Encoding Functions with Constant Online Rate or How to Compress Garbled Circuits Keys*

Benny Applebaum[1], Yuval Ishai[2], Eyal Kushilevitz[2], and Brent Waters[3]

[1] School of Electrical Engineering, Tel-Aviv University
bennyap@post.tau.ac.il
[2] Department of Computer Science, Technion
{yuvali,eyalk}@cs.technion.ac.il
[3] Department of Computer Science, University of Texas
bwaters@cs.utexas.edu.

Abstract. *Randomized encodings of functions* can be used to replace a "complex" function $f(x)$ by a "simpler" randomized mapping $\hat{f}(x;r)$ whose output distribution on an input x encodes the value of $f(x)$ and hides any other information about x. One desirable feature of randomized encodings is low *online complexity*. That is, the goal is to obtain a randomized encoding $\hat{f}$ of f in which most of the output can be precomputed and published before seeing the input x. When the input x is available, it remains to publish only a short string $\hat{x}$, where the online complexity of computing $\hat{x}$ is independent of (and is typically much smaller than) the complexity of computing f. Yao's garbled circuit construction gives rise to such randomized encodings in which the online part $\hat{x}$ consists of n encryption keys of length κ each, where $n = |x|$ and κ is a security parameter. Thus, the *online rate* $|\hat{x}|/|x|$ of this encoding is proportional to the security parameter κ.

In this paper, we show that the online rate can be dramatically improved. Specifically, we show how to encode any polynomial-time computable function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ with online rate of $1 + o(1)$ and with nearly linear online computation. More concretely, the online part $\hat{x}$ consists of an n-bit string and a single encryption key. These constructions can be based on the decisional Diffie-Hellman assumption (DDH), the Learning with Errors assumption (LWE), or the RSA assumption. We also present a variant of this result which applies to *arithmetic formulas*, where the encoding only makes use of arithmetic operations, as well as several negative results which complement our positive results.

Our positive results can lead to efficiency improvements in most contexts where randomized encodings of functions are used. We demonstrate this by presenting several concrete applications. These include protocols for secure multiparty computation and for non-interactive verifiable computation in the preprocessing model which achieve, for the first time, an optimal online communication complexity, as well as non-interactive zero-knowledge proofs which simultaneously minimize the online communication and the prover's online computation.

* A preliminary full version is available at [6].

1 Introduction

Suppose that we want to perform some cryptographic task which involves computation and communication on n-bit data. In many scenarios, it is beneficial to minimize the online complexity (i.e., the resources spent after seeing the data) and shift the expensive computation and communication to an offline phase. This setting has been extensively studied in many contexts including signatures [17, 40], verifiable computation (delegation) [19, 4, 14], and secure computation [8, 32, 11, 15, 31]. The goal of the present paper is to further explore the question of minimizing the online complexity of cryptography.

Let us first consider the following concrete example from [5]. Imagine a scenario of sending a weak device U to the field in order to perform some expensive computation f on sensitive data x. The computation is too complex for U to quickly perform it on its own and, since the input x is sensitive, U cannot just send the entire input out. Ideally, we would like to have a *non-interactive* solution of the following form: In an offline phase, before sent to the field, U picks a short random secret key sk and publishes a (potentially long) related public key pk. Once it observes the input x, the device U applies some cheap computation to sk and x and sends out the result $\hat{x}$, a short "encrypted" version of x. The rest of the world should be able, at this point, to recover $f(x)$ and nothing else.

Abstracting the above, the computation of U can be described as a randomized function $\hat{f} : (x; \mathsf{sk}) \mapsto (\mathsf{pk}, \hat{x})$ that *encodes* the value $f(x)$ in the sense that $(\mathsf{pk}, \hat{x})$ reveals $f(x)$ but nothing else. Using the terminology of [3], the function $\hat{f}$ is referred to as a *randomized encoding* (RE) of f. The general motivation for using REs is the hope to make $\hat{f}$ in some sense "simpler" than f, where different applications dictate different notions of simplicity. The earliest uses of REs in cryptography were in the area of secure computation [42, 34, 18, 30]. Along the years, REs have found a diverse range of other applications to problems such as computing on encrypted data [39, 13], parallel cryptography [3, 2], verifiable computation [19, 4], software protection [25, 27, 9], functional encryption [38, 26], key-dependent message security [7, 1, 10], and others. We refer the reader to [10] for a finer-grained treatment of REs under the term "garbling schemes".

In the online/offline setting considered here, we would like to minimize the online computation and communication resources required for computing and distributing $\hat{x}$. That is, we would like the online time complexity of computing $\hat{x}$ to be much smaller than the time required for computing f, and the length of $\hat{x}$ to be not much bigger than that of x.

The best known general constructions of online-efficient REs are based on Yao's garbled circuit technique [42]. In this case, the output of $f(x)$ is encoded by an offline part pk which consists of a big "garbled circuit" and an online part $\hat{x}$ which consists of n keys $K_1, \ldots, K_n$ of size κ each, where n is the bit-length of x and κ is a security parameter. (Under a standard asymptotic security convention in which n serves both as an input length parameter and a security parameter, κ can be thought of as n^{ε}, for some small constant $\varepsilon > 0$.) Each key K_i is selected from a pair of keys $(K_{i,0}, K_{i,1})$ according to the i-th input bit x_i. Hence, the online computation and communication complexity are both

$O(n\kappa)$. An appealing feature is that the online computation complexity is nearly linear in the input length, independently of the complexity of f. However, an undesirable feature is that the *online rate* of the construction — i.e., the ratio between the bit-length of $\hat{x}$ and the bit length of x — grows linearly with the security parameter κ. Hence, we ask:

> Is it possible to obtain a *constant* online rate or even rate of $1+o(1)$ (e.g., $|\hat{x}| = n + \mathrm{poly}(\kappa)$) while keeping the online computation independent of the complexity of f?

1.1 Our Contribution

We answer the above question in the affirmative by constructing, under a variety of standard intractability assumptions, an online-efficient RE with rate $1 + o(1)$ for every polynomial-time computable function.

Theorem 1. *(Informal) Under the Decisional Diffie-Hellman Assumption (DDH), the RSA Assumption, or the Learning-with-Errors Assumption (LWE), every polynomial-time computable function $f : \{0,1\}^n \to \{0,1\}^{m(n)}$ admits an RE with online rate $1 + o(1)$ and with $O(n^{1+\varepsilon})$ online computation, for any $\varepsilon > 0$.*

In more concrete terms, our constructions efficiently compile any boolean circuit C into a corresponding RE with succinct and efficiently computable online part. These constructions can be viewed as analogues of the garbled circuit construction in which the n keys determined by x are compressed into a shorter string $\hat{x}$ whose length is very close to that of x. This comes at the cost of a slight increase in the online computation complexity, which still remains nearly linear in n. An additional (related) difference is that in contrast to the standard garbled circuit construction, where each bit of $\hat{x}$ depends only on a single bit of x, in our constructions there are bits of $\hat{x}$ which depend on many bits of x. We prove that this is inherent for REs with constant or even logarithmic online rate. In particular, it is impossible to obtain a direct generalization of the garbled circuit construction in which each input bit x_i selects between a pair of keys $(K_{i,0}, K_{i,1})$ which have constant size.

The DDH and LWE based constructions are *affine* in the sense that after the private randomness is fixed in the offline phase, the remaining computation can be described as an affine function of the inputs x (over some ring R, e.g., $\mathsf{R} = \mathbb{Z}_p$ where p is the size of a DDH group). This captures a strong form of algebraic simplicity which is useful for some of the motivating applications (e.g., secure computation).

Motivated by the concrete efficiency of encoding *arithmetic* computations, we also present an LWE-based arithmetic variant of the above result that applies to arithmetic *formulas* (i.e., circuits of fan-out 1) over large finite fields, where the encoding is restricted to applying arithmetic operations to the inputs. Specifically, we obtain an affine randomized encoding (ARE, for short) with optimal online rate (i.e., $1+o(1)$) for arithmetic mod-p formulas, assuming that elements of $\mathbb{Z}_p$ can be viewed as elements of $\mathbb{Z}_q$ for some $q \gg p$. If we insist on working

in the more restricted model of [5], where the encoding should be affine over the integers, then we get a constant-rate encoding.

It should be mentioned that the online *computational* overhead of our constructions is still polynomial in the security parameter. Whether this overhead can be improved remains an interesting open question.

Lower bounds. We further explore the complexity of REs in the online/offline setting by proving several lower bounds on the online and offline rate of REs which complement our positive results. Among other results, we study the minimal achievable online rate. The online rate is clearly lower-bounded by 1 for some functions with long outputs (this is the case, for instance, for the identity function). This leaves open the possibility of achieving a strictly better rate for boolean functions. We show that even in the case of boolean functions, the online rate of *affine* REs (satisfying the algebraic simplicity condition discussed above) cannot generally be smaller than 1. Thus, achieving rate $1 + o(1)$ is essentially optimal for affine REs. While we cannot unconditionally prove a similar result for non-affine REs with, say, quadratic online computation, such a negative result follows from the conjecture that for any $c > c'$, an input for a time-(n^c) computation cannot generally be "compressed" by a time-$(n^{c'})$ algorithm into a shorter string which contains sufficient information to recover the output. See [29, 16] for related conjectures.

Adaptive security. Informally, an offline/online RE is adaptively secure if $\hat{f}(x; r) = (\mathsf{pk}, \hat{x})$ remains private even if the online input x is adaptively chosen based on the offline part of the encoding, pk. Similarly to all other known implementations of garbled circuits with short keys, our constructions cannot be proved to satisfy this stronger notion of security unless analyzed in the (programmable) random oracle model. We prove that this is inherent to some extent: in any RE whose adaptive security holds in the plain model, the length of the online part $\hat{x}$ should grow with the output length of f. (This negative result is similar in spirit to negative results for non-committing encryption [37] or functional encryption [12].) In contrast, our constructions in the non-adaptive setting (or the adaptive setting with random oracles) have online rate of $1+o(1)$, independently of the output length of f. Adaptive security of garbled circuits has recently been considered in the work of Bellare et al. [9]. The above negative result partially settles a question left open by [9].

On concrete efficiency. In concrete terms, our offline/online REs reduce the online communication of Yao's garbled circuit construction by a factor of $\kappa \approx 100$ at the expense of introducing "public-key" computations. This is not always a good tradeoff in practice. For instance, communicating 100 bits is typically less expensive than a single modular exponentiation. Luckily, our REs are also very cheap in online computation. For instance, the online encoding in the DDH-based construction involves at most one mod-p *addition* per input bit, where p is the order of the DDH group. Since a mod-p addition is typically much cheaper than the amortized cost of communicating a bit (let alone 100 bits),

we improve the overall concrete online complexity by roughly a factor of 100. This is contrasted with most applications of public-key cryptography towards improving communication complexity, where the additional computational cost outweighs the savings in communication (cf. [41]). While our REs do increase the complexity of the offline encoding and online decoding, the additional overhead is insignificant when the circuit complexity of f is much bigger than its input size. Thus, our offline/online REs seem to have a true practical potential in secure computation or delegation scenarios in which a weak client (who performs the offline and online encoding) interacts with a powerful server (who performs the online decoding).

1.2 Applications

Our positive results can lead to efficiency improvements in most contexts in which randomized encodings of functions are used. We focus on three representative applications.

Secure Multiparty Computation (MPC). In the online/offline model (or preprocessing model) for MPC, there are t players who wish to securely compute some fixed public function f. In the offline phase, before the inputs "arrive", the parties are allowed to invoke some (relatively expensive) protocol; later, in the online phase, the parties get their inputs and apply an online (hopefully cheap) protocol. The close connection of REs to MPC [30] allows to translate our results into highly efficient MPC protocols in the offline/online setting. In Section 5, we further extend and optimize these reductions (exploiting the affinity property and the information-theoretic techniques from [11]). This leads to general MPC protocols in which the online phase only requires each party to broadcast a message of the same length as its input along with a message of size $\mathrm{poly}(\kappa)$, where κ is a security parameter. Again, this is information-theoretic optimal, and it beats, in terms of online communication complexity, all previously known results even in the simplest case of two semi-honest parties. We note, however, that our protocols do not offer provable security against malicious parties which adaptively choose their inputs based on the information they receive in the offline phase, except in the random oracle model or under nonstandard assumptions. See full version for further discussion.

It is instructive to compare the efficiency of our RE-based protocols to protocols which are based on fully homomorphic encryption (FHE). The following discussion is restricted to the preprocessing model, which does not seem to significantly improve the complexity of FHE-based protocols. In FHE based protocols (as well as all other general MPC protocols from the literature) the communication complexity grows at least linearly with the total input and output length $n+m$. In contrast, the online communication complexity of our protocol does *not* depend on the output length. This is particularly useful when securely computing functionalities that have a short online secret input (say, shares of a signature key) and a long output (say, signatures on many predetermined messages using the shared signature key). Furthermore, our protocols can be made completely

non-interactive in certain scenarios, e.g., when part of the secret input is known offline and the online part is known in its entirety to one of the parties. This is impossible to get using FHE.[4] On the other hand, our protocols are incomparable to FHE-based protocols in terms of their online computational complexity. In the case of computing a complex function f which takes inputs from Alice and Bob and delivers an output to Alice, our approach yields two-message protocols in which Bob's online computation is very efficient (nearly linear in its input), whereas FHE provides similar protocols in which Alice's computation is very efficient (quasilinear in the input and output). From a concrete efficiency point of view, the online phase of our protocols is much "lighter" (e.g., Bob only needs to add a subset of $\mathbb{Z}_p$ elements corresponding to its input) and they can also be based on a wider variety of assumptions.

Verifiable Computation. In an online/offline protocol for *verifiable computation* (VC), a computationally weak client with an input x delegates a complex computation f to an untrusted server in a two phase manner. In the offline phase the client sends to the server a possibly long and computationally expensive message pk, and at the online phase (when the input x arrives) the client sends a message $\hat{x}$ to the server, and receives back the result of the computation y together with a certificate for correctness. This setting was studied in several works (e.g., [36, 25, 33, 19, 14, 4, 9]). Specifically, in [19] Yao's garbled circuit technique was used to achieve efficient VC in the online/offline model. (The security of the construction follows from standard assumptions only when the input x is picked by the client independently of pk [9].) This connection was generalized and optimized in [4]. By plugging our encodings in these protocols, we get *communication optimal* VC protocols, where the bit-length of the up-stream (online) message from the client to the server is $n+\kappa$ and the bit-length of the down-stream message (from server to client) is $m+\kappa$, where n is the input length, m is the output length and κ is the security parameter. Information-theoretically, $n+m$ bits are necessary even if the server is fully trusted. To the best of our knowledge, all previous protocols, including ones which are based on fully homomorphic encryption, have a *multiplicative* overhead of κ, either with respect to n or to m.

Non-Interactive Zero-Knowledge (NIZK). The complexity of NIZK has received much attention. The length of traditional NIZK proofs for NP grows linearly with the size of a circuit $R(x, w)$ which verifies that w is a legal witness for the statement $x \in L$. Using FHE, these traditional NIZKs can be converted into ones whose length is only $|w| + \mathrm{poly}(\kappa)$ bits [20, 28]. The proof consists of an FHE encryption c of w, along with a traditional NIZK proving that the ciphertext resulting from evaluating the verification algorithm on c encrypts the result of a correct verification. Thus, the prover's computation grows linearly with the time required for verifying $R(x, w)$, which can be an arbitrary polynomial in $|w|$. Moreover, there seems to be no obvious way to reduce this computational cost

[4] Similarly, FHE does not yield a non-interactive solution to the motivating problem described in the beginning of the introduction.

using offline preprocessing. Our results yield offline/online NIZK proofs with online proof length of $|w| + \text{poly}(\kappa)$ bits as before, but where the prover's online computation is nearly linear in $|w| + |x|$. This is done as follows. The common reference string of the NIZK defines a function f which maps w (along with a short seed which generates the prover's secret randomness) into a NIZK proof π. Applying our offline/online REs to this f yields the desired result. We note that while the length of NIZK *arguments* can be made sublinear in $|w|$ (under nonstandard but plausible assumptions), breaking this barrier in the case of *proofs* seems highly unlikely [22].

1.3 Techniques

We briefly sketch some of the ideas used to prove Theorem 1. Our starting point is a standard garbled-circuit based encoding, such as the one from [2]. In the offline phase of this encoding, we garble the circuit f and prepare, for each input i, a pair of random secret keys (K_i^0, K_i^1). In the online phase, for each i, we use the i-th bit of x to select a key $K_i^{x_i}$ and output the selected keys. In order to reduce the online complexity of the encoding, we would like to have a compact way to reveal the selected keys. Let us consider the following "riddle" which is a slightly simpler version of this problem. In the offline phase, Alice has n vectors $M_1, \ldots, M_n \in \{0,1\}^k$. She is allowed to send Bob a long encrypted version of these vectors. Later, in the online phase, she receives a bit vector $x \in \{0,1\}^n$. Her goal is to let Bob learn only the vectors which are indexed by x, i.e., $\{M_i\}_{i:x_i=1}$ while sending only a single message of length $O(n)$ bits (or even $n + \kappa$ bits).[5]

Before solving the riddle, let us further reduce it to an algebraic version in which Alice wants to reveal a 0-1 linear combination of the vectors which are indexed by x. Observe that if we can solve the new riddle with respect to nk-bit vectors $T = (T_1, \ldots, T_n)$, then we can solve the original riddle with k-bit vectors $(M_1, \ldots, M_n)$. This is done by placing the M_i's in the diagonal of T, i.e., T_i is partitioned to k-size blocks with M_i in the i-th block and zero elsewhere. In this case, Tx simply "packs" the vectors $\{M_i\}_{i:x_i=1}$.

It turns out that the linear version of the riddle can be efficiently solved via the use of a symmetric-key encryption scheme with some (additive) homomorphic properties. Specifically, let (E, D) be a symmetric encryption scheme with both key homomorphism and message homomorphism as follows: A pair of ciphertexts $\mathsf{E}_k(x)$ and $\mathsf{E}_{k'}(x')$ can be mapped (without any knowledge of the secret keys) to a new cipheretxt of the form $\mathsf{E}_{k+k'}(x+x')$. Given such a primitive the answer to the riddle is easy: Alice encrypts each vector under a fresh key K_i and publishes the ciphertexts C_i. At the online phase Alice sends the sum of keys $K_x = \sum K_i x_i$ together with the indicator vector x. Now Bob can easily construct $C = \mathsf{E}_{K_x}(Mx)$ by combining the ciphertexts indexed by x and, since K_x is known, Bob can decrypt the result. Intuitively, Bob learns nothing about

[5] The main difference between the riddle and the garbled-circuit problem is that in the latter case, the vector x itself should remain hidden; this gap is bridged by permuting the pairs and randomizing the vector x; see Section 4.

a column M_j which is not indexed by x as the online key K_x is independent of the j-th key. Our DDH and LWE based solutions are based on (approximate) implementations of this primitive. (A somewhat different approach is used in the RSA-based construction.)

The arithmetic setting is more challenging. Here, instead of computing the selection function, we should compute an affine function $Mx + v$ over the integers or over $\mathbb{Z}_p$, for some large integer p (not necessarily a prime). While it is possible to solve this via a similar encryption scheme with (stronger) additive homomorphism, there are several technical problems. Typically, all (or most) of the coordinates of x are non-zero and so we should argue that given K_x the secrecy of the key K_i was not compromised, despite the fact that K_i may participate in the linear combination K_x. This translates to some form of security under Related-Key attacks. In addition, it is harder to achieve homomorphism for integers or over $\mathbb{Z}_p$ directly, and so one should somehow embed this domain in a larger, less "friendly", message space. Still, it turns out that a variant of this gadget can be implemented based on the LWE assumption. Specifically, we use the following variant of the key-shrinking gadget of [5] (which was originally introduced as a tool for garbling arithmetic circuits). Intuitively, we create a noisy version $\hat{M}$ and $\hat{v}$ of the matrix M and the vector v, and then plant them in a random linear space W of a low dimension κ over $\mathbb{Z}_q$ (where $q \gg p$). The space W is made public. Now every linear combination of $\hat{M}$ and $\hat{v}$ lies in W, and so it can be succinctly described by its coefficients with respect to W. In particular, to reveal the output $Mx+v$, it suffices for the encoding to reveal the coefficients of its representation $\hat{M}x+\hat{v}$. The security of the construction follows from the LWE assumption.

Concurrent and subsequent works. The recent works [24, 23] gives the first *reusable* construction of garbled circuits. This implies REs in which a single offline computation can support an arbitrary polynomial number of efficient online computations. The question of optimizing the online rate of reusable garbled circuits remains open. On a different front, improvements in the size of garbled circuits for uniform Turing Machine or RAM computations were recently given in [35, 23]. These lead to REs with succinct offline outputs. Our construction can be applied on top of these constructions, yielding REs with an online output of size $n+o(n)$, nearly linear online computation, and offline outputs that are only longer by an additive term of $O(n^{\varepsilon} \cdot T)$ than those in [35, 23], where T is the online computational complexity of the original constructions.

Organization. Section 2 gives the necessary background on randomized encodings. In Section 3, we present several constructions of succinct randomized encodings for a concrete boolean function called the subset function (SF). Later, in Section 4, we use these encodings as a building block and obtain succinct encodings for general boolean functions. In Section 5, we sketch the application of succinct randomized encodings to secure multiparty computation (MPC). Applications related to non-interactive zero-knowledge proofs (NIZK), and verifiable computation (VC) in the preprocessing model are deferred to the full version [6],

which also contains the construction of succinct encoding for arithmetic formulas, some lower bounds and a detailed treatment of the issue of adaptivity.

2 Randomized Encoding of Functions

Intuitively, a randomized encoding of a function $f(x)$ is a randomized mapping $\hat{f}(x;r)$ whose output distribution depends only on the output of f. We formalize this intuition via the notion of *computationally-private perfectly-correct randomized encoding* (in short RE) from [2]. In the following, we assume that f is defined over $\mathbb{Z}_p^n$ for some integer p (by default $p=2$), and allow the encoding $\hat{f}$ be defined over a possibly larger alphabet $\mathbb{Z}_q^n$ for $p \leq q$ under the convention that a vector $x \in \mathbb{Z}_p^n$ can be naturally identified with a vector $x \in \mathbb{Z}_q^n$.

Definition 1 (Randomized Encoding (RE)). *Let $p = p(n), q = q(n)$ where $p(n) \leq q(n) \leq 2^{\mathrm{poly}(n)}$ and $\ell = \ell(n), m = m(n), s = s(n) = \mathrm{poly}(n)$ be integer valued functions. We naturally view $\mathbb{Z}_p$ as a subset of $\mathbb{Z}_q$. Let $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^\ell$ be an efficiently computable function. We say that an efficiently computable randomized function $\hat{f} : \mathbb{Z}_q^n \times \{0,1\}^m \to \mathbb{Z}_q^s$ is a* perfectly-correct computationally-private randomized encoding *of f (in short, RE), if there exist an efficient decoder algorithm* Dec *and an efficient simulator* Sim *that satisfy the following conditions:*

- **Perfect correctness.** *For every $x \in \mathbb{Z}_p^n$, $\Pr_r[\mathsf{Dec}(1^n, \hat{f}(x;r)) \neq f(x)] = 0$.*
- *(t, ε)* **privacy.** *For every sequence $\{x_n\}_n$, where $x_n \in \mathbb{Z}_p^n$, and every $t(n)$-size circuit $\mathcal{A}$*

$$\left|\Pr[\mathcal{A}(\hat{f}(x_n;r)) = 1] - \Pr[\mathcal{A}(\mathsf{Sim}(1^n, f(x_n))) = 1]\right| \leq \varepsilon(n).$$

By default, $t = n^{\omega(1)}$ and $\varepsilon = n^{-\omega(1)}$, i.e., the distributions are computationally indistinguishable *(denoted by $\stackrel{c}{\equiv}$). The encoding is* statistically *secure if t is unbounded and* perfectly *secure if, in addition, $\varepsilon = 0$.*

Remarks.

- (Security parameter.) The above definition uses n both as an input length parameter and as a cryptographic "security parameter" quantifying computational privacy. When describing our constructions, it will be convenient to use a separate parameter κ for the latter, where computational privacy will be guaranteed as long as $\kappa \geq n^\varepsilon$ for some constant $\varepsilon > 0$.
- (Collections) Let $\mathcal{F}$ be a collection of functions with an associated representation (by default, a boolean or arithmetic circuit). We say that a class of randomized functions $\hat{\mathcal{F}}$ is an RE of $\mathcal{F}$ if there exists an efficient algorithm (compiler) which gets as an input a function $f \in \mathcal{F}$ and outputs (in time polynomial in the representation length $|f|$) three circuits $(\hat{f} \in \hat{\mathcal{F}}, \mathsf{Dec}, \mathsf{Sim})$ which form a $(t = n^{\omega(1)}, \varepsilon = n^{-\omega(1)})$-RE of f.

2.1 Efficiency Measures

So far the notion of RE can be trivially satisfied by taking $\hat{f} = f$ and letting the simulator and decoder be the identity functions. To make the definition non-trivial, we should impose some efficiency constraint. In this work, our main measure of efficiency is online complexity.

Online/Offline Complexity. We would like to measure separately the complexity of the outputs of $\hat{f}$ which depend solely on r (*offline* part) from the ones which depend both on x and r (*online* part). Without loss of generality, we assume that $\hat{f}$ can be written as $\hat{f}(x;r) = (\hat{f}_{\mathsf{off}}(r), \hat{f}_{\mathsf{on}}(x;r))$, where $\hat{f}_{\mathsf{off}}(r)$ does not depend on x at all. The *online communication complexity* (resp., *online computational complexity*) of $\hat{f}$ is the bit-length (resp., the time complexity) of $\hat{f}_{\mathsf{on}}(x;r)$. Similarly, the *offline communication complexity* (resp., *offline computational complexity*) of $\hat{f}$ is the bit-length (resp., the time complexity) of $\hat{f}_{\mathsf{off}}(r)$. The *rate* of $\hat{f}$ is ρ if the online communication complexity is at most ρ-times larger than the bit-length $n \log p$ of the input of the encoded function f.

Efficient online encodings. Let $\hat{\mathcal{F}}$ be an encoding of the collection $\mathcal{F}$. We say that $\hat{\mathcal{F}}$ is *online-efficient* if for every function $f \in \mathcal{F}$, the online computational complexity of the encoding $\hat{f}$ is *independent* of the computational complexity (i.e., circuit size) of the encoded function f (but grows with the bit-length of the input of f). The encoding is *online-succinct* (or simply succinct) if, in addition to being online efficient, every $f \in \mathcal{F}$ is encoded by a $1 + o(1)$-rate encoding.

*Remark 1 (**Online inputs**).* In some applications, it is natural to think of the encoded function f as having online inputs x_{on} and offline inputs x_{off}. In this case, we measure the online commuincation/computational complexity of the encoding $\hat{f}$ with respect to the outputs that depend on x_{on}. By default, we simply assume that all the input x is an online input and there is no offline part.

Some of the applications of REs further require some form of algebraic simplicity; this is captured by the notion of affinity.

Affine RE. We say that an encoding $\hat{f} : \mathbb{Z}_q^n \times \{0,1\}^m \to \mathbb{Z}_q^s$ is an *affine randomized encoding* (ARE) if, for every fixing of the randomness r, the online part of the encoding $\hat{f}_{\mathsf{on}}(x;r)$ becomes an affine function over the ring $\mathbb{Z}_q$, i.e., $\hat{f}_{\mathsf{on}}(x;r) = M_r \cdot x + v_r$, where M_r (resp., v_r) is a matrix (resp., vector) that depends on the randomness r. It will sometimes be the case that certain outputs of $\hat{f}$ are restricted to an interval $[0, q']$ in $\mathbb{Z}_q$. Each such entry will only contribute $\lceil \log_2 q' \rceil$ towards computing the rate.

*Remark 2 (**ARE vs. DARE**).* Previous works considered a stronger form of affinity called *decomposable affine randomized encoding* (DARE).[6] Decomposability requires that each output of $\hat{f}$ depends on a single deterministic input x_i. Hence, a decomposable affine randomized encoding can be written as

[6] In fact, in the conference version of [5] the term ARE was used to denote DARE.

$\hat{f}(x;r) = (\hat{f}_{\mathsf{off}}(r), \hat{f}_1(x_1;r), \ldots, \hat{f}_n(x_n;r))$ where each function $\hat{f}_i$ is affine with respect to x_i. It is known how to convert an ARE to DARE, however, the known transformation introduces a non-constant ($O(n)$) multiplicative blow-up in the online communication complexity. In the full version, we show that this is inherent and decomposability *cannot* be achieved with constant rate.

*Remark 3 (**On Adaptive Security**).* In the online/offline model, it is natural to ask if the encoding can be *adaptively* secure, namely, if security holds when the online input x is chosen based on the offline part of the encoding. In the full version, we show that, in the standard model, adaptively secure REs cannot be online-efficient, let alone have constant rate (assuming the existence of one-way functions). On the other hand, it turns out that this barrier can be bypassed via the use of a (programmable) random oracle.

3 Succinct AREs for the Subset Function

In order to succinctly encode boolean circuits, we will need a succinct encoding for the following concrete function g, called the *Subset Function*. It has length parameter n and message size κ and is defined by

$$g(M, x) = ((M_i)_{i \in x}, x),$$

where $M = (M_1, \ldots, M_n) \in (\{0,1\}^\kappa)^n$ is a vector of n "messages", and $x \in \{0,1\}^n$ is a selection vector which is viewed as the set $\{i : x_i = 1\}$. (The latter convention will be implicit through the whole section.) Our goal is to encode g by an RE of the form $\hat{g}(M, x; r) = (\hat{g}_{\mathsf{off}}(M; r), x, K(x; r))$ where $K(x; r)$ is of bit-length κ^c for some universal constant c. Security will hold as long as n is bounded by some arbitrary polynomial in κ whose degree may be *independent* of the constant c. We will construct such an encoding based on several assumptions. Specifically, we will show that such an encoding can be based on a special form of symmetric-key encryption with additive homomorphism which, in turn, can be constructed under the DDH assumption or the LWE assumption. In the full version, we also present a direct encoding (which does not go through the additive homomorphism) under the RSA assumption.

3.1 Encoding the Subset Function via Additive Homomorphism

Definition 2 (Additive Homomorphic Encryption (AHE)). *An additive homomorphic Encryption is a triple of efficient algorithms* $(\mathsf{Setup}, \mathsf{E}, \mathsf{D})$ *for which the following hold:*

- ***Syntax:*** *The randomized algorithm* Setup *takes a length parameter* 1^κ *and outputs a string* param *which specifies four (additive) groups: key-space* $\mathcal{K}$*, message-space* $\mathcal{M}$*, ciphertext-space* $\mathcal{C}$ *and public randomness space* $\mathcal{W}$*. We assume that* κ*-bit strings can be efficiently embedded in* $\mathcal{M}$ *and denote the identity element of* $\mathcal{M}$ *by* $\mathbf{0}$*. The input to the encryption and decryption*

algorithms consist of a message/ciphertext, a key K, some private randomness, and some public randomness $W \xleftarrow{R} \mathcal{W}$ which is selected during the encryption. Both algorithms also depend on the string param*. (We make this dependency implicit to simplify notation.)*

- ***Semantic security:*** *Let* $\mathsf{param} = (\mathcal{K}, \mathcal{M}, \mathcal{C}, \mathcal{W}) \xleftarrow{R} \mathsf{Setup}(1^\kappa)$*. For every* $n = \mathrm{poly}(\kappa)$ *and every* n*-tuple of messages* $M_1, \ldots, M_n \in \mathcal{M}$*, we have that*

$$\left(\mathsf{param}, (W_i, \mathsf{E}_K(M_i; W_i))_{i\in[n]}\right) \stackrel{c}{\equiv} \left(\mathsf{param}, (W_i, \mathsf{E}_K(\boldsymbol{0}; W_i))_{i\in[n]}\right),$$

  where $W_i \xleftarrow{R} \mathcal{W}$*,* $K \xleftarrow{R} \mathcal{K}$*, and indistinguishability is parameterized by* κ*.*
- ***Additive Homomorphism:*** *For every* $n = \mathrm{poly}(\kappa)$ *and every* n*-tuple of keys* $K_1, \ldots, K_n \in \mathcal{K}$*,* n*-tuple of messages* $M_1, \ldots, M_n \in \mathcal{M}$*, and public randomness* $W \in \mathcal{W}$*, we have that*

$$\mathsf{D}_{\sum_i K_i}\left(\sum_i E_{K_i}(M_i; W); W\right) = \sum_i M_i,$$

  where sums are computed over the corresponding groups. In fact, it suffices to have a relaxed *form of additive homomorphism which holds in the special case where all messages, except for one, equal to* $\boldsymbol{0} \in \mathcal{M}$*.*

The definition implies that the key size is *independent* of the homomorphism parameter n. This will be crucial for our applications. As a concrete example of AHE consider the following symmetric-key version of ElGamal encryption. Let $\mathcal{M} = \mathcal{C} = \mathcal{W}$ equal to a cyclic group $\mathbb{G}$ of prime order p and let $\mathcal{K} = \mathbb{Z}_p$. Using the standard multiplicative notation, encryption is defined by $\mathsf{E}_K(M; W) = W^K \cdot M$ and decryption by $\mathsf{D}_K(C; W) = C/W^K$. It is not hard to show that if the DDH assumption holds in $\mathbb{G}$ then the scheme is an AHE with relaxed homomorphism. (More details about this implementation, as well as a description of an analogous implementation under LWE appear in the full version.) We show how to encode the subset function $g(M, x)$ with length n and message size κ based on AHE.

Lemma 1. *Assume that AHE exists. Then the Subset Function* $g(M, x)$*, where* $M \in (\{0,1\}^\kappa)^n, x \in \{0,1\}^n$*, has an encoding*

$$\hat{g}(M, x; r) = (\hat{g}_{\mathsf{off}}(M; r), x, \sum_{i\in x} K_i(r)),$$

where $\hat{g}_{\mathsf{off}}$ *outputs* $O(n^2)$ *ciphertexts in* $\mathcal{C}$*, the functions* K_i *output an element in* $\mathcal{K}$*, and the sum is computed over the key-space* $\mathcal{K}$*.*

Proof. At the offline phase, we invoke $\mathsf{Setup}(1^\kappa)$ and obtain a specification param of $\mathcal{K}, \mathcal{M}, \mathcal{C}$ and $\mathcal{W}$. We encode each entry of the offline input $M = (M_1, \ldots, M_n)$ by an element of $\mathcal{M}$, and from now on identify M_i with its encoding. We define a diagonal $n \times n$ matrix $\{M_{i,j}\}$ whose diagonal equals to the message vector M, i.e., $M_{i,i} = M_i, \forall i \in [n]$ and $M_{i,j} = \mathbf{0}, \forall i \neq j$. Next, we select a

tuple of public random elements $W = (W_1, \ldots, W_n) \xleftarrow{R} \mathcal{W}^n$, a tuple of random keys $K = (K_1, \ldots, K_n) \xleftarrow{R} \mathcal{K}^n$ and compute a matrix of "ciphertexts" $C = (C_{i,j}) \in \mathcal{C}^{n \times n}$, where $C_{i,j} = \mathsf{E}_{K_i}(M_{i,j}; W_j)$. The output of $\hat{g}_{\mathsf{off}}$ consists of the tuple (param, W, C) and the online part $\hat{g}_{\mathsf{on}}$ consists of the pair $(x, K_x = \sum_{i \in x} K_i)$.

Decoding. Given $(\mathsf{param}, W, C, x, K_x)$, we decode $(M_i)_{i \in x}$ by exploiting the homomorphism property of the above encryption. Namely, for each $j \in x$ we compute

$$Y_j = \sum_{i \in x} C_{i,j} = \sum_{i \in x} \mathsf{E}_{K_i}(M_{i,j}; W_j),$$

and output the value $\mathsf{D}_{K_x}(Y_j; W_j)$.

Simulation. For $\ell = 0, \ldots, n$ define the hybrid $H_\ell(M, x)$ exactly as in $\hat{g}$ except that

$$M_{i,i} = \begin{cases} M_i & \text{if } i < \ell \text{ or } i \in x, \\ \mathbf{0} & \text{otherwise} \end{cases}$$

The first hybrid H_0 can be sampled based on $((M_i)_{i \in x}, x)$, and so it is being used as the simulator. The last hybrid H_n corresponds to the distribution of the encoding $\hat{g}$. Hence, by a standard argument, it suffices to show that each pair of neighboring hybrids is computationally indistinguishable. Assume, towards a contradiction, that $\mathcal{A}$ distinguishes the hybrid $H_{\ell-1}$ from H_ℓ with non-negligible advantage δ. Observe that in this case $x_\ell = 0$, as otherwise the two hybrids are identically distributed. We construct a new adversary $\mathcal{B}$ that breaks the semantic security of the scheme. Given a challenge $(\mathsf{param}, \boldsymbol{w}, \boldsymbol{c})$ where $\mathsf{param} \xleftarrow{R} \mathsf{Setup}(1^\kappa)$ and $\boldsymbol{w} = (w_1, \ldots, w_n) \xleftarrow{R} \mathcal{W}^n$, the adversary $\mathcal{B}$ distinguishes between $\boldsymbol{c} \xleftarrow{R} (\mathsf{E}_K(\mathbf{0}; w_1), \ldots, \mathsf{E}_K(\mathbf{0}; w_n))$ and $\boldsymbol{c} \xleftarrow{R} (\mathsf{E}_K(\mathbf{0}; w_1), \ldots, \mathsf{E}_K(M_\ell; w_\ell), \ldots, \mathsf{E}_K(\mathbf{0}; w_n)))$ as follows. Use param to compute the hybrid $H_{\ell-1}$ where the public randomness $W_1, \ldots, W_n$ is set to $\boldsymbol{w}$, and the ℓ-th row of the ciphertext matrix C takes the value $\boldsymbol{c}$. It is not hard to verify that the resulting distribution is identical to $H_{\ell-1}$ if $\boldsymbol{c} \xleftarrow{R} (\mathsf{E}_K(\mathbf{0}; w_1), \ldots, \mathsf{E}_K(\mathbf{0}; w_n))$, and to H_ℓ if $\boldsymbol{c} \xleftarrow{R} (\mathsf{E}_K(\mathbf{0}; w_1), \ldots, \mathsf{E}_K(M_\ell; w_\ell), \ldots, \mathsf{E}_K(\mathbf{0}; w_n)))$, and the claim follows. □

Complexity. To encode the online part, one has to compute n additions (over the key space) and send x together with a single key element. The cost of the offline part is n^2 encryptions/ciphertexts. One can obtain a smooth tradeoff between the offline part and the online part by partitioning the inputs to blocks (see full version). Also note that decoding costs n^2 additions over the key space (which can be reduced via the previous optimization) and n decryption operations. Finally, we mention that in our RSA-based solution the offline complexity is only linear in n but quadratic in κ. (The latter can be improved assuming subexponential hardness of RSA.)

4 Succinct AREs for Boolean Circuits

In this section, we will encode any efficiently computable function via a succinct encoding. We begin by showing that if $F : \{0,1\}^n \to \{0,1\}^\ell$ has a decomposable affine randomized encoding (DARE) then it also has a succinct encoding. In the following, let κ be a security parameter which is polynomially related to n, i.e., $\kappa = n^\delta$ for some fixed $\delta > 0$. We will employ a succinct encoding for the subset function $g(M, \hat{x})$ with length $N = 2n$ and message size κ. We will also make use of the following simple observation: if a $\kappa \times 2n$ matrix M is composed of n pairs of columns $(M_{2i-1}|M_{2i}) = (v_i^0, v_i^1)_{i\in[n]}$, then for any $x \in \{0,1\}^n$ the sub-matrix $(v_i^{x_i})_{i\in[n]}$ can be written as $(M_i)_{i\in \mathrm{pad}(x)}$, where $\mathrm{pad}(x)$ maps an n-bit vector x to the $2n$-bit vector $(1 - x_1, x_1, \ldots, 1 - x_n, x_n)$, and $i \in \mathrm{pad}(x)$ if $\mathrm{pad}(x)_i = 1$.

Lemma 2. *Let $F : \{0,1\}^n \to \{0,1\}^\ell$ be an efficiently computable function having a decomposable ARE $f(x;\rho) = (f_{\mathsf{off}}(\rho), f_1(x_1;\rho), \ldots, f_n(x_n;\rho))$, where the output length of each f_i is κ bits. Also, assume that the subset function $g(M, \hat{x})$ with length $2n$ and message size κ has an RE of the form $\hat{g}(M, \hat{x}; r) = (\hat{g}_{\mathsf{off}}(M; r), \hat{x}, K(\hat{x}; r))$. Then, F is encoded by the randomized function*

$$\hat{F}(x;\rho,s,r) = (f_{\mathsf{off}}(\rho), \hat{g}_{\mathsf{off}}(M;r), x \oplus s, K(\mathrm{pad}(x \oplus s); r)),$$

where

$$M = (f_1(s_1;\rho)|f_1(s_1 \oplus 1;\rho)| \cdots |f_n(s_n;\rho)|f_n(s_n \oplus 1;\rho)) \in \{0,1\}^{\kappa\times 2n}.$$

Proof. It will be useful to start by encoding the n-wise one-out-of-two selection function H which maps an online input $x \in \{0,1\}^n$ and an offline matrix of pairs $V = (v_1^0|v_1^1| \ldots |v_n^0|v_n^1) \in \{0,1\}^{\kappa\times 2n}$ to the tuple $(v_i^{x_i})_{i\in[n]}$. Observe that the output of H is essentially the value of the subset function g applied to the matrix V and the vector $\mathrm{pad}(x) \in \{0,1\}^{2n}$, except that H hides x whereas g reveals it. Nevertheless one can easily randomize x and then employ the subset function. Specifically, select a random mask $s \xleftarrow{R} \{0,1\}^n$, let $\hat{x} \in \{0,1\}^{2n}$ be the vector $\mathrm{pad}(x \oplus s)$, and construct the $\kappa \times 2n$ matrix $M = (v_1^{s_1}|v_1^{s_1\oplus 1}| \ldots |v_n^{s_n}|v_n^{s_n\oplus 1})$. It is not hard to show that the randomized mapping $h(V, x; s) \mapsto g(M, \hat{x})$ is an encoding of H. Indeed, the output distribution of $g(M, \hat{x})$ consists of the matrix $(M_i)_{i\in\hat{x}}$ and the vector $\hat{x}$ — the former simply equals to $(v_i^{x_i})_{i\in[n]}$ and the latter is just a sequence of n pairs of a random bit and its complement.

Next, let us view h as a deterministic function of V, x and s. Since h can be written as $g(M_{V,s}, \hat{x}_{x,s})$, we can encode h by the mapping $\hat{g}(M_{V,s}, \hat{x}_{x,s}; r)$. It is not hard to show that the latter encoding also encodes H. Overall, our encoding for $H(V, x)$ is defined as follows:

$$(V, x; s, r) \mapsto (\hat{g}_{\mathsf{off}}(M_{V,s}; r), \mathrm{pad}(x \oplus s), K(\mathrm{pad}(x \oplus s); r)).$$

To improve the online complexity, we replace the redundant value $\mathrm{pad}(x \oplus s)$, which is sent in the clear, with $x \oplus s$.

We can now prove the lemma. Let us view ρ as a deterministic input and encode the deterministic function $f(x, \rho)$. Since f is decomposable, we can write it as

$$(f_{\mathsf{off}}(\rho), H(V_\rho, x)), \quad \text{where } V_\rho = (f_1(0;\rho)|\ldots|f_n(0;\rho)|f_1(1;\rho)|\ldots|f_n(1;\rho))$$

and H is the n-wise one-out-of-two selection function. Using appropriate substitution and concatenation lemmas (see full version), it can be shown that f is encoded by $(f_{\mathsf{off}}(\rho), \hat{h}(V, x; s, r))$, where $\hat{h}$ encodes H. Plugging in our (improved) encoding of H, we obtain an encoding of the form

$$\hat{f}(x, \rho; s, r) = (f_{\mathsf{off}}(\rho), \hat{g}_{\mathsf{off}}(M_{s,\rho}; r), x \oplus s, K(\mathrm{pad}(x \oplus s); r)).$$

Finally, a similar (composition) argument shows that the function $\hat{f}(x; \rho, s, r)$ encodes $F(x)$ and the lemma follows. □

It follows that F has an encoding with online complexity of $n + \mathsf{Len}(K)$, online computational complexity of $O(n + \mathsf{Comp}(K))$, and offline computational complexity of $\mathsf{Comp}(f_{\mathsf{off}}) + \mathsf{Comp}(\hat{g}_{\mathsf{off}})$, where $\mathsf{Comp}(\cdot)$ and $\mathsf{Len}(\cdot)$ measure the computational complexity (circuit size), and the output length (in bits) of a given function. Furthermore, observe that for every fixed randomness s each bit of the term $\mathrm{pad}(x \oplus s)$ can be written as x_i or as $1 - x_i$ and so if $K(\hat{x}; r)$ is affine (over some ring) then so is $\hat{F}_{\mathsf{on}}$.

In [2] it is shown that, assuming the existence of one-way functions, any efficiently computable function $F(x)$ can be encoded by a decomposable ARE $f(x; \rho) = (f_{\mathsf{off}}(\rho), f_1(x_1; \rho), \ldots, f_n(x_n; \rho))$, where the output length of the f_i's is κ bits, and the computational complexity of f_{off} is $\kappa \cdot \mathsf{Comp}(f)$. Combining this with Lemma 2 and our encodings for the Subset Function, we derive succinct encodings for general boolean functions. By using an optimized version of Lemma 1 (which encodes the subset function in blocks), we can do this while keeping the online computational complexity asymptotically "almost linear", as in the following theorem whose proof is deferred to the full version.

Theorem 2 (Theorem 1 restated). *Assume that the DDH assumption, or LWE assumption or the RSA assumption holds. Let $\varepsilon > 0$ be an arbitrary constant. Then, every efficiently computable function $F : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ has an encoding $\hat{F}$ with online communication of $n + o(n)$, online computational complexity of $O(n^{1+\varepsilon})$, and offline computational/communication complexity of $O(n^\varepsilon \mathsf{Comp}(F))$. Furthermore, in the case of LWE and DDH the encoding is affine.*

5 MPC with Optimal Online Communication

In this section, we sketch the application of succinct randomized encodings to secure multiparty computation (MPC) in the preprocessing model. We start with the two-party case, and later generalize to the multiparty case. For concreteness, we focus on distributing the DDH-based encoding obtained by combining

Lemmas 1 and 2 with the DDH-based AHE. Similar protocols can be obtained based on any succinct *Affine* RE. We do not know how to get similar results from general (non-affine) succinct REs.

Let F be a deterministic two-party functionality which takes an input $a \in \{0,1\}^{n_a}$ from Alice and an input $b \in \{0,1\}^{n_b}$ from Bob, and delivers an output c to Alice.[7] The DDH-based encoding of F can be written as

$$\hat{F}(a,b;R) = (\hat{F}_{\mathsf{off}}(R),\ a \oplus r^a,\ b \oplus r^b,\ \sum_{i=1}^{n_a} K^A_{i,a_i \oplus r^a_i} + \sum_{i=1}^{n_b} K^B_{i,b_i \oplus r^b_i} \mod p),$$

where the "masks" $r^a \in \{0,1\}^{n_a}$, $r^b \in \{0,1\}^{n_b}$, and the "keys" $K^A_{i,\sigma}, K^B_{i,\sigma} \in \mathbb{Z}_p$ are random and independent of a, b (these values are given as part of R).

In the semi-honest model, the protocol is straightforward. In the offline phase, a trusted party samples R and sends the value $\hat{F}_{\mathsf{off}}(R)$ together with the mask r^a to Alice, and the mask r^b along with the $2n_a + 2n_b$ keys $K^A_{i,\sigma}, K^B_{i,\sigma}$ to Bob. (Of course, in the real world, this step is implemented via the use of any off-the-shelf secure two-party protocol.) In the online phase, Alice sends to Bob $a \oplus r^a$ and Bob replies with $b \oplus r^b$ and $\sum_{i=1}^{n_a} K^A_{i,a_i \oplus r^a_i} + \sum_{i=1}^{n_b} K^B_{i,b_i \oplus r^b_i} \mod p$. Alice computes the output using the decoder of $\hat{F}$. Note that the view of Bob is completely random, whereas the view of Alice contains the output of $\hat{F}$ which can be simulated given $F(a,b)$. This proves the following:

Theorem 3. *Suppose that the DDH assumption holds in a prime order group of size $p = p(\kappa)$. Let $F(a,b)$ be a polynomial-time computable functionality which delivers its output to Alice. Assume trusted preprocessing which does not depend on the inputs. Then, F can be securely realized in the semi-honest model by a protocol in which Alice sends a message of length $|a|$ and Bob sends a message of length $|b| + \lceil \log p \rceil$, independently of the length of the output or the complexity of F.*

In the full version [6], we describe an efficient extension of this protocol to the malicious model and to the multiparty model, and discuss the issue of adaptive security. Applications related to NIZKs and verifiable computation are also deferred to the full version.

Acknowledgements. The first author was supported by Alon Fellowship, ISF grant 1155/11, Israel Ministry of Science and Technology (grant 3-9094), and GIF grant 1152/2011. The second author was supported by the European Research Council as part of the ERC project CaC (grant 259426). The third author was supported by ISF grant 1361/10 and BSF grant 2008411. The fourth author was supported by NSF grants CNS-0915361 and CNS-0952692, AFOSR Grant No: FA9550-08-1-0352, DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, and Microsoft Faculty Fellowship, and

[7] The case of general two-party functionalities reduces to this case via a standard reduction, cf. [21].

Packard Foundation Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

References

1. B. Applebaum. Key-dependent message security: Generic amplification and completeness theorems. In *EUROCRYPT*, 2011.
2. B. Applebaum, Y. Ishai, and E. Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.
3. B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.
4. B. Applebaum, Y. Ishai, and E. Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *ICALP (1)*, pages 152–163, 2010.
5. B. Applebaum, Y. Ishai, and E. Kushilevitz. How to garble arithmetic circuits. In *FOCS*, pages 120–129, 2011.
6. B. Applebaum, Y. Ishai, E. Kushilevitz, and B. Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. Cryptology ePrint Archive, Report 2012/693, 2012. http://eprint.iacr.org/.
7. B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. In *EUROCRYPT*, pages 423–444, 2010.
8. D. Beaver. Precomputing oblivious transfer. In *CRYPTO*, 1995.
9. M. Bellare, V. T. Hoang, and P. Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. In *ASIACRYPT*, pages 134–153, 2012. Full version: http://eprint.iacr.org/2012/564.
10. M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. In *ACM Conference on Computer and Communications Security*, pages 784–796, 2012. Full version: http://eprint.iacr.org/2012/265.
11. R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, 2011.
12. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.
13. C. Cachin, J. Camenisch, J. Kilian, and J. Müller. One-round secure computation and secure autonomous mobile agents. In *ICALP*, pages 512–523, 2000.
14. K.-M. Chung, Y. T. Kalai, and S. P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *CRYPTO*, 2010.
15. I. Damgard, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. Cryptology ePrint Archive, Report 2011/535, 2011. http://eprint.iacr.org/.
16. B. Dubrov and Y. Ishai. On the randomness complexity of efficient sampling. In *STOC*, pages 711–720, 2006.
17. S. Even, O. Goldreich, and S. Micali. On-line/off-line digital signatures. *J. Cryptology*, 9, 1996.
18. U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation. In *STOC*, 1994.
19. R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.

20. C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
21. O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.
22. O. Goldreich, S. P. Vadhan, and A. Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1-2):1–53, 2002.
23. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Overcoming the worst-case curse for cryptographic constructions. *IACR Cryptology ePrint Archive*, 2013:229, 2013.
24. S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.
25. S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, 2008.
26. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179, 2012.
27. V. Goyal, Y. Ishai, A. Sahai, R. Venkatesan, and A. Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, pages 308–326, 2010.
28. J. Groth. Minimizing non-interactive zero-knowledge proofs using fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2011:12, 2011.
29. D. Harnik and M. Naor. On the compressibility of NP instances and cryptographic applications. *SIAM J. Comput.*, 39(5):1667–1713, 2010.
30. Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, pages 294–304, 2000.
31. Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *TCC*, pages 600–620, 2013.
32. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.
33. Y. T. Kalai and R. Raz. Probabilistically checkable arguments. In *CRYPTO*, pages 143–159, 2009.
34. J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31, 1988.
35. S. Lu and R. Ostrovsky. How to garble ram programs? In *EUROCRYPT*, pages 719–734, 2013.
36. S. Micali. CS proofs (extended abstract). In *FOCS*, pages 436–453, 1994.
37. J. B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, pages 111–126, 2002.
38. A. Sahai and H. Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM Conference on Computer and Communications Security*, pages 463–472, 2010.
39. T. Sander, A. Young, and M. Yung. Non-interactive cryptocomputing for NC^1. In *FOCS*, pages 554–567, 1999.
40. A. Shamir and Y. Tauman. Improved online/offline signature schemes. In *CRYPTO*, pages 355–367, 2001.
41. R. Sion and B. Carbunar. On the practicality of private information retrieval. In *NDSS*, 2007.
42. A. C. Yao. How to generate and exchange secrets. In *FOCS*, pages 162–167, 1986.