility that, in contrast to common belief, the ultimate efficiency in this area of cryptography can be obtained via *non-black-box* techniques.

On the efficiency of our constructions. Recall that we use the standard convention of viewing the input length n also as a security parameter, requiring that any attacker that runs in time $\mathrm{poly}(n)$ can only gain a negligible advantage in n. It is important to note that most of our constructions involve some (typically moderate) polynomial security loss that is hidden by this convention. That is, if the constructions are broken on inputs of length n then the underlying intractability assumptions are violated on inputs of length n^ϵ for some constant $\epsilon > 0$. Viewed differently (and informally), when decoupling the input length n from the cryptographic security parameter k, our constructions involve (in addition to the constant multiplicative overhead) an *additive* overhead that depends polynomially on k but is independent of n. This means that the constant overhead becomes effective only for inputs that are sufficiently long compared to the desired level of security. Note, however, that in our motivating scenario of large-scale cryptographic computations one would typically be better off with constructions that involve (say) a quadratic security loss and a small multiplicative overhead than constructions that involve no security loss and a bigger overhead.

1.2 Related Work

Linear-time encodable error-correction codes. A similar endeavor to ours was already made in the domain of coding theory. A seminal result of Spielman [45] (building on [43]) shows how to construct error-correcting codes that have a good minimal distance and can be encoded by linear-size circuits and decoded in linear time on a RAM machine. As noted above, some of our constructions rely on this type of codes; however, they do not require the efficient decoding feature, and thus can be fully implemented by linear-size circuits. Furthermore, this very same decoding feature seems to make these codes bad candidates for our constructions of PRGs and encryption schemes with constant overhead.

Smith [44] recently suggested a heuristic construction of linear-time block ciphers using linear-time encodable codes with a good distance. However, the conjectured security of this construction does not seem to follow

from any natural or easily stated one-wayness or indistinguishability assumption concerning these codes. Smith also does not deal with the other cryptographic tasks that we consider.

Efficient secure computation. The efficiency of secure computation [47, 30] has been the subject of a very large body of work. Still, all previous protocols for secure two-party computation that we are aware of, even in the semi-honest model and even protocols that are tailored for specific natural tasks, involve a large multiplicative overhead which grows with the security parameter compared to an insecure implementation of the same task. In particular, all general protocols for secure two-party computation require performing cryptographic computations for every gate in a circuit being evaluated. This overhead remains even when considering *communication* alone, rather than computation. To make things worse, in this area of cryptography it is difficult to come up with heuristic constructions; in particular, the above discussion is relevant even when considering protocols that were suggested without any proof of security (but are not easily "broken"). That is, we are not aware of even a heuristic suggestion in the literature for secure two-party computation with constant overhead.

2 Preliminaries

2.1 Computational Models

We measure computational work by circuit size, namely counting the number of gates in a boolean circuit with constant fan-in and fan-out. We allow circuits to be randomized, in which case there are specially designated gates that contain random bits which are picked freshly in each invocation. The term "linear size" circuits refers to circuits whose size is linear in the sum of their input size and output size.

The general convention we make when defining *constant overhead* cryptographic primitives is that the circuit size of the implementation needs to be bounded by a constant multiple of the circuit complexity of implementing the same task without security (e.g., message length in case of encryption, output length in case of PRGs, and the functionality's circuit size in the case of secure computation), regardless of the desired level of security. To this end we use the latter size measure as a cryptographic security parameter, requiring the advantage of a polynomial-time adversary attacking the scheme to be negligible in this size.[1] Thus, on the one hand security is only required to hold for large inputs, but on the other hand it gets better with the input size.

We often refer (either explicitly or implicitly) to *collections* of cryptographic primitives. By this we mean that there is a probabilistic polynomial time setup algorithm that, on size parameter 1^n, may probabilistically generate the circuit or circuits implementing the primitive. This setup only needs to be run *once and for all* (e.g., for generating cryptographic hardware) and its random coins, and in particular the circuits it produces, are known to the adversary.

2.2 PRGs in NC^0

Constructions of PRGs with sublinear stretch in NC^0 can be obtained under standard assumptions [3]. Below we sketch a previous construction of a *linear-stretch* PRG from [5] that relies on a specific assumption.

Let $m = m(n) > n$ be an output length parameter, let $\ell = \ell(n)$ be a locality parameter (typically a constant), and let $0 < \mu < 1/2$ be a noise parameter. Let $\mathcal{M}_{m,n,\ell}$ be the set of all $m \times n$ matrices over $\mathbb{F}_2$ in which each row contains ℓ ones. For a matrix $M \in \mathcal{M}_{m,n,\ell}$ denote by $D_\mu(M)$ the distribution of the random m-bit vector $Mx + e$, where $x \leftarrow U_n$ and $e \in \{0,1\}^m$ is a random error vector in which each entry is chosen to be 1 with probability μ (independently of other entries), and arithmetic is over $\mathbb{F}_2$. The following assumption,

[1] Note that for basic primitives such as one-way functions it is trivial to meet these efficiency and security requirements by simply applying an inefficient construction to a small portion of the input and copying the remainder of the input to the output. However, this padding approach cannot be applied in the context of natural cryptographic *applications* such as those considered in this work.

presented in [5], is implied by a conjecture suggested by Alekhnovich [1, Conjecture 1]. [2]

Assumption 2.1 *For any $m(n) = O(n)$, and any constant $0 < \mu < 1/2$, there exists a positive integer ℓ, and an (explicit) infinite family of matrices $\{M_n\}_{n\in\mathbb{N}}$, $M_n \in \mathcal{M}_{m(n),n,\ell}$, such that*

$$D_\mu(M_n) \overset{c}{\approx} D_{\mu+m(n)^{-1}}(M_n)$$

It is shown in particular that this implies that $D_\mu(M_n) \overset{c}{\approx} U_{m(n)}$.

The difficulty of directly obtaining an efficient PRG from the above assumption is that generating the noise vector using a minimal amount of randomness is a computationally nontrivial task. We now describe the PRG construction of [5]. This is an NC^0 PRG that has linear stretch. Let t and ℓ be positive integers, and $c, k > 1$ be real numbers that will affect the stretch factor. Let $m = kn$ and let $\{M_n \in \mathcal{M}_{n,m,\ell}\}$ be an infinite family of matrices as above. Let $g : \{0,1\}^{tm/c} \to \{0,1\}^{tm}$ be the NC^0 ϵ-biased generator of [38]. Define

$$G_n(x, y, r) = (M_n x + E(y), g(r) + y),$$

where $x \in \{0,1\}^n, y \in \{0,1\}^{t\cdot m}, r \in \{0,1\}^{t\cdot m/c}, E(y) = \left(\prod_{j=1}^{t} y_{t\cdot(i-1)+j}\right)_{i=1}^{m}$. Thus,

$$G_n : \{0,1\}^{n+tm+\frac{tm}{c}} \to \{0,1\}^{m+tm}.$$

It is easy to observe that G_n is an NC^0 function. It is shown in [5] that, based on Assumption 2.1 and with an appropriate choice of parameters, the function G_n is a linear-stretch PRG.

[2] The two versions of the assumption are very similar. The main difference is that in [1] the noise vector e is a random vector of weight exactly $\lceil \mu m \rceil$, while in [5] the entries of the noise vector are chosen to be 1 independently with probability μ.

2.3 Randomized Encoding of Functions

Our constructions for secure computation rely on a variant of the notion of randomized encoding of functions from [3] which abstracts the combinatorial object that underlies OT-based two-party protocols such as Yao's protocol [47] and Kilian's protocol [35]. Let $f : \{0,1\}^n \to \{0,1\}^m$ be a function. We say that $\hat{f}(x,r)$ is a *randomized encoding* of f (or encoding of f for short) if the following requirements hold. (1) Correctness: given $\hat{f}(x,r)$ it is possible to efficiently recover (or "decode") $f(x)$. (2) Privacy: Given $f(x)$, it is possible to efficiently sample from the distribution of $\hat{f}(x,r)$ induced by a uniform choice of r. We say that an encoding is *decomposable* if every output bit of $\hat{f}$ depends on at most a single bit of x (but can depend arbitrarily on r). This is incomparable to the notion of NC^0 encoding employed in [3] but the constructions of NC^0 encodings from [3] also enjoy the decomposability property. Other constructions of decomposable encodings are implicit in the secure computation literature (e.g., [35, 23, 33]).

Observe that a non-boolean function f can be encoded by concatenated independent randomized encodings of the functions defined by its output bits. Thus we have the following:

Lemma 2.2 *Let $f : \{0,1\}^n \to \{0,1\}^m$ be a function in NC^0. Then f admits a decomposable randomized encoding $\hat{f}(x,r)$ in which both the encoding and the decoding operations can be performed by circuits of size $O(m)$.*

We are not aware of any other "interesting" classes of functions which enjoy this linear-size encoding property that will be crucial for our purposes.

3 Linear-Size Circuits for Universal Hashing

In this section, we will show how to construct universal (pairwise-independent) hash functions that can be evaluated by linear-size circuits. This refutes a conjecture of Mansour, Nisan, and Tiwari [37], that universal hash functions mapping n bits to n bits require circuits of size $\Omega(n \log n)$. Along the way, we will also construct perfect exposure-resilient functions [15] (also known as extractors for bit-fixing sources [18]) that have linear circuit size. Both these objects will be useful tools for us later. And of course, universal hash functions and exposure-resilient functions have many algorithmic and cryptographic applications aside from the ones we use them for.

3.1 Linear-Size Circuits for Exposure Resilient Functions

We first introduce some notation. Let $L \subset \{1,\ldots,n\}$, and let $y \in \{0,1\}^n$. By $[y]_L$ we denote y restricted to the bits of y corresponding to locations in L (we will also use this notation with a general alphabet replacing bits). Then, following [15], we define a perfect ℓ-ERF as follows:

Definition 3.1 *A function $f : \{0,1\}^n \to \{0,1\}^m$ is a (perfect) ℓ-ERF (exposure resilient function) if for any $L \subset \{1,\ldots,n\}$ of size $|L| = n - \ell$, and choosing $r \in \{0,1\}^n$ and $R \in \{0,1\}^m$ uniformly and independently at random, we have that the following distributions are identical:*

$$([r]_L, f(r)) \equiv ([r]_L, R)$$

Intuitively, this definition guarantees that $f(r)$ will be uniformly random even if only ℓ bits of r are chosen at random. We obtain the following theorem:

Theorem 3.2 *There exists an infinite family of perfect ℓ-ERF's that are computable by linear-size circuits where $\ell = \Theta(n)$ and $m = \Theta(n)$.*

Proof: In [18] it is shown that if G is the generator matrix for a (linear) error-correcting code mapping m bits to n bits, with minimum distance d, then the transpose matrix G^t mapping n bits to m bits, will be a perfect ℓ-ERF where $\ell = n - d + 1$.

Furthermore, Spielman [45] (see also [20]) constructs an asymptotically good linear code with generator matrix G that maps m bits to n bits with distance d, such that $d = \Theta(m)$ and $n = \Theta(m)$, with the property that the encoding can be computed via a circuit C of linear size, where the circuit uses only fan-out gates and XOR gates (we call such circuits *XOR-fanout* circuits).

To obtain the theorem, we make use of the following simple observation (originally due to [13]). Suppose that you have an XOR-fanout circuit C that computes G. Then to make a circuit C' that computes G^t is easy: simply "reverse" the direction of all the wires in C, and replace all XOR gates with fan-out gates, and all fan-out gates with XOR gates. Such a circuit would also be of linear size, and thus we would obtain a perfect ℓ-ERF with $\ell = \Theta(n)$ computable with a linear-size circuit.

To see that this process works, we give the following more general argument: Let C be a circuit computing the action of (left-) multiplication by a matrix G which maps m bits to n bits, where each individual "gate" in C is an arbitrary constant-size linear transformation. Let N be the total number of wires in C. Then we can define E to be the matrix that maps m bits to N bits by appending $(N - m)$ zero's to the end of its input. And we can define R to be the matrix that maps N bits to n bits by projecting to the last n coordinates of the input. (Observe that E^t is a projection from N bits to m bits, and that R^t appends $(N - n)$ zero's to the start of its input.) Thus, using the circuit C we can decompose $G = R \cdot G_1 \cdot G_2 \cdots G_q \cdot E$, where each G_i is an N by N matrix representing the action of the i'th "gate" from C, and acting as the identity on all wires not involved in the i'th gate. Then, trivially, we have that $G^t = E^t \cdot G_q^t \cdot G_{q-1}^t \cdots G_1^t \cdot R^t$. Here, each G_i^t differs from the identity matrix in only a constant number of places. Thus the action of each G_i^t can be carried out by a constant number of gates, and thus we can construct a circuit C' for computing G^t such that the size of C' is only a constant factor bigger than C. □

3.2 Construction of Universal Hash Function Family

In this section we will construct a family of pairwise-independent hash functions (mapping n bits to n bits) such that there exist linear-size circuits implementing each hash function.

Theorem 3.3 *There exist universal families of hash functions $\mathcal{H}_n$ for an infinite number of values n, such that each $h \in \mathcal{H}_n$ can be implemented using a circuit of size $O(n)$. Alternatively, there exists a single circuit $Eval$ of size $O(n)$ such that for any $h \in \mathcal{H}_n$, we have that $Eval(h, x) = h(x)$.*

Intuition. The intuition behind our construction is simple. The high level idea is to use a linear number of constant-size universal hash functions to implement a large universal hash function. To accomplish this, we first use a linear-size encodable (but very good) error-correcting code C to encode the input. Then, we apply different constant-size universal hash functions h_i on every symbol of the encoding. The key observation is that if $x \neq y$, then $C(x)$ and $C(y)$ will differ in many places. For each place where $C(x)_i = C(y)_i$, we will obviously have that $h_i(C(x)_i) = h_i(C(y)_i)$; but in each place where $C(x)_i \neq C(y)_i$, we will have that $h_i(C(x)_i)$ is distributed uniformly at random and *independently* of $h_i(C(y)_i)$ which is also uniform, by the pairwise independence property of each h_i. Thus, finally, we can apply the linear-size ERF f from the previous section, which will lead to uniform and independent outputs for $f(\vec{h}(C(x)))$ and $f(\vec{h}(C(y)))$.

Proof: We now describe the steps of our construction more formally. The code C will be a linear-size encodable code of Guruswami and Indyk [31], which is a simple combination of the code of Spielman [45] and the ABNNR code of Alon et al. [2], that achieves relative distance $(1 - \epsilon)$ for any ϵ with symbols of size $\mu = O(1/\epsilon^2)$ bits. C has rate $r = O(1/\epsilon^2)$, and so it maps n bits to $m = rn = O(n/\epsilon^2)$ bits. This encoding

function can be implemented by a linear-size circuit. Note that we will choose the exact value of the constant ϵ later.

For each of the $O(n)$ symbols of $C(x)$, we choose a constant-size universal hash function h_i mapping μ bits to μ bits, and apply it to the encoded codeword symbols. We will denote this output $\vec{h}(C(x))$, which will also be m bits in length. Obviously, this step is also implementable by a linear-size circuit. Note that the choice of these hash functions is the only choice involved in the construction of our hash family. Therefore, this choice can be hard-wired into the circuit, or it can be provided as input to one circuit $Eval$ that evaluates any hash function in our family. In both cases, the size will be $O(n)$.

Finally, we will apply the ERF f constructed in the last section. f will map m bits to $n' = c_1 m$ bits, and have $\ell = (1 - c_2)m$, where c_1 and c_2 are constants arising from Spielman's code. We need to ensure that $c_2 > \epsilon$ so that the ERF f will function properly, and that $c_1 > r = O(1/\epsilon^2)$ so that the hash function will output at least n bits. Since we can choose ϵ to be any constant, we can ensure that this holds. This completes the description of the construction.

To prove that our family is universal, we can consider any two n-bit inputs $x \neq y$. Let L denote the places where $C(x)_i = C(y)_i$, and $\bar{L}$ denote the places where $C(x)_i \neq C(y)_i$. By the distance property of the code C, we know that L has at most an ϵ fraction of the m/μ positions. For $i \in \bar{L}$, we know that when these h_i are chosen at random, that each $h_i(C(x)_i)$ and $h_i(C(y)_i)$ will be distributed uniformly and independently of each other. That is, we know that $[\vec{h}(C(x))]_{\bar{L}}$ and $[\vec{h}(C(y))]_{\bar{L}}$ are distributed uniformly and independently of each other. Let us fix some choice for $\{h_i\}_{i \in L}$. Conditioned on this choice, since $c_2 > \epsilon$, by the definition of ERF, we have that $f(\vec{h}(C(x)))$ and $f(\vec{h}(C(y)))$ are distributed uniformly, and since they are functions of independent variables, they are distributed independently. But this fact holds conditioned on every possible choice of $\{h_i\}_{i \in L}$. Therefore it holds unconditionally, and the theorem is established. □

4 Basic Cryptographic Tasks

In this section we sketch constant-overhead constructions for basic cryptographic tasks such as (public and private key) encryption, commitment, MACs, and signatures. We refer the readers to [25] for the (standard) definitions of these primitives. Most of the constructions in this section are fairly straightforward given the linear-size PRGs and the universal hash functions, and involve a standard use of these primitives for amortizing the cost of inefficient implementations that are only applied on short inputs.

As usual, the concrete complexity of the following constructions might involve an additive term that depends polynomially on the security parameter; thus, the efficiency advantage over traditional constructions may become effective only when the input (e.g., message to be encrypted or signed) is large.

4.1 Encryption and Commitment

We start by noting that given a linear-size PRG with linear stretch, the standard tree-based approach for extending the output length of PRGs yields a linear-size PRG with polynomial stretch (recall that in the context of PRGs, "linear" size always means linear in the output length).[3] Next, we note that a linear-size PRG with polynomial stretch allows us to get public key (and hence also private key) encryption schemes with constant computational overhead. This is done via a standard use of hybrid encryption: an encryption of a message m

[3] Assume first that we are given a PRG with stretch $2n$: i.e., let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a PRG, computable by a circuit of size cn. By viewing the $2n$-bit output of G as consisting of two n-bit parts (left and right) and re-applying G to each of them, we get a tree-like construction. If this is repeated for d levels we get an output of length $2^d n$. The pseudo-randomness of this construction is well known (cf. [28]) and follows from a standard hybrid argument. Let $T(m)$ be the circuit size for producing an output of length m (and assume m is of the form $m = 2^i d$) then we have $T(2n) = cn$ and $T(2^d n) = T(2^{d-1}n) + 2^{d-1}cn$ which implies that $T(2^d n) = cn \sum_{i=1}^{d-1} 2^i < c(2^d n)$. If the underlying G has smaller (but still linear) stretch, i.e. αn for a constant $\alpha < 2$, then iterate it first $1/(\alpha - 1)$ times (which is still constant and hence can be done in linear size) to obtain stretch $2n$ and then proceed as before.

of length n using a public key pk is $(E_{pk}(s), G(s) \oplus m)$, where $E(\cdot)$ is an arbitrary public-key encryption scheme that encrypts a message of length t using circuits of size $O(t^c)$, s is a random seed of length $n^{1/c}$, and G is a PRG stretching $n^{1/c}$ bits into n bits that has a circuit of size $O(n)$. A similar procedure is applied for decryption. Commitment schemes with constant overhead can be implemented in an analogous way from any linear-size PRG.

Remark 4.1 (On relaxing the assumptions.) Recall the PRG construction of [5], described in Section 2.2. The aim of this construction is to obtain a PRG in NC^0 with linear stretch. Here, our goal is only to obtain *linear-size* PRGs with linear stretch, which allows both to relax and generalize the underlying security assumptions.

Firstly, we can "hedge" our hardness bets by making use of the linear-size ERFs constructed in Section 3.1. To give an example, Alekhnovitch [1] conjectured that any good (bipartite) expander graph will make his assumption work. Rather than relying on one particular explicit construction of expanders, we can instead use random constant-degree graphs. Note that such graphs will fail to be expanders with some inverse polynomial probability. However, as long as we are confident that a large majority of the PRGs are good, we can use the linear-size perfect ERFs to deterministically extract a constant fraction of all the output bits that is guaranteed to be pseudo-random.

Secondly, since here we only require linear-size circuits and not NC^0 circuits, one could potentially use codes that are computable in linear time but not in NC^0 (just like the NC^0 codes, such codes would not necessarily have to have good distance properties).

4.2 MACs, PRFs and CCA-Secure Encryption

The existence of constant-overhead unconditionally secure *one-time* MACs (message authentication codes) follows immediately from the construction of universal hash functions in the previous section. In fact, we need not apply the final ERF in the universal hash function construction to get MACs, since there is no need for full pairwise independence. Note, however, that the key length of the MAC will be linear in the length of the message.

We can proceed to construct unrestricted, computationally secure MACs with short keys using two approaches. The first approach is to directly combine an unconditional MAC with a constant-overhead PRG that is used to generate the long keys. This yields a constant-overhead (stateful) MAC under the assumptions from Section 4.1. Below we describe an alternative approach that relies on pseudorandom functions (PRFs) to construct a linear-time stateless computational MAC from any one-way function.

Using a linear-size hash function, we can transform *any* family of PRFs into linear-size computable PRFs that output n^ϵ bits on any input of length n.[4] Suppose that a PRF family $\{f_s\}$ is such that each f can be evaluated on an n-bit input in n^c time. We will construct a new PRF family $\{f'_{h,s}\}$ where $f'_{h,s}(x) = f_s(h(x))$. Here, h is a linear-size computable pairwise-independent hash function that hashes down the input x to $n^{1/c}$ bits. The new PRF f' first applies h to shrink the input and then applies the original PRF f which can now be computed using $O(n)$ size circuits. Note that the pairwise independence of h implies that the collision probability is very low, which yields the security of the new PRF. Since the key of f' includes a description of h, its length is linear in the input length. Using standard domain extension techniques for PRFs (e.g., CBC-style chaining) one can extend the PRF f' to inputs of an arbitrary polynomial length without increasing the key length and while maintaining constant computational overhead.

We can apply the PRFs constructed above to obtain constant-overhead computationally secure MACs under the minimal assumption that one-way functions exist. We note, however, that the concrete efficiency of the PRG-based approach described above may be better than that of the more general PRF-based approach.

[4] By making use of the PRGs from Section 4.1 (which require stronger assumptions), we can also expand the output length of the PRF family to any desired polynomial length, while still maintaining computational complexity linear in the combined length of the input and output.

Finally, by utilizing any CCA-secure public key encryption [21] in a hybrid mode with an authenticated symmetric encryption, we obtain a CCA-secure public-key encryption scheme with constant overhead. The underlying assumption here is as in Section 4.1.

4.3 Signatures

We can obtain signatures with constant computational overhead by adapting the approach of Naor and Yung [41]. The main tool we use here is our linear-size universal hash function; we also make use of several standard techniques.

We now sketch our approach in two parts: First, we will show that given any one-way function that is (1) exponentially hard, (2) computable by linear-size circuits, and (3) one-to-one,[5] we can construct signatures with constant computational overhead. To see this, we first note that we can transform any such function $f : \{0,1\}^n \to \{0,1\}^m$ where $m > n$ into one which maps n' bits to $n' + o(n')$ bits, that is still exponentially hard and (almost) one-to-one. This can be done by replacing the function f with $f'(x,h) = (h, h(f(x)))$, where h is a linear-size pairwise-independent hash function mapping m bits to $n + o(n)$ bits. This transformation preserves exponential hardness and (almost) one-to-one-ness by a simple application of the Leftover Hash Lemma [32]. Now, a straightforward modification of the approach of Naor and Yung yields universal one-way hash functions (UOWHF's) that can shrink the message by a constant factor in linear time (by again making use of our linear-size universal hash functions), and therefore signature schemes with constant computational overhead.

We also note that candidates for constant-overhead exponentially strong (almost) one-to-one one-way functions can be obtained from a variety of exponential hardness assumptions, such as those arising from exponential one-way versions of a variant of Alekhnovich's assumption[6] [1] and an assumption of exponential hardness of Goldreich's candidate one-way function [27]. (We stress that in contrast to the assumption that we use for encryption, here we only require that the function be (exponentially) *one-way* rather than satisfy some indistinguishability property.) The main difficulty here is in ensuring one-to-one-ness. In the case of Alekhnovich's function, while it is not one-to-one itself, it is easy to construct variants of the function in which the maximum size of the preimage of any possible output value is $2^{\epsilon n}$ for an arbitrarily small constant ϵ. We can choose this $\epsilon = \delta/10$, where $2^{\delta n}$ is the hardness level assumed for the function. Then, we define $f'(x,h) = (h, h(x), f(x))$, where h is a linear-size universal hash function mapping n bits to $2\epsilon n$ bits. Again using the Leftover Hash Lemma, it is easy to see that f' will be (almost) one-to-one, while still retaining an exponential level of hardness, as needed. A similar analysis can be done for Goldreich's candidate one-way function.

5 Secure Computation

In this section we present protocols for secure multiparty computation in which the computational complexity of the parties (and in particular the communication complexity) is a constant multiple of the size of the circuit being computed. We refer the reader to [14, 26] for standard definitions of secure computation.

In the following we view the number of parties k as being constant. Assuming the existence of an honest majority (namely, that there are at most $t < k/2$ corrupted parties in the semi-honest model or $t < k/3$ in the malicious model), standard MPC protocols from the literature (e.g., [10]) achieve constant overhead.

[5] In fact, it will be sufficient to be *almost* one-to-one, by which we mean that only for an exponentially small fraction of inputs x will there be an $x' \neq x$ such that $f(x) = f(x')$.

[6] This assumption does not seem to be affected by recent subexponential algorithms for learning parity with noise [11]. The best known versions of these algorithms apply only to linear codes whose block length is polynomial (rather than linear) in the input length [36].

The situation is very different in the case of no honest majority and in particular in the important two-party case. In this case, despite an extensive amount of research, the best known protocols require to perform cryptographic computations for every gate in a circuit computing the function. Even protocols that are optimized for specific and highly structured tasks, such as set intersection (cf. [24]), require a significant overhead (that grows with the security parameter) compared to insecure implementations of the same tasks. This is the state of affairs even in the case of secure computation in the semi-honest model.

We begin with some high level overview of our approach. Our protocol is based on the GMW protocol [30]. The main difficulty in achieving low overhead is the need to implement the $O(s)$ oblivious transfers (OTs) required by the GMW protocol. To this end, we to reduce $O(s)$ OTs on bits to much fewer OTs on longer strings such that the total length of the strings is roughly the same (namely, still $O(s)$). The cost of the latter OTs can then be amortized via a constant-overhead symmetric encryption. The above reduction employs a linear-size decomposable randomized encoding (see Section 2) of a function defined by an underlying PRG. Here it is crucial that our PRG be in NC^0, and not just computable in linear time, as otherwise the randomized encoding is not efficient enough. It is also crucial that the PRG has polynomial (rather than just linear) stretch, in order to guarantee that the number of OTs required by our reduction is polynomially smaller than the number of OTs we need to produce, so that the cost of the (inefficient) implementations of the former OTs will be amortized away. The details follow.

5.1 The Semi-Honest Model

From here on we refer to secure two-party computation in the semi-honest model. Extension to the case of $k > 2$ parties will be discussed later in this section and extensions to the malicious model will be discussed in Section 5.2.

A first observation is that using the GMW protocol [30, 26], evaluating a circuit C of size s can be reduced to $O(s)$ invocations of an oblivious transfer protocol (OT) [42, 22, 26] on pairs of *bits* plus $O(s)$ additional work. Note that the number of rounds in this protocol (not counting the implementation of the OTs) will be proportional to the circuit depth. However, viewed as an interaction between two interlocked circuits, the combined size of the interactive circuits (again, excluding the OTs) is $O(s)$.

It remains to show how to implement the $O(s)$ OT calls using circuits of total size $O(s)$. The next observation is that it suffices to implement $O(s)$ simultaneous invocations of OT (more precisely, a secure protocol for the functionality that allows a receiver to select one bit from each of $O(s)$ pairs of bits held by a sender). This follows from the easy fact that following an invocation of OT on random inputs, an invocation of OT on arbitrary inputs can be implemented using a constant amount of work [8]. Thus, by first implementing $O(s)$ parallel OTs on random inputs, each OT invocation during the execution of the protocol only requires a constant number of gates to implement. In fact, following the initial random OT generation, the local computation performed by each party in the protocol of [30] closely resembles the computation of the original circuit C.

The above discussion is summarized by the following lemma. Let n denote a security parameter (in the final protocol, one can let $n = \sqrt{s}$). For a polynomial $p = p(n)$, we let OT^p denote the functionality of $p = p(n)$ independent bit-OTs. Namely OT^p takes p selection bits $(r_1, \ldots, r_p)$ from a receiver R and p pairs of bits $((s_1^0, s_1^1), \ldots, (s_p^0, s_p^1))$ from a sender S. The functionality gives no output to S and gives the p selected bits $s_i^{r_i}$ to R.

Lemma 5.1 *Let $C = \{C_n\}$ be a family of circuits of polynomial size $s(n)$ defining a two-party functionality $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. Then there is a two-party protocol π_f that realizes f with perfect security (in the semi-honest model) using a single oracle call to OT^p, where $p = O(s(n))$ and where each party in π_f can be implemented by a circuit of size $O(p)$.*

By standard composition theorems for secure computation [14, 26], it suffices to deal with the remaining task of securely implementing OT^p with constant overhead. Our basic approach to this problem combines a

variant of Beaver's (non-black-box) technique for extending OTs [9] with a decomposable randomized encoding. The efficiency of the latter relies crucially on the fact that the PRG is in NC^0; we are unable to exploit a general linear-size PRG with these parameters for our purposes. Our protocol will also employ an encryption scheme with constant overhead, but the existence of such a scheme is already implied by the strong assumption we make on the PRG (see Section 4.1).

We start by formalizing the PRG assumption we will use.

Assumption 5.2 (Polynomial-stretch PRG in NC^0) *There are constants $c > 1$ and d for which there exists a PRG (alternatively, PRG collection) $G : \{0,1\}^n \to \{0,1\}^{n^c}$ in which each output bit depends on at most d input bits.*

The main building block of our protocol is an $O(p)$-size, computationally secure reduction from OT^p to $\sqrt{p}$ instances of string-OT on $\sqrt{p}$ pairs of strings (α_i, β_i) whose total length is $O(p)$. This will complete our job, since using Assumption 5.2 and an arbitrary bit OT protocol π_{OT} we can obtain a string-OT protocol with a constant computational overhead: to implement string-OT on a pair of strings of length ℓ, the sender picks a pair of keys of length $\sqrt{\ell}$ for a constant-overhead symmetric encryption (implied by our assumption) and allows the receiver to pick one of them by invoking π_{OT} $\sqrt{\ell}$ times. The security parameter in these invocations is set to a sufficiently small function ℓ^ϵ so that each invocation requires circuits of size $\sqrt{\ell}$. Finally, the sender sends to the receiver the pair of strings, each encrypted with the corresponding key using a constant-overhead encryption.[7]

We stress that while the above steps involve a polynomial loss of security, the advantage of the adversary is still negligible in the size of the circuit and the overhead remains constant.

It remains to describe the $O(p)$-size, computationally secure reduction from OT^p to $\sqrt{p}$ instances of string-OT.

Lemma 5.3 *Suppose Assumption 5.2 holds. Then, for any polynomial $p = p(n)$, there is a computationally secure (in the semi-honest model) protocol π that realizes OT^p using $\sqrt{p}$ calls to string OT on pairs of strings (α_i, β_i) whose total length is $O(p)$, where each party in π can be implemented by a circuit of size $O(p)$.*

Proof: Let $G : \{0,1\}^{\sqrt{p}} \to \{0,1\}^p$ be a PRG with a constant output locality. (The existence of G is implied by Assumption 5.2; one way of obtaining such a PRG is to use a constant-depth tree of PRGs with a smaller stretch guaranteed by the assumption.) Define the following two-party functionality g. The input of R consists of a PRG seed ρ of length $\sqrt{p}$ and the input of S consists of p pairs of bits $((\sigma_1^0, \sigma_1^1), \ldots, (\sigma_p^0, \sigma_p^1))$. The output of R consists of the p bits $\sigma_i^{\rho'_i}$, where $\rho' = G(\rho)$, and S has no output. Intuitively, g can be used to implement a pseudorandom selection of one bit from each pair. We first describe a linear-size implementation of π using an oracle call to g, and then describe a linear-size implementation of a secure protocol for g using $\sqrt{p}$ string OTs.

IMPLEMENTING OT^p USING g:

- R picks $\rho \in_R \{0,1\}^{\sqrt{p}}$ and S picks
$$((\sigma_1^0, \sigma_1^1), \ldots, (\sigma_p^0, \sigma_p^1)) \in_R \{0,1\}^{2p}.$$
- R and S invoke g; let σ' denote the vector of p output bits received by R. Recall that $\sigma'_i = \sigma_i^{\rho'_i}$, where $\rho' = G(\rho)$.
- R sends to S its masked input $r' = r \oplus \rho'$.

[7] Note that the security of this string-OT protocol is only meaningful when applied to sufficiently long strings; thus, we need to pad short pairs (α_i, β_i) so that their length is at least, say, $\sqrt{p}$. Since there are only $\sqrt{p}$ such strings this padding does not increase the overall asymptotic complexity.

- S sends to R, for each $1 \leq i \leq p$ and $b \in \{0,1\}$, its masked and permuted inputs $s'^b_i = s_i^{b \oplus r'_i} \oplus \sigma_i^{b \oplus r'_i}$.
- R outputs, for each $1 \leq i \leq p$, the bit $s_i^{r_i} = s'^{\rho_i}_i \oplus \sigma'^{\rho_i}_i$.

The view of R in this protocol can be perfectly simulated. Moreover, if g were augmented to use a completely random ρ' (rather than a pseudorandom ρ' picked by R) the view of S could also be perfectly simulated. The perfect simulation is degraded to a computational simulation when ρ' is pseudorandom as in the above protocol.

It remains to implement g using $\sqrt{p}$ calls to string OT, on strings (α_i, β_i) whose total length is $O(p)$, and $O(p)$ additional work. To this end we apply an information-theoretic version of Yao's garbled circuit technique [47]. Note that g is in NC^0 (indeed, g is obtained by composing the NC^0 PRG G with an NC^0 selection function). Thus, by Lemma 2.2, the function $g(s, \rho)$ admits a decomposable randomized encoding $\hat{g}((s, \rho), r_e)$ for which both encoding and decoding can be done by circuits whose size is linear in the output size p of g. In particular, the output size of $\hat{g}$ is $O(p)$.

By the decomposability property of $\hat{g}$ we can write its output as $(\hat{g}'(s, r_g), \hat{g}_1(\rho_1, r_g), \ldots, \hat{g}_{\sqrt{p}}(\rho_{\sqrt{p}}, r_g))$. Letting ℓ_i denote the output length of $\hat{g}_i$, we have $\sum_{i=1}^{\sqrt{p}} \ell_i = O(p)$.

IMPLEMENTING g USING $\sqrt{p}$ STRING OTS OF TOTAL LENGTH $O(p)$ AND $O(p)$ ADDITIONAL WORK:

- S picks randomness r_g for $\hat{g}$ and sends $\hat{g}'(s, r_g)$ to R. It also prepares $\sqrt{p}$ pairs (α_i, β_i) where $\alpha_i = \hat{g}_i(0, r_g)$ and $\beta_i = \hat{g}_i(1, r_g)$.
- Using $\sqrt{p}$ calls to a string OT oracle, R selects for each $1 \leq i \leq \sqrt{p}$ either α_i if $\rho_i = 0$, or β_i if $\rho_i = 1$. Now R knows the entire encoded output $\hat{g}((s, \rho), r_g)$.
- R applies the ($O(p)$-size) reconstruction function of $\hat{g}$ to obtain $g(s, \rho)$.

The correctness and (perfect) security of the above protocol follow directly from the definition of a randomized encoding. □

Composing the above protocols, we get the following:

Lemma 5.4 *Suppose Assumption 5.2 holds and a (standard) OT protocol exists. Then, for any polynomial $p(n)$, there exists a secure protocol for $OT^{p(n)}$ in the semi-honest model in which each party can be implemented by a circuit of size $O(p(n))$.*

Combined with Lemma 5.1 we get the main theorem of this section:

Theorem 5.5 *Let $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be a two-party functionality that can be computed by a (uniform) family of circuits of polynomial size $s(n)$. Suppose Assumption 5.2 holds and a (standard) OT protocol exists. Then there is a two-party protocol π_f that realizes f in the semi-honest model, where each party in π_f can be implemented by a circuit of size $O(s(n))$.*

Remark 5.6 (On generalizing to $k > 2$ parties) The proof of Theorem 5.5 extends in a straightforward way to the case of a constant number of parties $k > 2$. In this case the general structure of the GMW protocol [30] remains essentially the same, except that each pair of parties needs to perform $O(s)$ OTs. Thus, the overhead increases by a factor of $O(k^2)$. A similar generalization applies to all variants of the basic protocol described below.

Remark 5.7 (On instantiating Assumption 5.2) Candidates for polynomial-stretch PRGs in NC^0 can be obtained from the work of Mossel et al. [38]. They present polynomial-stretch NC^0 functions that are proved to be ϵ-biased generators [39] for a negligible function ϵ, and pose the security of these functions as cryptographic

PRGs as an open question. We view their construction as providing good PRG candidates on which our protocol can be based. One important caveat to be noted is that the construction from [38] relies on unbalanced expanders for which no explicit construction is known. A randomized choice of a graph with the required parameters yields a good expander graph except with small but *non-negligible* probability. Specifically, the failure probability is inverse polynomial in the output length, where the exponent of the polynomial grows with the locality. Thus, when applying this candidate with our protocol one needs to accept a "once and for all" setup phase that may fail with a tiny but non-negligible probability; given that the setup succeeds the protocol is fully secure. Alternatively, one can combine $k(n)$ independent PRG candidates by taking the exclusive-or of their output, where $k(n) = \omega(1)$ is an arbitrarily small super-constant function. This will lead to protocols with an arbitrarily small super-constant overhead and negligible failure probability.

5.2 The Malicious Model

We now briefly sketch extensions of previous results to the malicious model. Our main result is a secure two-party computation protocol secure in the malicious model with *constant communication complexity overhead* compared to the circuit size. The problem of obtaining security in the malicious model with constant computational overhead remains an interesting open problem, but we are able to achieve some partial solutions that are close to achieving this goal.

The general approach we take is to adapt the "GMW paradigm" [30, 26] for upgrading the security of the protocol of Theorem 5.5 from the semi-honest model to the malicious model. Roughly speaking, this would require players to commit to their inputs and random coins, and then execute the protocol for the semi-honest model while proving in zero-knowledge, for every message they send, that this message is consistent with the committed information and with previous messages. Note, however, that in the case where the circuit C is narrow and deep, the number of rounds in the protocol may be linear in $|C|$. In such a case, this approach will inherently fail to maintain a constant overhead even when considering communication complexity alone.

Instead, we would like the players to use zero-knowledge sparingly; intuitively, what we'd like is for players to execute the protocol until the final round, and before sending their final messages prove in zero-knowledge that every message they sent until this point was valid. This variant of the GMW paradigm is not secure in general, since a malicious behavior of one player can compromise the privacy of the other even at early stages of the protocol. However, for the protocol of Theorem 5.5 this kind of approach can be securely implemented.

Specifically, we implement the protocol so that all of the real OT invocations use an OT protocol with security in the malicious model (since there are only few such invocations they do not form an efficiency bottleneck) and the same player who acts as the receiver in the OT^p subprotocol also acts as the receiver in the $O(s)$ OT's invoked by the GMW protocol. The resulting protocol has the desirable feature that until the final messages are sent none of the players can gain information about the other player's input even by behaving maliciously. Thus, it suffices that the players give one zero-knowledge proof that refers to all of the messages sent before the last message. Note that by implementing the required commitments using a commitment scheme with a constant overhead (whose existence follows from our underlying assumption), the size of the circuit defining the predicate for which zero-knowledge proofs should be given is linear in s. Thus, realizing the zero-knowledge proofs with a constant overhead would yield constant-overhead secure computation in the malicious model.

Unfortunately, we do not know how to construct zero knowledge protocols with a constant overhead and view this as an intriguing open problem. However, we can obtain a constant *communication* overhead. In this case, one could use either a variant of the GMW compiler that employs sublinear-communication zero-knowledge arguments (assuming collision-resistant hashing) [40] or alternatively use the recent "constant-rate" zero-knowledge protocols from [34]. An advantage of the latter protocols is that their computational overhead, while not constant, depends only on a *statistical* (rather than computational) security parameter. Thus, the computational overhead can be made polylogarithmic without resorting to exponential hardness assumptions.

We finally note that by relaxing the standard requirement of security in the malicious model, one can maintain the constant computational overhead while still maintaining an appealing security guarantee. A simple approach would be to apply the protocols from [34] with a small constant soundness error. This yields protocols in which cheaters can gain advantage with at most ϵ-probability, but by attempting to cheat they take the risk of being caught cheating. This notion of "security against covert adversaries" was recently formalized in [7]. By taking a more sophisticated approach, we can reduce the failure probability to $\epsilon = n^{-c}$ using a computational overhead of $O(c)$ in a RAM model. This yields standard security in the malicious model with an arbitrarily small super-constant computational overhead (on a RAM machine). Further details are deferred to the final version of this paper.

6 Conclusions and Open Questions

We have shown that most cryptographic tasks can be implemented with a constant computational overhead under strong, yet arguably plausible, intractability assumptions. Our work leaves many interesting open questions and research directions. One natural direction is to relax the type of assumptions on which we rely. While we believe that achieving our goals *requires* the use of assumptions that are now considered nonstandard, there is still a lot of room for generalizing these assumptions and connecting them with well-studied problems. For the case of secure two-party computation, an interesting question is to either obtain a fully explicit candidate for a polynomial-stretch PRG in NC^0 (avoiding the subtle caveat discussed in Remark 5.7) or, better yet, avoid this assumption altogether.

There are several important primitives for which we do not know how to achieve constant overhead even under the type of assumptions used in this work. These include collision-resistant hash functions, zero-knowledge proofs, and secure computation in the (strict) malicious model.

A final interesting goal is to reduce the exact constants in the asymptotic overhead incurred by our constructions (which we did not attempt to optimize) as well as the additive terms that depend on the concrete level of security.

Acknowledgments. We thank Omer Reingold for pointing out the construction of constant-overhead PRFs. We also thank Jon Feldman, Venkat Guruswami, Omer Reingold, Ronny Roth, and Avi Wigderson for helpful discussions and comments.

References

[1] M. Alekhnovich. More on average case vs approximation complexity. In *Proc. 44th FOCS*, pages 298–307, 2003.

[2] N. Alon, J. Bruck, J. Naor, M. Naor, and R. M. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory* 38(2) (1992).

[3] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006. Earlier version in FOCS 2004.

[4] B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications. *Computional Complexity*, 15(2):115–162, 2006. Earlier version in CCC 2005.

[5] B. Applebaum, Y. Ishai, and E. Kushilevitz. On pseudorandom generators with linear stretch in NC^0. In *Proc. 10th Random*, 2006.

[6] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography with Constant Input Locality. In *Proc. of Crypto*, 2007.

[7] Y. Aumann and Yehuda Lindell. Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries. In *Proc. TCC 2007*, pages 137-156.

[8] D. Beaver. Precomputing oblivious transfer. In *CRYPTO*, pages 97–109, 1995.

[9] D. Beaver. Correlated pseudorandomness and the complexity of private computations. In *28th STOC*, pages 479–488, 1996.

[10] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of 20th STOC*, pages 1–10, 1988.

[11] A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. In *Proc. of 32nd STOC*, pages 435-440, 2000.

[12] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13:850–864, 1984. Preliminary version in FOCS 82.

[13] J. L. Bordewijk. Inter-reciprocity applied to electrical networks. *Applied Scientific Research B: Electrophysics, Acoustics, Optics, Mathematical Methods*, 6: 1–74, 1956.

[14] R. Canetti. Security and composition of multiparty cryptographic protocols. In *J. of Cryptology*, 13(1), 2000.

[15] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-Resilient Functions and All-or-Nothing Transforms. In *Proc EUROCRYPT 2000*, pages 453-469.

[16] M. R. Capalbo, O. Reingold, S. P. Vadhan, and A. Wigderson. Randomness conductors and constant-degree lossless expanders. In *Proc. STOC 2002*, pages 659-668.

[17] L. Carter and M. N. Wegman. Universal Classes of Hash Functions. *J. Comput. Syst. Sci.* 18(2): 143-154 (1979).

[18] B. Chor, O. Goldreich, J. Haastad, J. Friedman, S. Rudich, and R. Smolensky. The Bit Extraction Problem of t-Resilient Functions (Preliminary Version) In *Proc. FOCS 1985*, pages 396-407.

[19] M. Cryan and P. B. Miltersen. On pseudorandom generators in NC^0. In *Proc. 26th MFCS*, 2001.

[20] R. L. Dobrushin, S. I. Gelfand, and M. S. Pinsker. On complexity of coding. In *Proc. 2nd Internat. Symp. on Information Theory*, pages 174-184, 1973.

[21] D. Dolev, C. Dwork and M. Naor. Non-malleable Cryptography. *SIAM Journal of Computing*, 30(2):391-437, 2000.

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

[23] U. Feige, J. Killian, and M. Naor. A minimal model for secure computation (extended abstract). In *Proc. of the 26th STOC*, pages 554–563, 1994.

[24] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient Private Matching and Set Intersection. In *EUROCRYPT 2004*, pages 1-19.

[25] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[26] O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[27] O. Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(090), 2000.

[28] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM* 33(4): 792-807, 1986.

[29] S. Goldwasser, S. Micali, and P. Tong. Why and How to Establish a Private Code on a Public Network. In *FOCS 1982*, pages 134-144.

[30] O. Goldreich, S. Micali, and A. Wigderson. How to Prove all NP-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design. In *CRYPTO 1986*, pages 171-185.

[31] V. Guruswami and P. Indyk. Expander-Based Constructions of Efficiently Decodable Codes. In *Proc. FOCS 2001*, pages 658-667.

[32] R. Impagliazzo, L. A. Levin, and M. Luby. Pseudo-random generation from one-way functions. In *Proc. STOC 1989*, pages 12-24.

[33] Y. Ishai and E. Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. 29th ICALP*, pages 244–256, 2002.

[34] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-Knowledge from Secure Multiparty Computation. In *Proc. STOC 2007*.

[35] J. Kilian. Founding cryptography on oblivious transfer. In *20th STOC*, pages 20–31, 1988.

[36] V. Lyubashevsky. The Parity Problem in the Presence of Noise, Decoding Random Linear Codes, and the Subset Sum Problem. In *Proc. APPROX-RANDOM 2005*, pages 378-389.

[37] Y. Mansour, N. Nisan, and P. Tiwari. The Computational Complexity of Universal Hashing. In *Proc. STOC 1990*, pages 235-243.

[38] E. Mossel, A. Shpilka, and L. Trevisan. On ϵ-biased generators in NC^0. In *Proc. 44th FOCS*, pages 136–145, 2003.

[39] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[40] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proc. 33rd STOC*, pages 590–599, 2001.

[41] Moni Naor, Moti Yung. Universal One-Way Hash Functions and their Cryptographic Applications. In *Proc. STOC 1989*, pages 33-43.

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

[43] M. Sipser and D. A. Spielman. Expander Codes. In *Proc. FOCS 1994*, pages 566-576.

[44] W. D. Smith. 1. AES seems weak. 2. Linear time secure cryptography. Cryplology ePrint report 2007/248.

[45] D. A. Spielman. Linear-time encodable and decodable error-correcting codes. STOC 1995: 388-397

[46] A. C. Yao. Theory and application of trapdoor functions. In *Proc. 23rd FOCS*, pages 80–91, 1982.

[47] A. C. Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pages 162–167, 1986.