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

Benny Applebaum* Yuval Ishai† Eyal Kushilevitz‡ Brent Waters§

December 10, 2012

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. 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.

*School of Electrical Engineering, Tel-Aviv University, bennyp@post.tau.ac.il. Supported by Alon Fellowship, ISF grant 1155/11, Israel Ministry of Science and Technology (grant 3-9094), and GIF grant 1152/2011.

†Department of Computer Science, Technion, yuvali@cs.technion.ac.il. Supported by the European Research Council as part of the ERC project CaC (grant 259426), ISF grant 1361/10, and BSF grant 2008411.

‡Department of Computer Science, Technion, eyalk@cs.technion.ac.il. Supported by ISF grant 1361/10, and BSF grant 2008411.

§Department of Computer Science, University of Texas, bwaters@cs.utexas.edu. Supported by NSF 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 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.

Contents

1 Introduction

Suppose that we want to perform some cryptographic task that 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 [19, 46], verifiable computation (delegation) [21, 6, 15], and secure computation [9, 36, 12, 17]. 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 [7]. 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 going 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 [5], 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 [48, 38, 20, 33]. Along the years, REs have found a diverse range of other applications to problems such as computing on encrypted data [45, 14], parallel cryptography [5, 4], verifiable computation [21, 6], software protection [27, 29, 10], functional encryption [44, 28], key-dependent message security [8, 1, 11], and others.

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 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 [48]. 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 1.1. *(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 however can still remain nearly linear in n. An additional (related) difference is that, while in the standard garbled circuit construction 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 $(K_{i,0}, K_{i,1})$ of 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., $R = \mathbb{Z}_p$ 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 ARE 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 [7], 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 [31, 18] 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 [40] or functional encryption [13].) 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 a concurrent and independent work of Bellare et al. [10]. The above negative result partially settles a question left open by [10].

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 [33] allows to translate our results into highly efficient MPC protocols in the offline/online setting. In Section 7.1, we further extend and optimize these reductions (exploiting the affinity property and information-theoretic techniques from [12]). 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 Section 7.1 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.[1] 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

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

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 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., [39, 27, 37, 22, 15, 6, 10]). Specifically, in [22] Yao's garbled circuit technique was used to achieve efficient VC in the online/ofline model. (The security of the construction follows from standard assumptions only when the input x is picked by the client independently of pk [10].) This connection was generalized and optimized in [6]. 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 [23, 30]. 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 using offline preprocessing. Our results yield offline/online NIZK proofs with online proof length of $|w| + \mathrm{poly}(\kappa)$ 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 [26].

1.3 Techniques

We briefly sketch some of the ideas used to prove Theorem 1.1. Our starting point is the garbled-circuit based encoding from [4]. 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).[2]

Before solving the riddle, let us further reduce it to a linear 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 one-time secure 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 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 [7] (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. See Section 5 for details.

Organization. Section 2 gives the necessary background on randomized encodings (with some additional material in Appendix A). In Section 3 we present several constructions of succinct randomized encodings for a concrete boolean function called subset function (SF). Later, in Section 4, we use these encodings as a building block and obtain succinct encodings for general boolean func-

[2]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.

tions. The arithmetic case appears in Section 5. In Section 6, we deal with some lower bounds (Section 6.1) and the issue of adaptivity (Section 6.2). In Section 7, we sketch the application of succinct randomized encodings to secure multiparty computation (MPC), non-interactive zero-knowledge proofs (NIZK), and verifiable computation (VC) in the preprocessing model.

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 [4]. 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 2.1 (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 $\{x_n\}_n$, $x_n \in \mathbb{Z}_p^n$ 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 $k \geq n^\varepsilon$ 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 depends both on x and r (*online* part). Without loss of generality, we assume that $\hat{f}$ can be written as $\hat{f}(x;r) = (\hat{f}_{\mathsf{of}}(r), \hat{f}_{\mathsf{on}}(x;r))$, where $\hat{f}_{\mathsf{of}}(r)$ 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 $\hat{f}_{\mathsf{of}}(x;r)$. 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 be $\hat{\mathcal{F}}$ 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 2.2 (**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_{of}. 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.[3] 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 2.3 (**ARE vs. DARE**). *Previous works considered a stronger form of affinity called* decomposable affine randomized encoding *(DARE).*[4] *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 $\hat{f}(x;r) = (\hat{f}_{\mathsf{of}}(r), \hat{f}_1(x_1;r), \ldots, \hat{f}_n(x_n;r))$ 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 ($O(n)$) blow-up in the online communication complexity. In Section 6.1 we show that this is inherent and decomposability* cannot *be achieved with constant rate.*

Remark 2.4 (**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 (See Definition 6.5). We will show (Lemma 6.4) 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 (Lemma 6.7).*

It is well known that REs can be manipulated via composition and concatenation [5]. These standard properties (and others) are deferred to Section A.

[3] Note that, for every r, one can always "learn" the matrix/vector M_r, v_r by solving a system of linear equations over $\mathbb{Z}_q$.

[4] In fact, in the cnference version of [7] the term ARE was used to denote DARE.

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.

The *Subset Function* g with length n and message size κ 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 $\{i : \hat{x}_i = 1\}$. (The latter convention will be implicit through the whole section.) We will always assume that κ and n are polynomially related, namely, that $\kappa = \Theta(n^\varepsilon)$ for some constant ε (typically, $\varepsilon < 1$). Our goal is to encode g by an RE of the form $\hat{g}(x, M; r) = (\hat{g}_{\mathsf{of}}(M; r), x, K(x; r))$ where $K(x; r)$ is of bit-length κ^c for some universal constant c which is independent of ε, and so $K(x; r)$ does not grow with n. We will construct such an encoding based on several assumptions. Specifically, we will show (Section 3.1) that such an encoding can be based on a special form of symmetric-key encryption with additive homomorphism. The latter can be constructed under the DDH or LWE assumption (Section 3.2). We also provide a direct RSA-based encoding for SF (Section 3.3).

3.1 ARE for SF via Additive Homomorphic Encryption

Definition 3.1 (Additive Homomorphic Encryption (AHE))**.** *An additive homomorphic Encryption is a triple of efficient algorithms* $(\mathsf{Setup}, \mathsf{E}, \mathsf{D})$ *for which the followings hold:*

- ***Syntax:*** *The randomized algorithm* Setup *takes a length parameter* 1^κ *and outputs a string* param *which specifies four groups: key-space* $\mathcal{K}$*, message-space* $\mathcal{M}$*, ciphertext-space* $\mathcal{C}$ *and public randomness space* $\mathcal{W}$*. We assume that that* κ*-bit strings can be efficiently embedded in* $\mathcal{M}$ *and denote the identity element of* $\mathcal{M}$ *by* $\mathbf{0} \in \mathcal{M}$*. We further assume that the encryption and decryption algorithms depend on* param *and on some public coins* $W \xleftarrow{R} \mathcal{W}$*, in addition to the private randomness, a message/ciphertext, and a key* K*. (We make the dependency in* param *implicit.)*

- ***One-time semantic-security:*** *Let* $\mathsf{param} = (\mathcal{K}, \mathcal{M}, \mathcal{C}, \mathcal{W}) \xleftarrow{R} \mathsf{Setup}(1^\kappa)$*. For every message* $M \in \mathcal{M}$ *we have*

$$(\mathsf{param}, W, \mathsf{E}_K(M; W)) \stackrel{c}{\equiv} (\mathsf{param}, W, \mathsf{E}_K(\mathbf{0}; W)),$$

  where $W \xleftarrow{R} \mathcal{W}$*,* $K \xleftarrow{R} \mathcal{K}$*, 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* $W \xleftarrow{R} \mathcal{W}$*, 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 AH which holds in the special case where all messages, except for one, equal to* $\mathbf{0} \in \mathcal{M}$*.*

We emphasize that key size is *independent* of the homomorphism parameter t. This is crucial for our application. We show how to encode the SF $g(M, x)$ with length n and message size κ based on AHE.

Lemma 3.2. *Assume that AHE exists. Then the SF $g(M,x)$, where $M \in (\{0,1\}^\kappa)^n, x \in \{0,1\}^n$, has an encoding*

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

where $\hat{g}_{\mathsf{of}}$ contains $O(n^2)$ ciphertexts in $\mathcal{C}$, $K_i \in \mathcal{K}$ 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. In addition, 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 $C_{i,j} = \mathsf{E}_{K_i}(M_{i,j}; W_j)$ and

$$M_{i,j} = \begin{cases} \mathbf{0} & \text{if } j \neq i, \\ M_i & \text{if } j = i. \end{cases}$$

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

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

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 hybrid argument, it suffices to show that two neighboring hybrids are 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 one-time semantic security of the scheme. Given a challenge (param, w, c) where $\mathsf{param} \xleftarrow{R} \mathsf{Setup}(1^\kappa)$ and $w \xleftarrow{R} \mathcal{W}$, the adversary $\mathcal{B}$ will distinguish between $c \xleftarrow{R} \mathsf{E}_K(\mathbf{0}; w)$ and $c \xleftarrow{R} \mathsf{E}_K(M_\ell; w))$ as follows. Use param to compute the hybrid $H_{\ell-1}$ where $(W_\ell, C_{\ell,\ell})$ take the values (w, c). It is not hard to verify that the resulting distribution is identical to $H_{\ell-1}$ if $c \xleftarrow{R} \mathsf{E}_K(\mathbf{0}; w)$, and to H_ℓ if $c \xleftarrow{R} \mathsf{E}_K(M_\ell; w)$, and the claim follows. □

3.2 AHE based on LWE and DDH

DDH. The DDH assumption asserts that there exists an efficient problem generator DDH which given a security parameter 1^κ outputs a specification param of a cyclic group $\mathbb{G}$ of order p where p is κ-bit long prime with generator α for which a random DDH tuple $(\mathsf{param}, \alpha, \alpha^a, \alpha^b, \alpha^{ab})$ is computationally indistinguishable from a random tuple $(\mathsf{param}, \alpha, \alpha^a, \alpha^b, \alpha^c)$ where $a, b, c \xleftarrow{R} \mathbb{Z}_p$.

A DDH-based AHE can be constructed via the following symmetric-key version of ElGamal encryption. The algorithms $(\mathsf{Setup}, \mathsf{E}, \mathsf{D})$ are defined via

$$\mathsf{Setup}(1^\kappa) = (\mathcal{K} = \mathbb{Z}_p, \mathcal{M} = \mathcal{C} = \mathcal{W} = \mathbb{G}) \qquad \text{where } (\mathbb{G}, \mathbb{Z}_p) \leftarrow \mathsf{DDH}(1^\kappa),$$

and

$$\mathsf{E}_K(M;W) = W^K \cdot M, \qquad \mathsf{D}_K(C;W) = C/W^K.$$

The one-time security of the scheme easily follows from the security of ElGamal public-key encryption in which a ciphertext/public-key contains in addition to $(\alpha, W, W^K \cdot M)$ also the value of α^K. Since ElGamal is secure under DDH so is the above scheme. Furthermore, it is not hard to see that the scheme satisfies relaxed homomorphism as

$$\mathsf{D}_{\sum_i K_i}\left(\prod_i \mathsf{E}_{K_i}(M_i;W);W\right) = \left(W^{\sum_i K_i} \cdot \prod_i M_i\right)/W^{\sum_i K_i}$$

which equals to M_1 assuming that all other M_i's equal to the identity element.

LWE. The Learning With Errors (LWE) assumption of [42] generalizes the Learning Parity with noise problem (LPN) and asserts that it is hard to solve a random system of noisy linear equations. Formally, let LWE be a problem generator which given a security parameter 1^κ outputs a modulus q, an integer μ and a noise-sampling circuit χ_μ which samples integers of absolute value bounded by μ. We say that the (decisional) LWE problem is *hard* if for every polynomial $t = t(\kappa)$, it holds that

$$(\mathsf{param}, W, Wk + e) \stackrel{c}{\equiv} (\mathsf{param}, W, z),$$

where

$$\mathsf{param} = (q, \mu, \chi_\mu) \stackrel{R}{\leftarrow} \mathsf{LWE}(1^\kappa), W \stackrel{R}{\leftarrow} \mathbb{Z}_q^{t\times\kappa}, k \stackrel{R}{\leftarrow} \mathbb{Z}_q^\kappa, e \stackrel{R}{\leftarrow} \chi_\mu^t, z \stackrel{R}{\leftarrow} \mathbb{Z}_q^t.$$

We will assume that the problem is hard for q which is super-polynomial in κ, e.g., $O(\kappa^{\log\kappa})$ and for some noise distribution χ_{q^α} where $\alpha \in (0,1)$ is a constant. One can define such a problem generator (e.g., by letting χ_μ be "truncated" discrete gaussian) for which the hardness of LWE follows from the worst-case hardness of approximating shortest-vector problems in a lattice of dimension κ to within a quasi-polynomial ratio $2^{\mathrm{polylog}(\kappa)}$ [42, 41]. (See also discussion in [7].)

An LWE-based AHE can be constructed via the following LWE-based symmetric-key encryption which generalizes the LPN-based construction of [24]. (See also [3].) The algorithms $(\mathsf{Setup}, \mathsf{E}, \mathsf{D})$ are defined via

$$\mathsf{Setup}(1^\kappa) = (\mathcal{K} = \mathbb{Z}_q^\kappa, \mathcal{M} = \mathbb{Z}_2^\kappa, \mathcal{C} = \mathbb{Z}_q^\kappa, \mathcal{W} = \mathbb{Z}_q^{\kappa\times\kappa}) \qquad \text{where } (q,\mu,\chi_\mu) \stackrel{R}{\leftarrow} \mathsf{LWE}(1^\kappa),$$

and

$$\mathsf{E}_K(M;W,E) = WK + e + \Delta M, \qquad \mathsf{D}_K(C;W) = \lfloor (C - WK)/\Delta \rceil,$$

where $e \stackrel{R}{\leftarrow} \chi_\mu^\kappa$, $\Delta = \lfloor q/2 \rceil$ and the operator $\lfloor\cdot\rceil$ denotes rounding to the closest integer. The one-time semantic security of the scheme follows immediately from the LWE assumption (cf. [24]). Furthermore, it is not hard to see that the scheme satisfies homomorphism. For $n = \mathrm{poly}(\kappa)$ messages, the "merged" ciphertext

$$\sum_{i=1}^n \mathsf{E}_{K_i}(M_i;W) = W\sum_{i=1}^n K_i + \Delta\sum_{i=1}^n M_i + \sum_{i=1}^n e_i$$

contains noise whose magnitude is bounded by $n \cdot q^\alpha < \Delta/2$ and therefore decryption succeeds.

Complexity. In the DDH-based AHE, the bit-length of the keys and the ciphertexts is κ, whereas the LWE-based AHE has bit-length of $\tilde{O}(\kappa)$ for the keys and $\tilde{O}(\kappa^2)$ for the ciphertexts. Hence, the encoding of Lemma 3.2 has online complexity of $n + \kappa$ (resp., $n + \tilde{O}(\kappa)$) and offline complexity of $O(n^2\kappa)$ (resp., $n^2\tilde{O}(\kappa^2)$) based on DDH (resp., LWE). (See Section 3.4 for a generic optimization.)

3.3 Encoding SF based on RSA

The RSA assumption [43] asserts that for every efficient adversary $\mathcal{A}$ of complexity $\text{poly}(\kappa)$

$$\Pr[\mathcal{A}(N, e, \alpha^e) = \alpha] \leq \text{neg}(\kappa),$$

where N is a random κ-bit RSA modulus (i.e., product of a pair of random primes p, q) e is a randomly chosen prime of length κ which is co-prime to $\varphi(N)$, and $\alpha \xleftarrow{R} \mathbb{Z}_N$. (More generally, p, q, e can be chosen according to some other distribution specified by some efficient problem generator $\mathsf{Gen}(1^\kappa)$.) We present an RSA based encoding for the subset function. We begin with an encoding for the subset function with length n and block size of 1. In this simple case, the input consists n single bit messages $m = (m_1, \ldots, m_n)$ and a selection vector $x \in \{0,1\}^n$ and it outputs the messages $(m_i)_{i \in x}$ chosen by x ,together with the selection vector x. We will later show (Lemma 3.4) that such an encoding can be upgraded to encode the SF with κ-bit messages via simple concatenation.

Lemma 3.3. *Under the RSA assumption, the simplified SF $h(m, x)$ where $x \in \{0,1\}^n$ and $m \in (\{0,1\})^n$, has an encoding of the form $\hat{h}(x, m; r) = (\hat{h}_{\mathsf{of}}(m; r), x, K(x; r))$ where $\hat{h}_{\mathsf{of}}$ is of bit-length $n\kappa$ and $K(x; r)$ is of length κ.*

Proof. The encoding $\hat{h}$ relies on a symmetric variant of RSA encryption. At the offline phase, generate a random RSA modulus N together with its factorization p and q, choose a random $u \xleftarrow{R} \mathbb{Z}_N$ and n random primes $e_1, \ldots, e_n$ of length κ which are all co-prime to $\varphi(N)$), and a string $r \xleftarrow{R} \mathbb{Z}_2^\kappa$. For every $i \in [n]$ compute

$$y_i = u^{1/e_i}, \quad C_i = m_i \oplus \text{hc}(r, y_i),$$

where hc is the Goldreich-Levin hardcore predicate (i.e., inner-product over $\mathbb{Z}_2$). The offline part of the encoding is $(N, u, (e_1, \ldots, e_n), r, (C_1, \ldots, C_n))$, and the online part is the pair $(x, v = u^{\prod_{i \in x} 1/e_i})$.

Decoding. For $i \in x$, recover y_i by computing $v^{\prod_{j : \in (x \setminus i)} e_j} = u^{e_i}$ and let m_i be $C_i \oplus \text{hc}(r, y_i)$.

Simulation. Fix some $x \in \{0,1\}^n$ and $m = (m_1, \ldots, m_n) \in \{0,1\}^n$. Given $(x, (m_i)_{i \in x})$ we simulate the distribution $\hat{h}(m, x; r)$ by sampling $\hat{h}(m', x; r)$ where m'_i equals to m_i if $i \in x$, and $m'_i \xleftarrow{R} \{0,1\}$ otherwise. We prove that $\hat{h}(m, x; r)$ is computationally indistinguishable from $\hat{h}(m', x; r)$ via a hybrid argument as follows.

Security. For $i \in [n]$ define the hybrid distribution H_i by $\hat{h}((m_{1:i} | m'_{i+1:n}), x; r)$. Clearly, H_0 corresponds to the simulated distribution while H_n corresponds to the distribution of the real encoding. Suppose there exists a distinguisher $\mathcal{A}$ that distinguishes $H_{\ell-1}$ from H_ℓ with non-negligible advantage ε. Observe that it must be the case that $\ell \notin x$; otherwise the distributions $H_{\ell-1}$ and H_ℓ are the identical. We use $\mathcal{A}$ to solve an RSA challenge $(N, e, z = \alpha^e)$ as follows. First, for every $i \neq \ell$ we choose a random κ-bit prime e_i and let

$$u = z^{\prod_{i \neq \ell} e_i}, \quad v = z^{\prod_{i \neq \ell, i \notin x} e_i}, \qquad y_i = z^{\prod_{j \neq \ell, i} e_j}.$$

Letting $e_\ell = e$ we can write

$$u = \alpha^{\prod_i e_i}, \quad v = \alpha^{\prod_{i \notin x} e_i}, \qquad y_i = \alpha^{\prod_{j \neq i} e_j}.$$

Let us condition on the event that all the e_i's are co-primes to $\varphi(N)$ (which happens with all but negligible probability). In this case, the e_i's are distributed exactly as in the real encoding, and u is uniformly and independently distributed in $\mathbb{Z}_N$ (since $\alpha \xleftarrow{R} \mathbb{Z}_N$ and exponentiation to the power

of $\prod_i e_i$ induces a permutation over $\mathbb{Z}_N$). Furthermore, v equals to $u^{\prod_{i\in x}}$ and y_i equals to u^{1/e_i} for $i \neq \ell$. Hence, the joint distribution of N, u, the e_i's, and the y_i's is exactly as in the real encoding. The only missing information is y_ℓ which should take the value $u^{1/e_\ell} = \alpha^{\prod_{i\neq\ell} e_i}$. We will use $\mathcal{A}$ to recover u^{1/e_ℓ} and then use it to find α.

By the Goldreich-Levin theorem, in order to recover u^{1/e_ℓ} it suffices to construct an algorithm $\mathcal{B}$ that distinguishes the pair $(r \xleftarrow{R} \mathbb{Z}_2^\kappa, \sigma = \mathrm{hc}(u^{1/e_\ell}, r))$ from the pair $(r \xleftarrow{R} \mathbb{Z}_2^\kappa, \sigma \xleftarrow{R} \{0,1\})$ with noticeable advantage. To achieve this we let $\mathcal{B}(r,\sigma)$ be the outcome of $\mathcal{A}(N, u, (e_i)_{i\in[n]}, r, (C_i)_{i\in[n]}, x, v)$ where

$$C_i = \begin{cases} m_i \oplus \mathrm{hc}(y_i, r) & \text{if } i < \ell \text{ or } i \in x, \\ m_i \oplus \sigma & \text{if } i = \ell, \\ R_i \xleftarrow{R} \{0,1\} & \text{if } i > \ell \text{ and } i \notin x \end{cases}.$$

It is not hard to verify that, when $(N, z, e_1, \ldots, e_n, r)$ are uniformly chosen, the resulting distribution corresponds to H_ℓ if $\sigma = \mathrm{hc}(u^{1/e_\ell}, r)$, and to $H_{\ell-1}$ if $\sigma \xleftarrow{R} \{0,1\}$. Hence, by Markov's inequality, with probability at least $\varepsilon/2$ the tuple $(N, z, e_1, \ldots, e_n)$ is *good* in the sense that $\mathcal{B}$ has distinguishing advantage of $\varepsilon/2$. Therefore, we recover u^{1/e_ℓ} with noticeable probability. Finally, we employ Shamir's algorithm [47] which, given $X, Y \in \mathbb{Z}_N$ and relatively prime integers a, b for which $X^a = Y^b$, efficiently computes $Y^{1/a}$. Letting $X = u^{1/e_\ell}, Y = z$ and $a = e_\ell, b = \prod_{j\neq\ell} e_j$, we recover the RSA solution z^{1/e_ℓ}. □

The above encoding can be easily used to encode the subset function with κ-bit messages.

Lemma 3.4. *Under the RSA assumption, the SF $g(M,x)$ where $x \in \{0,1\}^n$ and $M \in (\{0,1\}^\kappa)^n$, has an encoding of the form $\hat{g}(x, M; r) = (\hat{g}_{\mathsf{of}}(M; r), x, K(x; r))$ where $\hat{g}_{\mathsf{of}}$ is of bit-length $n\kappa^2$ and $K(x; r)$ is of length κ^2.*

Proof. Observe that $g(M,x)$ is (deterministically) encoded by $(h(x, m^i))_{i=1}^\kappa$ where $m^i = (M_{1,i}, \ldots, M_{n,i})$. By the concatenation lemma, the latter function can be encoded by concatenating the encodings $\hat{h}(x, m^i; r^i)$ for $i \in [\kappa]$ from Lemma 3.3. By the composition lemma, the resulting function also encodes g. This encoding almost satisfies the lemma except that there are κ copies of x. These multiple copies can be replaced by a single copy (formally, think of x as a deterministic encoding of its copies and invoke the composition lemma) leading to an encoding that satisfies the lemma. □

Remark 3.5 (Optimization)**.** *Suppose that RSA is secure against 2^τ-time adversaries. In this case, one can extract $t = \Theta(\tau)$ independent pseudorandom bits by using t independent copies of the Goldreich-Levin hardcore predicate. Applying this optimization to Lemma 3.3 (i.e., replacing the single hardcore bit with t hardcore bits $\mathrm{hc}(r_1, y), \ldots, \mathrm{hc}(r_t, y)$) allows to encode the SF with message length of t without increasing the overhead. Therefore, by using κ/t-time concatenation (as in Lemma 3.4), the SF with message length κ can be encoded with offline complexity (bit-length of $\hat{g}_{\mathsf{of}}$) of $n\kappa^2/t$ and online complexity (bit-length of $K(x; r)$) of κ^2/t. Specifically, assuming that RSA is subexponentially hard against 2^{κ^ε}-time adversaries, the offline complexity becomes $\Theta(n\kappa^{2-\varepsilon})$ and the online complexity is $\Theta(\kappa^{2-\varepsilon})$.*

3.4 Reducing the Offline Complexity of SF

Given a succinct encoding for SF, one can obtain a new encoding with a smooth tradeoff between the online and the offline complexity as follows. Let $h(M', x')$ be the subset function with length N and message size κ, i.e., $M' \in (\{0,1\}^\kappa)^N, x' \in \{0,1\}^N$, and assume that we have an encoding $\hat{h}(M', x') = (\hat{h}_{\mathsf{of}}(M'; r), x', K'(x'; r))$ where $\hat{h}_{\mathsf{of}}$ has of computational complexity of $N^a\kappa^{a'}$ and K' is

of length κ^b and computational complexity of $N\kappa^b$ for some constants a, a', b.[5] In order to encode the SF $g(M, x)$ with input length n, partition the input to n/N blocks of size N each, i.e., let $M^i = (M_{iN+1} \dots M_{(i+1)N})$ and $x^i = (x_{iN+1} \dots x_{(i+1)N})$, and think of $g(M, x)$ as the concatenation of $h(M^i, x^i)$ for $i = 1, \dots, n/N$. Now encode g by the encoding

$$\hat{g}(M, x; (r^1, \dots, r^{n/N})) = (\hat{h}(M^i, x^i; r^i))_{i=1}^{n/N}$$

reordering the outputs and letting $r = (r^1, \dots, r^{n/N})$, we can write $\hat{g}(M, x; r)$ as $\hat{g}_{\mathsf{of}}(M; r), x, K(x; r))$. Let $\mathsf{Comp}(f)$ denote the computational complexity (circuit size) of a function f, and $\mathsf{Len}(f)$ denotes the output length of f (in bits). Then, the complexity of the new encoding satisfies

$$\begin{aligned}
\mathsf{Comp}(\hat{g}_{\mathsf{of}}) &= \frac{n}{N} \cdot \mathsf{Comp}(\hat{h}_{\mathsf{of}}) = nN^{a-1}\kappa^{a'} \\
\mathsf{Len}(K) &= \frac{n}{N} \cdot \mathsf{Len}(K') = \frac{n}{N}\kappa^b \\
\mathsf{Comp}(K) &= \frac{n}{N} \cdot \mathsf{Comp}(K') = n\kappa^b
\end{aligned}$$

Hence, a larger value of N reduces the online complexity, while a smaller value reduces the offline complexity. By letting $N > \omega(\kappa^b)$, the online communication remains $n + o(n)$ (as κ is polynomially related to n). Combining the above with Lemma 3.4 and Lemma 3.2 and the LWE/DDH constructions from Section 3.2, we derive the following lemma:

Lemma 3.6 (Encoding SF)**.** *Assume that the DDH assumption, or LWE assumption or the RSA assumption holds. There exists a universal constant C such that for every n and κ for which $n^{\Omega(1)} < \kappa < o(n)$ the Subset Function $g(M, x)$ with length n and message size κ has an RE of the form $\hat{g}(x, M; r) = (\hat{g}_{\mathsf{of}}(M; r), x, K(x; r))$ where:*

- *$K(x; r)$ is of bit-length at most $o(n)$.*
- *The encoding $\hat{g}$ (including both the online and offline parts) can be computed in time $n\kappa^C$.*
- *In the case of DDH and LWE $K(x; r)$ is affine in x (for every fixed value of r).*

4 Succinct AREs for Boolean Circuits

In this section we will encode any efficiently computable function via a succinct encoding based on a succinct encoding for the subset function g. Recall that the *Subset Function* g with length N and message size κ is defined by

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

where $M = (M_1, \dots, M_N) \in (\{0,1\}^\kappa)^N$ is a vector of N "messages", and $\hat{x} \in \{0,1\}^N$ is a selection vector. (We renamed the inputs and input-lengths to avoid confusion).

Let $F : \{0,1\}^n \to \{0,1\}^\ell$ be an efficiently computable function and fix some polynomially-related security parameter $\kappa = n^\varepsilon$. We will show how to succinctly encode the function $F(x)$ based on a succinct encoding for the Subset Function with length $N = 2n$ and message size κ. Jumping ahead, $\hat{x}$ will "encode" the vector x and therefore we will think of it as the online input, and on M as an offline part.

[5] For example, for DDH $a = 2, a' = 1$ and $b = 1$, for LWE $a = 2, a' = 2$ and $b = 1$, and for RSA $a = 1, a' = 2$ and $b = 2$.

Lemma 4.1. *Assume that one-way functions exist and that g has an RE of the form $\hat{g}(\hat{x}, M; r) = (\hat{g}_{\mathsf{of}}(M; r), \hat{x}, K(\hat{x}; r))$. Then, F has an RE $\hat{F}$ with (1) online complexity of $n + \mathsf{Len}(K)$; (2) Online computational complexity of $O(n + \mathsf{Comp}(K)$; (3) Offline computational complexity of $O(|F|\kappa) + \mathsf{Comp}(\hat{g}_{\mathsf{of}}(M; r))$. Furthermore, if $K(\hat{x}; r)$ is affine then so is $\hat{F}$.*

Recall that $\mathsf{Comp}(f)$ denote the computational complexity (circuit size) of a function f, and $\mathsf{Len}(f)$ denotes the output length of f (in bits).

Proof. In [4] it is shown that, assuming the existence of one-way functions, $F(x)$ can be encoded by a Decomposable ARE $f(x; \rho) = (f_{\mathsf{of}}(\rho), f_{\mathsf{on}}(x; \rho))$ where $f_{\mathsf{on}}(x; \rho)$ is essentially the selection function which generates $2n$ pairs of κ-bit "keys" $(v_i^0(\rho), v_i^1(\rho))_{i\in[n]}$, and outputs the keys $(v_i^{x_i})_{i\in[n]}$ indexed by x. By the concatenation and composition lemmas (Facts A.2 and A.3), it suffices to encode the online-part $f_{\mathsf{on}}(x; \rho)$ (viewed as a single argument function) by a succinct encoding $\hat{f}_{\mathsf{on}}(x, \rho; R)$. Indeed, in this case $F(x)$ can be encoded by $\hat{F}(x; (\rho, R)) = (f_{\mathsf{of}}(\rho), \hat{f}_{\mathsf{on}}(x; (\rho, R)))$. Furthermore, by the substitution lemma (Fact A.1), we may simply encode the selection function $(x, V) \to (v_i^{x_i})_{i\in[n]}$ where $V = (v_i^0, v_i^1)_{i\in[n]}$ and $v_i^b \in \{0,1\}^\kappa$. For simplicity, we abuse notation and identify this function with f_{on}.

Claim 4.2. *The selection function f_{on} can be perfectly encoded by an encoding of the form*

$$h : (V, x; s) \mapsto g(M, \hat{x})$$

where $s \xleftarrow{R} \{0,1\}^n$, $M \in (\{0,1\}^\kappa)^{2n}$ depends on V and the randomness s (but not on x) and $\hat{x} \in \{0,1\}^{2n}$ equals to $(x \oplus s \oplus \mathbf{1}, x \oplus s)$.

First we show that the claim implies the lemma. Consider the encoding

$$\hat{h} : ((V, x); (s, r)) \mapsto \hat{g}(M, \hat{x}; r) = (\hat{g}_{\mathsf{of}}(M; r), \hat{x}, K(\hat{x}; r))$$

where M and $\hat{x}$ are computed as in the claim. Then, by the substitution lemma (Fact A.1), $\hat{h}$ encodes the function h, and so, by the composition lemma (Fact A.3), it also encodes f_{on}. Observe that $\hat{h}(x, V; (r, s))$ consists of $2n + \kappa$ bits which depend on x as $\hat{x} \in \{0,1\}^{2n}$. This can be reduced to $n + \kappa$ by replacing $\hat{x}$ with $x \oplus s$. Call this modified function $\hat{f}_{\mathsf{on}}$. Since $x \oplus s$ (determinstically) encodes $\hat{x}$, the function $\hat{f}_{\mathsf{on}}$ perfectly encodes $\hat{h}$ and so by the composition lemma (Fact A.3), we obtain an encoding of f_{on}. Overall, the encoding $\hat{F}(x; (\rho, s, r)) = (f_{\mathsf{of}}(\rho, s, r), \hat{f}_{\mathsf{on}}(x; (s, r)))$ satisfies the properties of the lemma. □

It remains to prove Claim 4.2.

Proof of claim. For each $i \in [n]$ we choose a random bit s_i, and let $L_i = v_i^{s_i}$ and $R_i = v_i^{1-s_i}$. Let $\hat{x}_L = x \oplus s \oplus \mathbf{1}$ and let $\hat{x}_R = x \oplus s$. It is not hard to verify that the vector $(L_i)_{i\in\hat{x}_L}$ contains exactly the entries v^{s_i} for which $x_i = s_i$. Similarly, $(R_i)_{i\in\hat{x}_R}$ contains the entries $(v^{1-s_i})_{i:x_i=1-s_i}$. Hence, we encode the function f by the function $g(M, \hat{x})$ where $\hat{x} = (\hat{x}_L, \hat{x}_R)$ and $M = (L_1, \ldots, L_n, R_1, \ldots, R_n)$. We already saw that the value $v_i^{x_i}$ appears in $(M_i)_{i\in\hat{x}}$ if $\hat{x}_i = 1$, whereas $v_i^{x_i}$ appears if $\hat{x}_i = 0$. Hence, it is possible to decode the value f from the encoding $\hat{f}$. As for privacy, given $(v_i)_{i\in[n]}$ we can simulate the pair $(Z, \hat{x})$ by letting $Z = (v_i)_{i\in[n]}$ and $\hat{x} = (s, \mathbf{1} \oplus s)$ where $s \xleftarrow{R} \{0,1\}^n$. It is not hard to verify that the simulation is perfect. □

We can now apply Lemma 4.1 with our encodings for SF and derive succinct encodings for general boolean functions. For example, combining Lemma 4.1 with the optimized encoding of Lemma 3.6 we derive the following theorem:

Theorem 4.3 (Theorem 1.1 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 the following properties:*

- *The online communication is $n + o(n)$ and the online computational complexity is $O(n^{1+\varepsilon})$*
- *The offline computational/communication complexity is $O(n^{\varepsilon}|F|)$ where $|F|$ is the circuit size of F.*
- *In the case of LWE and DDH the encoding is affine.*

Proof. Let $\kappa = n^{\delta}$ for sufficiently small constant δ whose value will be determined later. By Lemma 3.6 we obtain an encoding for the $\hat{g}$ that satisfies the properties of Lemma 4.1. Furthermore, $K(x;r)$ is of bit-length at most $o(n)$ and the encoding is computable in time $O(n\kappa^C)$ for some universal constant C. Applying Lemma 4.1 we obtain an encoding with online communication of $n + o(n)$, online complexity of $O(n\kappa^C)$ and offline complexity of $O(|F|\kappa) + O(n\kappa^C)$. The theorem now follows by letting $\delta = \varepsilon/(2C)$. □

Remarks.

- (Reduction) The proof of Lemma 4.1 shows that the task of succinctly encoding a function F that admits an online efficient DARE reduces (information-theoretically) to the task of succinctly encoding the subset function g.
- (General compiler) Theorem is constructive, i.e., it describes a compiler that given a circuit for F outputs a description of the encoding $\hat{F}$, its decoder and simulator.
- (Online inputs) The theorem generalizes to the case where the function f has some online inputs x_{on} and offline inputs x_{of} as in Remark 2.2. Namely, the part of the encoding $\hat{f}$ which depends on x_{on} is of length $|x_{\mathsf{on}}| + o(|x_{\mathsf{on}}|)$.

5 Succinct ARE for Arithmetic Formulas

In this section we construct a succinct ARE for arithmetic formulas over subexponentially large modulus p (i.e., $p = 2^{o(n)}$) as follows.

Theorem 5.1. *Let $F : \mathbb{Z}_p^n \to \mathbb{Z}_p^{\ell}$ be an efficiently-computable arithmetic formula where $p = \Theta(2^{n^{\delta}})$ for some $\delta \in (0,1)$. Then, assuming LWE, F has a succinct ARE $\hat{F}$ over $\mathbb{Z}_q$ where $q = \Theta(2^{n^{\delta'}})$ for some $\delta' \in (\delta, 1)$ of the following form:*

$$\hat{F}(x;r,s) = (\hat{F}_{\mathsf{of}}(r), \hat{x} = x + s \pmod p, K(\hat{x}, r)),$$

where $K(\hat{x}, r)$ is short (of length $o(n \log p)$) and affine over $\mathbb{Z}_q$.

We note that by lifting mod-q computations to the integers, we can get an encoding over the integers with constant rate. The theorem will be proven in two steps. In Section 5.1, we show that it suffices to obtain a succinct encoding for the mod-q *universal affine function* (AF), and in Section 5.2, we construct such an encoding under the LWE assumption.

5.1 Reduction to the Affine Function

Let $F : \mathbb{Z}_p^n \to \mathbb{Z}_p^\ell$ be an efficiently-computable arithmetic formula where $p = \Theta(2^{n^\delta})$ for some $\delta \in (0,1)$. Let $m = m(n)$ be some polynomial (whose value will be related to the size of F). The *universal affine function* (AF) $g = g_{n,m,p}$ over $\mathbb{Z}_p$ is defined by

$$g(M, v, \hat{x}) \mapsto (M\hat{x} + v, \hat{x}), \quad \text{where } M \in \mathbb{Z}_p^{m \times n}, v \in \mathbb{Z}_p^m, \hat{x} \in \mathbb{Z}_p^n.$$

Lemma 5.2 (Succinct ARE for mod-p formulas)**.** *The function F has a succinct ARE $\hat{F}$, assuming that the function g has an ARE $\hat{g}$ of the form*

$$\hat{g}(M, v, \hat{x}; r) = (\hat{g}_{\mathsf{of}}(M, v; r), \hat{x}, K(\hat{x}; r))$$

where $K(\hat{x}; r) \in \mathbb{Z}_q^\kappa$ is an affine function in $\hat{x}$ and $\kappa \log q = o(n \log p)$.

Proof. Let $F : \mathbb{Z}_p^n \to \mathbb{Z}_p^\ell$ be a function computable by $s = \text{poly}(n)$-size arithmetic formula. In [34, 5] (see also [16]) the function F is information-theoretically encoded via an ARE $f(x; \rho) = (f_{\mathsf{of}}(\rho), f_{\mathsf{on}}(x; \rho))$ over $\mathbb{Z}_p$. Namely, $f_{\mathsf{on}}(x; \rho) = M_\rho x + v_\rho$ where $M_\rho \in \mathbb{Z}_p^{m \times n}$ and $v_\rho \in \mathbb{Z}_p^m$ are computed based on the randomness ρ, and $m = O(\ell s^2)$.

By the composition and concatenation lemmas (Facts A.2 and A.3), it suffices to encode the online-part $f_{\mathsf{on}}(x; \rho)$ (viewed as a single argument function) by a succinct encoding $\hat{f}_{\mathsf{on}}(x, \rho; R)$. Indeed, in this case $F(x)$ can be encoded by $\hat{F}(x; (\rho, R)) = (f_{\mathsf{of}}(\rho), \hat{f}_{\mathsf{on}}(x; (\rho, R)))$. Furthermore, by the substitution lemma (Fact A.1), we may simply encode the affine function $L : (M, v, x) \to Mx+v$. Observe that unlike g the function L does not reveal x. However, one can succinctly encode L by g as follows.

Claim 5.3. *The function $L(M, v, x)$ is perfectly encoded via the encoding*

$$h : (M, v, x; s) \mapsto g(M, v', \hat{x}), \qquad \text{where } s \xleftarrow{R} \mathbb{Z}_p^n, \quad v' = v - Ms, \quad \hat{x} = x + s.$$

Proof. Decoding is trivial as $M\hat{x} + v' = Mx + v$. Given an output y of L we perfectly simulate $h(M, v, x; s)$ via (y, z) where $z \xleftarrow{R} \mathbb{Z}_p^n$. □

We can now complete the proof of the lemma. Consider the encoding

$$\hat{h} : ((M, v, x, s); r)) \mapsto \hat{g}(M, v' = v - Ms, \hat{x} = x + s; r) = (\hat{g}_{\mathsf{of}}(M, v'; r), \hat{x}, K(\hat{x}; r)).$$

Then, by the substitution lemma (Fact A.1), $\hat{h}$ encodes the function h, and so, by the composition lemma (Fact A.3), it also encodes L. Combining everything together, we encode the function F via an ARE

$$\hat{F}(x; (\rho, s, r)) = (f_{\mathsf{of}}(\rho), \hat{g}_{\mathsf{of}}(M_\rho, v_\rho - M_\rho s; r), x + s, K(x + s; r))$$

with optimal online rate, as promised. □

5.2 ARE for the Universal Affine Function

Our goal is to encode the universal affine function $g_{n,m,p}$. We will make use of the following fact which "lifts" g to the integers.

Fact 5.4. *The function $g_{n,m,p}(M, v, x)$ can be encoded via the function*

$$(M, v, x; R) \mapsto (Mx + M_0, x)$$

where $R \xleftarrow{R} [0, p^2]^m$ $M_0 = v + pR$ addition and multiplication are computed over the integers. The encoding has statistical privacy error of $O(mnp^2/p^3)$ which is negligible in n.

The fact follows from [7, Lemma 5.2]. Hence, it suffices to construct a succinct encoding for the function $g'(M, M_0, x) = (Mx + M_0, x)$ computed over the integers where $M \in [0 : p-1]^{m \times n}, M_0 \in [0 : 2p^3]^m$ and $x \in [0 : p]^n$. We will construct such an encoding based on the LWE problem with the following parameters.

Parameters. Let $\alpha, \varepsilon_1, \varepsilon_2 \in (0, 1)$ be constants such that $\varepsilon_1\varepsilon_2 > \delta$ (recall that $p = \Theta(2^{n^\delta})$). We will base our encoding on the assumption that LWE is hard for dimension $\kappa = n^{\varepsilon_1}$, modulus $q = \Theta(2^{\kappa^{\varepsilon_2}})$, and noise distribution χ_{q^α} which samples integers bounded by q^α. In fact, we will need a family of distributions χ_μ which are almost invariant under small shifts. Namely, for every integer B the statistical distance between χ_μ and $\chi_\mu + B$ is bounded by $\mathrm{poly}(|B|/\mu)$. Standard noise distributions (e.g., discrete gaussian, or uniform over μ-size interval) have this property (cf. [42, 7]). Overall, for proper choice of parameters our assumption is implied by the worst-case hardness of approximating the shortest vector in a κ-dimensional lattice to within a subexponential ratio.

The encoding. We will use a variant of the LWE-based gadget from [7]. Let $\beta \in (\alpha, 1)$ and $\Delta = 3npq^\beta$. At the offline phase, select a random public matrix $W \xleftarrow{R} \mathbb{Z}_q^{m \times \kappa}$, a random secret key matrix $K \xleftarrow{R} \mathbb{Z}_p^{\kappa \times n}$ and a noise matrix $E \xleftarrow{R} \chi_{q^\alpha}^{m \times n}$. Then compute an $m \times n$ "padding" matrix $Y = WK + E \pmod q$ and an $m \times (n+1)$ "ciphertext" matrix $C = Y + \Delta \cdot M \pmod q$. In addition, let $K_0 \xleftarrow{R} \mathbb{Z}_q^\kappa$ and $E_0 \xleftarrow{R} \chi_{q^\beta}^\kappa$ and compute $C_0 = WK_0 + E_0 + \Delta \cdot M_0$. The output is

$$\hat{g}_{\mathsf{of}}(M, M_0) = (W, C, C_0), \qquad \hat{g}_{\mathsf{on}}(x) = (x, \hat{K} = K \cdot x + K_0 \pmod q)$$

Decoding. Given $(W, C, C_0, x, \hat{K})$ we decode the integer vector $Mx + M_0$ as follows: (1) compute $M' = Cx + C_0 - W\hat{K}$ over the integers; (2) To "clean" the noise divide each entry of M' by Δ and round to the closest integer. Output the resulting vector $\lfloor M'/\Delta \rceil$.

Claim 5.5. *The outcome of the decoder equals to $Mx + M_0$ over the integers.*

Proof. First observe that over $\mathbb{Z}_q$

$$M' = Cx + C_0 - W\hat{K} = WKx + Ex + WK_0 + E_0 + \Delta(Mx + M_0) - WKx = Ex + E_0 + \Delta(Mx + M_0).$$

Moreover, this equality also holds over the integers since $q > \Omega(2^{n^{\delta'}})$ where $\delta' > \delta$, while the absolute value of each entry in $Ex + E_0 + \Delta(Mx + M_0)$ (computed over the integers) is smaller than $O(npq^\alpha + q^\beta + \Delta(np^2 + p^3)) = o(q)$. Hence, q is large enough to ensure that there is no wraparound and the outcome of the first step is $Ex + E_0 + \Delta(Mx + M_0)$ over the integers. Now observe that Δ is large enough to ensure that there is no rounding error in the second step. To see this, note that each entry of $Ex + E_0$ is bounded by $npq^\alpha + q^\beta < \Delta/2$. We conclude that decoding succeeds with probability 1. □

Simulation. Given $(x, z = Mx + M_0)$ we simulate $\hat{g}$ as follows. Choose $W \xleftarrow{R} \mathbb{Z}_q^{m \times \kappa}$, $C \xleftarrow{R} \mathbb{Z}_q^{m \times n}$ and $\hat{K} \xleftarrow{R} \mathbb{Z}_q^\kappa$ and let

$$C_0 = W\hat{K} + E_0 + \Delta z - Cx \pmod q,$$

where $E_0 \xleftarrow{R} \chi_\beta^\kappa$. Output $(W, C, C_0, x, \hat{K})$.

Claim 5.6. *For every M, M_0, x the simulated distribution $\mathsf{Sim}(x, Mx + M_0)$ is computationally indistinguishable from the encoding $\hat{g}(M, M_0, x)$.*

Proof. It is not hard to show that, under the LWE assumption, the uniform distribution $(W \xleftarrow{R} \mathbb{Z}_q^{m\times\kappa}, Y \xleftarrow{R} \mathbb{Z}_q^{m\times n})$ is computationally indistinguishable from the following "product"-LWE distribution

$$(W \xleftarrow{R} \mathbb{Z}_q^{m\times\kappa}, Y = WK + E) \text{ where } K \xleftarrow{R} \mathbb{Z}_q^{\kappa\times n}, E \xleftarrow{R} \chi_{q^\alpha}^{m\times n}.$$

This can be proved via a standard hybrid argument, cf. [2]. We will show that if the claim does not hold then the above distributions can be distinguished.

Fix some (M, M_0, x) and assume, towards a contradiction, that there exists an adversary $\mathcal{A}$ that distinguishes $\mathsf{Sim}(x, Mx + M_0)$ from $\hat{g}(M, M_0, x)$ with advantage ε. We will use $\mathcal{A}$ to distinguish between the product LWE distribution and the uniform distribution with advantage $\varepsilon - \mathrm{neg}(n)$. Given a challenge (W, Y) compute $C = Y + Mx \pmod q$, $\hat{K} \xleftarrow{R} \mathbb{Z}_q^\kappa$ and $C_0 = W\hat{K} + E_0 + \Delta z - Cx \pmod q$ and output $\mathcal{A}(W, C, C_0, x, \hat{K})$.

It is not hard to verify that when the input (W, Y) is uniform in $\mathbb{Z}_q^{m\times\kappa} \times \mathbb{Z}_q^{m\times n}$ the tuple $(W, C, C_0, x, \hat{K})$ is distributed identically to $\mathsf{Sim}(x, Mx + M_0)$. Now assume the input is sampled according to the LWE distribution, i.e., $W \xleftarrow{R} \mathbb{Z}_q^{m\times\kappa}$ and $Y = WK + E$ where $K \xleftarrow{R} \mathbb{Z}_q^{\kappa\times n}$ and $E \xleftarrow{R} \chi_{q^\alpha}^{m\times n}$. We claim that the tuple $(W, C, C_0, x, \hat{K})$ is statistically close to $\hat{g}(M, M_0, x)$.

First observe that the joint distribution of (W, K, E, C) is statistically close in both experiments. (The two are not identical due to the fact that the encoding samples the error matrix E conditioned on being small – however, this increases the statistical distance by a negligible quantity.) Fix (W, K, E, C) and observe that in both experiments, $\hat{K}$ is uniformly distributed since in the encoding $\hat{K} = Kx + K_0$ where $K_0 \xleftarrow{R} \mathbb{Z}_q^\kappa$). Fix K_0 as well. Finally, since $K_0 = \hat{K} - Kx$ the value of C_0 in the encoding can be written as

$$C_0 = WK_0 + E_0 + \Delta\cdot M_0 = W(\hat{K} - Kx) + E_0 + \Delta(M_0 + Mx) - (WK + E + M)x + WKx + Ex, \pmod q$$

rearranging and substituting $z = Mx + M_0$ we get

$$C_0 = W\hat{K} + E_0 + Ex + \Delta z - Cx \pmod q.$$

Since each entry of Ex is bounded by $B = npq^\alpha$ and since $E_0 \xleftarrow{R} \chi_{q^\beta}$, the statistical distance between $E_0 + Ex$ and E_0 is at most $m\mathrm{poly}(B/q^\beta) = \mathrm{neg}(n)$. Hence, C_0 as computed by the algorithm is statistically-close to the distribution of C_0 in the encoding and the claim follows. □

6 More on Online/Offline Encodings

6.1 Some Lower Bounds

Lemma 6.1 (**DARE have super-constant online-rate**). *There exists a function f such that any $(t, \frac{1}{2})$ decomposable encoding $\hat{f}$ of f has online rate larger than $k = \log(t - s)/2$ where s is the complexity of the decoder.*

Specifically, super-polynomial security implies lower-bound of $\omega(\log n)$, and sub-exponential security 2^{n^ε} implies polynomial rate of n^ε which matches the construction of [4]. We also mention that the proof actually holds for a stronger statement as it rules out even an extremely poor distinguishing advantage ε which approaches to 1 exponentially fast (e.g., $1 - 2^{n/2}$ as long as $k = \min(\log(t - s)/2, n/4)$).

Proof. Let $f(x,y) = (x_1 + x_2 + \cdots + x_n) \cdot y$ where $x, y \in \mathbb{F}_2^n$ and multiplication is understood as a multiplication of a vector y by the scalar $\sum x_i$ over $\mathbb{F}_2$. It suffices to show that every input x_i affects at least k output bits of $\hat{f}(x,y;r)$, as the decomposablity of $\hat{f}$ implies that in this case the online communication complexity is at least nk. Say that x_i affects a set S of at most k outputs. Consider a uniformly chosen y and $x = 0^n$, we show how to distinguish in this case the distributions $\hat{f}((x,y);r)$ from $\mathsf{Sim}(f(x,y))$. Given z (sampled from one of the above distributions), enumerate all 2^k strings z' which differ from z only with respect to the indices which are influenced by x_i, and apply the decoder Dec to each of the modified strings. If the outcome corresponds to y the distinguisher outputs "pass" and otherwise it outputs "fail". (Recall that distinguishing should be hard even if the inputs x, y are known.)

We claim that when z is the outcome of the simulator the test passes with negligible probability. Indeed, the input to the simulator is independent of y (the simulator gets $f(x,y) = 0$), and therefore for every fixed outcome of the simulator z the probability that the modified string z' passes the test is at most $2^k/2^n$.

On the other hand, if z is the outcome of $\hat{f}((x,y);r)$ then: (1) the string $z' = \hat{f}((e_i,y);r)$, where e_i is the i-th unit vector, differs from z only on (subset of) the coordinates which are influenced by x_i; and (2) $\mathsf{Dec}(z') = f(x') = y$ and so the test passes with probability 1. Since the complexity of the test is $2^k + s < t$ and since it has a distinguishing advantage of $1 - 2^k/2^n$ the lemma follows. □

We now prove a lower bound on the online rate of AREs. It is clear that the online rate needs to be at least 1 for functions with a long output (such as the identity function). We show that this is also the case for some boolean functions.

Lemma 6.2 (**ARE have online rate of at least 1**). *There exists a boolean function $f : \{0,1\}^n \to \{0,1\}$ such that the online rate of any ARE of f (over an arbitrary ring) is at least $1 - o(1)$.*

Proof. Let $n = k + \log k$. We will show that the "universal" function $f(x,i) = x_i$, where x is in $\{0,1\}^k$ and i is in $\{0,1\}^{\log k}$, cannot be encoded by an ARE $\hat{f}(x,i;r)$ with online communication complexity smaller than k. Indeed, fix some randomness r and let $\hat{f}(x,i;r) = (F, A_r \cdot x + A'_r \cdot i + b)$ be a perfectly correct ARE of f over $\mathbb{Z}_q$ where F is the offline part. Then it is possible to fully recover x from $A_r x$ (by computing $A'_r i + b$ for $i = 1, \ldots, n$), from which it follows that the bit-length of $A_r x$ is at least k. □

While we cannot unconditionally prove a similar result for non-affine REs with, say, quadratic online computation, such a negative result follows from the (conjectured) impossibility of compression. Formally, we say that a function f is *compressed* by a function g if: (1) (lossless recovery) there exists a recovery function h such that for every input x, $h(g(x)) = f(x)$; and (2) (g is shrinking) for every x the length of $g(x)$ is shorter than the length of x. We conjecture that for every positive constants $c > c'$, there exists a function computable in time n^c that cannot be compressed by a time-$(n^{c'})$ function g. (See [31, 18] for related conjectures.)

Lemma 6.3 (**Incompressibility ⇒ RE have online rate of at least 1**). *Suppose that the above incompressibility conjecture holds. The class of efficiently computable functions does not have an online efficient encoding with online rate smaller than* 1.

Proof. Assume, towards a contradiction, that there exists a compiler C that for every polynomial-time computable function f outputs an encoding $\hat{f}$ whose online communication is smaller than n and its online computational complexity is $n^{c'}$ for some universal constant c'. Then, any efficiently computable function f can be compressed by the $n^{c'}$-time computable function $\hat{f}_{\mathsf{on}}(x;r)$ where r is some fixed string, e.g., the all-zero string. Indeed, it is possible to (efficiently) recover the value

of $f(x)$ by computing $z = \hat{f}_{\mathsf{of}}(r)$ and applying the decoder to $(z, \hat{f}_{\mathsf{on}}(x;r))$. This contradicts the incompressibility assumption. □

In the case of functions with many outputs (and assuming the existence of one-way functions) we can lower-bound the *offline* complexity by the output length ℓ where ℓ can be polynomially larger than n. To this end, we will prove that the output length of the simulator (or even the online-complexity of the simulator) is lower-bounded by ℓ.

Lemma 6.4 (**Communication complexity of RE is larger than the output length**). *Assuming one-way function, for every constant c there exists a function $f : \{0,1\}^n \to \{0,1\}^{n^c}$ such that every $(n^{\omega(1)}, 1/3)$-private RE of f has communication complexity of at least n^c bits. Furthermore, there are at least n^c bits in the output of the simulator $\mathsf{Sim}(y)$ that depend on the input y (as opposed to the randomness).*

Note that the existence of one-way functions is necessary, as otherwise one can obtain an encoding with total complexity n, by letting $\hat{f}(x;r) = x'$ where x' is random sibling of x under f, and take the decoder to be $\mathsf{Dec}(x') = f(x')$, and the simulator $\mathsf{Sim}(y) = x'$ where x' is a random preimage of y. If one-way functions do not exist then this encoding can be implemented efficiently.

Proof. Fix some constant c, and let $f : \{0,1\}^n \to \{0,1\}^\ell$ be a pseudorandom generator with output length $\ell = n^c$. (The existence of such a pseudorandom generator follows from the existence of one-way functions [32].) It suffices to prove the "furthermore" part as the online-complexity of the simulator lower-bounds the communication complexity of the encoding. Let $\hat{f}(x;r)$ be an RE of f with decoder Dec and simulator Sim such that the number of bits of $\mathsf{Sim}(y)$ that depend on y is smaller than ℓ. Then, we distinguish the output of f from a truly random string via the following test: Given a string $y \in \{0,1\}^\ell$, we accept if and only if the outcome of $\mathsf{Dec}(\mathsf{Sim}(y))$ is equal to y.

First we claim that when y is random the test accepts with probability at most $\frac{1}{2}$. Indeed, fix some value r for the randomness of the simulator and some value d for the randomness of the decoder. Then the image of $\mathsf{Sim}(y;r) = (z_r, \mathsf{Sim}_{\mathsf{on}}(y;r))$ can take at most $2^{\ell-1}$ values, and therefore the decoder $\mathsf{Dec}(\cdot;s)$ recovers y successfully for at most half of all y's in $\{0,1\}^\ell$.

On the other hand, if y is in the image of f, the test accepts with probability at least $2/3 - \mathrm{neg}(n)$. Indeed, let x be a preimage of y, then by definition $\mathsf{Dec}(\hat{f}(x;r))$ outputs $y = f(x)$ with probability 1. Since $\hat{f}(x;r)$ is $(t, 1/3)$ indistinguishable from $\mathsf{Sim}(f(x))$, it follows that $\mathsf{Dec}(\mathsf{Sim}(y)) = y$ with probability at least $2/3 - \mathrm{neg}(n)$. □

6.2 On Adaptive Security

The standard security definition of REs can be captured by the following game: (1) The challenger secretly tosses a random coin $b \xleftarrow{R} \{0,1\}$; (2) the adversary chooses an input x submits it to the challenger and gets as a result the string $\hat{y}$ which, based on the secret bit b, is either sampled from the encoding $\hat{f}(x;r)$ or from the simulator $\mathsf{Sim}(f(x))$. At the end, the adversary outputs his guess b' for the bit b. The security of REs says that the t-bounded adversaries cannot win the game (guess b) with probability better than $\frac{1}{2} + \varepsilon$. In the online/offline setting it is natural to consider an *adaptive* version of this game in which the adversary chooses its input x based on the offline part of the encoding. Syntactically, this requires an online/offline simulator $\mathsf{Sim}(y;r) = (\mathsf{Sim}_{\mathsf{of}}(r); \mathsf{Sim}_{\mathsf{on}}(x;r))$ whose offline part does not depend on its input $f(x)$, and has the same length as the offline part of the encoding. Formally,

Definition 6.5 (**Adaptively-secure RE**). *Let f be a function and $\hat{f}(x;r) = (\hat{f}_{\mathsf{of}}(r), \hat{f}_{\mathsf{on}}(x;r))$ be a perfectly-correct RE with decoder Dec and online/offline simulator $\mathsf{Sim}(y;r) = (\mathsf{Sim}_{\mathsf{of}}(r), \mathsf{Sim}_{\mathsf{on}}(y;r))$.*

We say that $\hat{f}$ is (t,ε) adaptively private *if every t-bounded adversary $\mathcal{A}$ wins the following game with probability at most $\frac{1}{2}+\varepsilon$: (1) The challenger secretly tosses a random coin $b \xleftarrow{R} \{0,1\}$, chooses randomness r and outputs*

$$\hat{y}_{\mathsf{of}} = \begin{cases} \hat{f}_{\mathsf{of}}(r) & \text{if } b=1, \\ \mathsf{Sim}_{\mathsf{of}}(r) & \text{if } b=0. \end{cases}$$

(2) Based on $\hat{y}_{\mathsf{of}}$ the adversary $\mathcal{A}$ chooses an input x, submits it to the challenger and gets as a result the string

$$\hat{y}_{\mathsf{on}} = \begin{cases} \hat{f}_{\mathsf{on}}(x;r) & \text{if } b=1, \\ \mathsf{Sim}_{\mathsf{on}}(f(x);r) & \text{if } b=0. \end{cases}$$

At the end, the adversary outputs his guess b' and wins if $b'=b$.

As follows from Lemma 6.4, in the standard model adaptively secure REs cannot be online-efficient let alone have constant rate (assuming the existence of one-way functions).

Corollary 6.6 (**Adaptive security requires long online-communication**). *Assuming one-way function, for every constant c there exists a function $f:\{0,1\}^n \to \{0,1\}^{n^c}$ such that every RE of f has online communication complexity of at least n^c bits.*

Proof. By Lemma 6.4, there exists a function f for which the online part of the simulator must longer than n^c. Privacy ensures that the online communication complexity of $\hat{f}$ satisfies the same bound. □

On the other hand, it turns out that this barrier can be bypassed via the use of a (programmable) random oracle as shown in the following lemma.

Lemma 6.7. *Suppose that $f:\{0,1\}^n \to \{0,1\}^\ell$ has a (t,ε)-private RE $\hat{f}$ with online communication αn. Then, for every $\beta>0$, f has a $(t'=\Omega(t), \varepsilon'=\varepsilon+t/2^{\beta n})$ adaptively-private RE g with online communication of $(\alpha+\beta)n$ in the random oracle model. Furthermore, if $\hat{f}$ is affine then so is g.*

Proof. Let $\hat{f}(x;r)=(\hat{f}_{\mathsf{on}}(x;r), \hat{f}_{\mathsf{of}}(r))$ be a (t,ε) (affine) RE of $f(x)$ over $\mathbb{Z}_q$ with decoder Dec, and simulator $\mathsf{Sim}=(\mathsf{Sim}_1,\mathsf{Sim}_2)$, where Sim_1 denotes the first αn bits of Sim and $\mathsf{Sim}_2, \hat{f}_{\mathsf{of}}(r)$ output s-long vectors in $\mathbb{Z}_q$. Let $H:\{0,1\}^{\beta n} \to \mathbb{Z}_q^s$ be a random oracle. Consider the encoding

$$g_{\mathsf{of}}(r,k) := \hat{f}_{\mathsf{of}}(r) + H(k) \pmod q \qquad\qquad g_{\mathsf{on}}(x;r,k) := (\hat{f}_{\mathsf{on}}(x;r), k)$$

where $k \xleftarrow{R} \{0,1\}^{\beta n}$. Given $g(x;(r,k)) = (\hat{f}_{\mathsf{of}}(r)+H(k), \hat{f}_{\mathsf{on}}(x;r), k)$, the decoder Dec' decodes $\hat{f}_{\mathsf{of}}(r)$ (by subtracting $H(k)$ from the first entry) and then applies the original decoder Dec to $\hat{f}(x;r)$. The simulator Sim' works as follows: at the offline phase it outputs $\rho \xleftarrow{R} \mathbb{Z}_q^s$, then at the online phase given y it outputs the pair $(\mathsf{Sim}_1(y;r),k)$, where $k \xleftarrow{R} \{0,1\}^{\beta n}$ and programs the random oracle so that $H(k) = \rho - \mathsf{Sim}_2(y;r) \pmod q$.

We claim that the resulting encoding is (t',ε') adaptively private. Assume, towards a contradiction, that we have a t'-bounded adversary $\mathcal{B}$ that wins the adaptive-RE game with probability $\frac{1}{2}+\varepsilon'$. Then, we can construct a t-bounded adversary that distinguishes $\hat{f}(x;r)$ from $\mathsf{Sim}(f(x))$ with advantage ε as follows: Choose a random string $z_1 \xleftarrow{R} \mathbb{Z}_q^s$ for the offline phase and send it to $\mathcal{B}$, let x be its response. If $\mathcal{B}$ makes a query to the random oracle, we record the query and answer it with a random string. (Without loss of generality, $\mathcal{B}$ never asks the same query twice.) Now, we

try to break the encoding $\hat{f}$ with respect to the input x. As a result, we get $\hat{y} = (\hat{y}_{\mathsf{on}}, \hat{y}_{\mathsf{of}})$ which is either sampled from $\hat{f}(x)$ or from $\mathsf{Sim}(f(x))$. Send $\mathcal{B}$ the string $z_2 = (\hat{y}_{\mathsf{on}}, k)$ where $k \stackrel{R}{\leftarrow} \{0,1\}^{\beta n}$, and set $H(k) = \rho - \hat{y}_{\mathsf{of}}$. If k happens to be a query that $\mathcal{B}$ already asked, we terminate with failure. Otherwise, we continue the emulation while answering queries k' to the random oracle randomly (if $k' \neq k$) or by $H(k)$ if $k' = k$.

To analyze the success probability of $\mathcal{A}$ first observe that the emulation fails with probability at most $t/2^{\beta n}$. Furthermore, assuming non-failure, the emulation is perfect in the sense that if $\hat{y}$ is sampled from the encoding $\hat{f}(x; r)$ (resp., from the simulator $\mathsf{Sim}(f(x))$) then the view of $\mathcal{B}$ is distributed identically to its view in the adaptive RE game when the challenge bit $b = 1$ (resp. $b = 0$). Hence, the lemma follows. □

7 Applications

7.1 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; 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.[6] The DDH-based encoding of f can be written as

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

where the offline part F depends only on R and 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.

In the semi-honest model, the protocol is straightforward. In the offline phase, Alice receives F and r^a, and Bob receives r^b along with the $2n_a + 2n_b$ keys $K^A_{i,\sigma}, K^B_{i,\sigma}$. 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 7.1. *Suppose DDH 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.*

The malicious model. Security in the malicious model is handled via a "homomorphic MAC" over $\mathbb{Z}_p$ which allows Alice to verify that Bob sent her the correct linear combination of his keys, namely the one defined by $a \oplus r^a$ and $b \oplus r^b$. This approach has been used by Bendlin et al. for

[6] The case of general two-party functionalities reduces to this case via a standard reduction, cf. [25].

securely computing arithmetic circuits in the preprocessing model [12].[7] Note that there is no room for cheating in sending $a \oplus r^a, b \oplus r^b$: Any choice of these messages uniquely defines the input, which can be easily extracted by the simulator.

The homomorphic MAC construction proceeds as follows. Suppose for simplicity that in the offline phase Bob is given n secret keys $K_1, \ldots, K_n \in \mathbb{Z}_p$ and in the online phase he is expected to reveal some publicly known linear combination $\sum_{i=1}^n \mu_i K_i$ of these keys. The goal is to provide Alice with a mechanism for checking the correctness of Bob's online message without revealing additional information about the keys. This is done by giving to Bob, in the offline phase, independent random field elements $r_1, \ldots, r_n \in \mathbb{Z}_p$, and giving to Alice a random $\alpha \in \mathbb{Z}_p$ along with the n values $\beta_i = \alpha K_i + r_i$. Note that Alice's offline information gives her no information about Bob's keys. In the online phase, when the coefficients μ_i are revealed (in our case $\mu_i \in \{0, 1\}$), Bob sends the values $(M = \sum_{i=1}^n \mu_i K_i, T = \sum_{i=1}^n \mu_i r_i)$. Alice accepts M only if $\alpha M + T = \sum_{i=1}^n \mu_i \beta_i$. Since Alice can compute T from M and α, she does not learn anything except the output. On the other hand, a forging attack by a malicious Bob can be used to guess α. See [12] for more details.

The multiparty case. In the case of a general number of parties we can proceed similarly, except that in this case the offline phase additively secret-shares each key between the parties over $\mathbb{Z}_p$. (This is needed in order to prevent the adversary from learning both the output of the encoding and the keys.) As before, assume without loss of generality that only one party Alice gets the output. The protocol proceeds in two rounds: in the first round each party broadcasts its masked input, and in the second round each party sums up and sends to Alice his shares of the keys defined by the public messages of the first round. Each of these messages is verified by Alice using the homomorphic MAC, as before.

On adaptive choice of inputs. In the presence of malicious parties, the protocol described above realizes the randomized encoding functionality $\hat{f}$ with *statistical* security in the preprocessing model. However, this functionality should be viewed as a reactive functionality which first delivers an offline part, then receives an online input from each party, and finally delivers the online output to Alice. In order for the final protocol to be secure, the encoding should be simulatable with such an adaptive choice of inputs. While we do not have a proof for the adaptive security of our constructions under standard assumptions, it may still hold heuristically when the output for f is shorter than the input, and can be made provably adaptive in the random oracle model (see Section 6.2 and [10]). As in the case of standard garbled circuits with short keys, obtaining adaptive security in the plain model under standard assumptions remains an interesting open problem. We note that this issue is not relevant in the semi-honest model, or when the online inputs are public and are generated independently of the offline phase (this is meaningful when there are secret offline inputs).

7.2 Non-Interactive Zero-Knowledge Proofs

We move on to the case of non-interactive zero-knowledge proofs (NIZK). Such proof systems are similar to standard zero-knowledge protocols except that interaction is traded for the use of a public random string σ to which both the prover and the verifier have a read-only access. Formally,

Definition 7.2. *A NIZK for an* **NP** *relation $R(x, w)$ is a pair of probabilistic polynomial-time algorithms (P, V) that satisfies the following properties:*

[7] In contrast to our protocols, the online communication complexity of the protocol from [12] depends on the circuit size.

- *(Completeness) for every $(x, w) \in R$, it holds that* $\Pr[V(x, \sigma, P(x, w, \sigma; \rho_P); \rho_V) = 1] > 1 - \mathrm{neg}(|x|)$*, where ρ_P is the private randomness of the prover and ρ_V is the private randomness of the verifier.*

- *(Statistical Soundness) for every $x \notin L_R$ (i.e., x such that $\forall w, (x, w) \notin R$) and every computationally unbounded prover algorithm P^* we have that* $\Pr[V(x, \sigma, P^*(x, \sigma); \rho_V) = 1] < \mathrm{neg}(|x|)$*;*

- *(Zero-knowledge) there exists a probabilistic polynomial-time simulator M such that for every string sequence $\{(x_n, w_n)\}$ where $(x_n, w_n) \in R$ it holds that*

$$\{(x_n, \sigma, P(x_n, w_n, \sigma; \rho_P))\} \stackrel{c}{\equiv} \{M(x_n)\},$$

where in all the above σ is uniformly distributed over $\{0,1\}^{\mathrm{poly}(|x|)}$.

In the online/offline setting we assume that the prover's message is partitioned into two parts: (1) An offline message that depends solely on the language L and the public reference string σ; and (2) An online message that depends on the public input x and the private witness w. (We also assume that the verifier gets x in the online phase.) Accordingly, for a prover $P(x, w, \sigma; \rho_P) = (P_{\mathsf{of}}(\sigma, \rho_P), P_{\mathsf{on}}(x, w, \rho_P))$ we define the online communication (resp. computational) complexity of P to be the output length (resp., the circuit size) of $P_{\mathsf{on}}(x, w, \rho_P)$.

Theorem 7.3. *Assume that the language L has a NIZK and assume that RSA, LWE, or DDH holds. Then, L has a NIZK $(\hat{P}, \hat{V})$ with online communication of $|w| + o(|x| + |w|)$ and online computation of $(|x| + |w|)^{1+\varepsilon}$ where ε is an arbitrary small constant.*

Proof. Let (P, V) be a NIZK for L. In [5, 4] it is shown that if $P(x, w, \sigma, \rho_P)$ is encoded by $\hat{P}(x, w, \sigma, \rho_P; r)$ then $(\hat{P}, \hat{V})$ is a NIZK for L where the new verifier $\hat{V} = V(x, \sigma, \mathsf{Dec}(\hat{y}); \rho_V)$ uses the decoder Dec to translate the prover's encoded message $\hat{y}$ to the corresponding message of the original prover, and then invokes the original verifier.

To prove the theorem we compile the prover $P(x, w, \sigma, \rho_P)$ into its succinct randomized encoding $\hat{P}(x, w, \sigma, \rho_P; r_x, r_w)$ constructed in Section 4 while treating (σ, ρ_P) as offline inputs and (x, w) as online inputs. As a result the online computation is $(|x| + |w|)^{1+\varepsilon}$ and the online communication is $|x| + |w| + o(|x| + |w|)$. To further reduce the online communication observe that the online part has the form $(x \oplus r_x, w \oplus r_w, K(x, w; r))$, since x is public this is information theoretically equivalent to sending the message $(r_x, w \oplus r_w, K(x, w; r))$, however, the randomness r_x can be sent at the offline phase and so the total online communication is $|w| + o(|x| + |w|)$ as needed. □

Remark 7.4 (Adaptivity)**.** *The zero-knowledge property is proven under the assumption that the online input x is chosen independently of the offline part. When x is chosen based on the offline part, we do not know how to efficiently simulate the view of the verifier under standard assumptions. Still, security in this case may hold heuristically, and can be provably achieved in the random oracle model (see Section 6.2). It is important to note that this issue is not relevant to the* soundness *of the protocol (which holds statistically). Furthermore, the problem is completely avoided when the online input is generated independently of the offline phase (e.g., by the prover).*

7.3 Verifiable Computation

In the problem of Verifiable Computation (VC) a computationally weak client C wishes to delegate the computation of a function f on an input x to a computationally strong but untrusted server P. We consider two-message protocols in the offline/online setting. Namely, the client sends an offline

message $\alpha = C_{\mathsf{of}}(\rho_C)$ before seeing the input x, then at the online phase the client sends a single message $\beta = C_{\mathsf{on}}(x, \rho_C)$ to the server (which potentially reveals x). The server responds with an answer $\gamma = P(\alpha, \beta, \rho_P)$. Based on this answer, the client applies some cheap verification process $V(x, \gamma, \rho_V)$ and either recovers the result $f(x)$ or announces an error in the case of a cheating server. Formally,

Definition 7.5. (Verifiable Computation) *A VC protocol for an efficiently computable function $f : \{0,1\}^n \to \{0,1\}^m$ is a pair of probabilistic polynomial-time algorithms, a client $C = (C_{\mathsf{on}}, C_{\mathsf{of}}, V)$ and a server P, that satisfies the following properties:*

- *(Completeness) for every $x \in \{0,1\}^n$, it holds with probability 1 that*

$$V\left(x, P(C_{\mathsf{of}}(\rho_C), C_{\mathsf{on}}(x, \rho_C); \rho_P); \rho_C\right) = f(x),$$

  where ρ_P is the private randomness of the server and ρ_C is the private randomness of the client.

- *(Soundness) for every x and every efficiently computable server P^* we have that*

$$\Pr[V\left(x, P^*(C_{\mathsf{of}}(\rho_C), C_{\mathsf{on}}(x, \rho_C); \rho_P); \rho_C\right) \notin \{f(x), \perp\}] < \mathrm{neg}(|x|).$$

We will be interested in *useful* protocols in which it is more efficient to run the client's algorithm than to compute the function itself.

Theorem 7.6. *Assume that RSA, LWE, or DDH holds and let $\varepsilon > 0$ be an arbitrary small constant. For every efficiently computable function $f : \{0,1\}^n \to \{0,1\}^m$ there exists an online/offline VC $C = (C_{\mathsf{on}}, C_{\mathsf{of}}, V)$ with the following complexity:*

- *Offline communication/computational complexity of the client is $|f| \cdot \kappa$, where $|f|$ denotes the circuit size of f and κ is a security parameter.*
- *Online communication complexity of the client is $n + o(n)$ and its online computational complexity is $n^{1+\varepsilon}$.*
- *The communication complexity of the server is $m + n^{\varepsilon}$ and its computational complexity is $|f| \cdot n^{\varepsilon}$.*
- *The verification step V has computational complexity of $O(m + n^{\varepsilon})$.*

Observe that the online communication of the protocol is essentially optimal (up to additive loss) as even if the server is fully trusted the client has to send at least n-bits to describe x and the server has to send at least m bits to describe $f(x)$.

Proof. Let $\kappa = n^{\varepsilon}$. In [6] it is shown that the following protocol is VC. Let $g(x, k) = \mathtt{MAC}_k(f(x)$ where $\mathtt{MAC}_k : \{0,1\}^m \to \{0,1\}^{\kappa}$ is a one-time (information-theoretic) MAC with an error of $2^{-\kappa}$. Let $\hat{g}(x, k; r)$ be the computationally-private perfectly-correct RE for g where k is treated as an offline input. Let $\hat{g}_{\mathsf{of}}(k; r)$ be the offline message of the client, and let $(x, \hat{g}_{\mathsf{on}}(x; r))$ be the online message of the client. The server P sends the pair $\gamma_1 = f(x)$ and $\gamma_2 = \mathsf{Dec}((\hat{g}_{\mathsf{of}}(k; r), \hat{g}_{\mathsf{on}}(x; r)))$ where Dec is the decoder of the encoding. Finally, the client accepts γ_1 if $\gamma_2 = \mathtt{MAC}_k(\gamma_1)$.

To prove the theorem we employ the encoding constructed in Section 4 and instantiate the MAC with the pair-wise independent hash function from [35] whose circuit complexity is $O(m + \kappa)$. □

We remark that one can add input privacy (i.e., hide x from the server) without increasing the complexity (see [6]). Also, as in [22], the offline phase can be re-used (and therefore amortize) without increasing the (asymptotic) complexity by encrypting $\hat{g}_{\mathsf{of}}(k;r)$ and $\hat{g}_{\mathsf{on}}(x;r)$ under fully-homomorphic encryption and letting the server return an encryption of γ_2. Re-using the offline phase remain secure as long as the server does not learn whether the client accepted or rejected the interaction.

Remark 7.7 (Adaptivity)**.** *We do not prove that the soundness property holds when the input x is chosen adaptively based on the offline part. As usual, this can be solved with the aid of a random oracle (see Section 6.2). It is important to note that in this context non-adaptive solution is still meaningful as in the typical scenario, where the client is the one who selects which input x to delegate, the problem is completely avoided.*

References

[1] B. Applebaum. Key-dependent message security: Generic amplification and completeness theorems. In *EUROCRYPT*, 2011.

[2] B. Applebaum, D. Cash, C. Peikert, and A. Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, pages 595–618, 2009.

[3] B. Applebaum, D. Harnik, and Y. Ishai. Semantic security under related-key attacks and applications. In *ICS*, pages 45–60, 2011.

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

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

[6] B. Applebaum, Y. Ishai, and E. Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *ICALP (1)*, pages 152–163, 2010.

[7] B. Applebaum, Y. Ishai, and E. Kushilevitz. How to garble arithmetic circuits. In *FOCS*, pages 120–129, 2011.

[8] B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. In *EUROCRYPT*, pages 423–444, 2010.

[9] Beaver. Precomputing oblivious transfer. In *CRYPTO*, 1995.

[10] M. Bellare, V. T. Hoang, and P. Rogaway. Adaptively secure garbling with applications to one-time programs and secure outsourcing. *IACR Cryptology ePrint Archive*, 2012:564, 2012. To appear in *Asiacrypt* 2012.

[11] M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. *IACR Cryptology ePrint Archive*, 2012:265, 2012. To appear in *ACM CCS* 2012.

[12] R. Bendlin, I. Damgård, C. Orlandi, and S. Zakarias. Semi-homomorphic encryption and multiparty computation. In *EUROCRYPT*, 2011.

[13] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.

[14] C. Cachin, J. Camenisch, J. Kilian, and J. Müller. One-round secure computation and secure autonomous mobile agents. In *ICALP*, pages 512–523, 2000.

[15] K.-M. Chung, Y. T. Kalai, and S. P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *CRYPTO*, 2010.

[16] R. Cramer, S. Fehr, Y. Ishai, and E. Kushilevitz. Efficient multi-party computation over rings. In *EUROCRYPT*, pages 596–613, 2003.

[17] 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/`.

[18] B. Dubrov and Y. Ishai. On the randomness complexity of efficient sampling. In *STOC*, pages 711–720, 2006.

[19] Even, Goldreich, and Micali. On-line/off-line digital signatures. *J. Cryptology*, 9, 1996.

[20] U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation. In *STOC*, 1994.

[21] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, 2010.

[22] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010. Earlier version: Cryptology ePrint Archive, Report 2009/547.

[23] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[24] H. Gilbert, M. J. B. Robshaw, and Y. Seurin. How to encrypt with the lpn problem. In *ICALP (2)*, pages 679–690, 2008.

[25] O. Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.

[26] O. Goldreich, S. P. Vadhan, and A. Wigderson. On interactive proofs with a laconic prover. *Electronic Colloquium on Computational Complexity (ECCC)*, 8(46), 2001.

[27] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, 2008.

[28] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179, 2012.

[29] V. Goyal, Y. Ishai, A. Sahai, R. Venkatesan, and A. Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, pages 308–326, 2010.

[30] J. Groth. Minimizing non-interactive zero-knowledge proofs using fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2011:12, 2011.

[31] D. Harnik and M. Naor. On the compressibility of *np* instances and cryptographic applications. *SIAM J. Comput.*, 39(5):1667–1713, 2010.

[32] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. Preliminary versions appeared in STOC' 89 and STOC' 90.

[33] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, pages 294–304, 2000.

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

[35] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography with constant computational overhead. In *STOC*, pages 433–442, 2008.

[36] Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591, 2008.

[37] Y. T. Kalai and R. Raz. Probabilistically checkable arguments. In *Proc. of CRYPTO*, 2009.

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

[39] S. Micali. CS proofs (extended abstracts). In *FOCS*, pages 436–453, 1994.

[40] Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *CRYPTO*, 2002.

[41] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC*, pages 333–342, 2009.

[42] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005. Full version in JACM 56(6), article 34, 2009.

[43] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Comm. of the ACM*, 21(2):120–126, 1978.

[44] 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.

[45] T. Sander, A. Young, and M. Yung. Non-interactive cryptocomputing for NC^1. In *FOCS*, pages 554–567, 1999.

[46] Shamir and Tauman. Improved online/offline signature schemes. In *CRYPTO*, 2001.

[47] A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Trans. Comput. Syst*, 1(1):38–44, 1983.

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

A Useful properties of REs

Fact A.1 (Substitution). *Suppose that the function $\hat{f}(x;r)$ is a (t,ε)-encoding of $f(x)$ with the simulator and decoder $(\mathsf{Sim},\mathsf{Dec})$. Let $h(z)$ be a function of the form $f(g(z))$ where $z \in \{0,1\}^k$ and $g : \{0,1\}^k \to \{0,1\}^n$. Then, the function $\hat{h}(z;r) = \hat{f}(g(z);r)$ is a (t,ε)-encoding of h with the same simulator and the same decoder.*

Proof. Follows immediately from the definition. For correctness we have:

$$\Pr_r[\mathsf{Dec}(\hat{h}(z;r)) \neq h(z)] = \Pr_r[\mathsf{Dec}(\hat{f}(g(z);r)) \neq f(g(z))] = 0,$$

and for privacy we have

$$\mathsf{Sim}(h(z)) \equiv \mathsf{Sim}(f(g(z))) \equiv_{t,\varepsilon} \hat{f}(g(z);r) \equiv \hat{h}(z;r),$$

as required. □

Fact A.2 (Concatenation). *Suppose that $\hat{f}_i(x;r_i)$ is a (t,ε)-encoding of the function $f_i : \{0,1\}^n \to \{0,1\}^{\ell_i}$ with simulator Sim_i, decoder Dec_i and complexity at most s, for every $i \in [c]$. Then the function $\hat{f}(x;(r_1,\ldots,r_c)) = (\hat{f}_i(x;r_i))_{i=1}^c$ is a $(t-cs, c\varepsilon)$-encoding of $f(x) = (f_1(x),\ldots,f_c(x))$ with simulator $\mathsf{Sim}(y) = (\mathsf{Sim}_i(y_i))_{i=1}^c$ and decoder $\mathsf{Dec}(\hat{y}) = (\mathsf{Dec}_i(\hat{y}_i))_{i=1}^c$.*

Proof. Perfect correctness follows from $\Pr_r[\mathsf{Dec}(\hat{f}(x;r)) \neq f(x)] \leq \sum \Pr_r[\mathsf{Dec}(\hat{f}_i(x;r_i)) \neq f_i(x)] = 0$. Privacy is proved via a standard hybrid argument. Specifically, suppose, towards a contradiction, that $\mathcal{A}$ is a $(t-cs)$ size adversary that distinguishes $\hat{f}(x;r)$ from $\mathsf{Sim}(f(x);\rho)$ with advantage $c\varepsilon$. Then, by an averaging argument, for some $j \in \{1,\ldots,c\}$ the adversary $\mathcal{A}$ distinguishes with advantage at least ε between the tuple

$$(\hat{f}_1(x;r_1),\ldots,\hat{f}_{j-1}(x;r_{j-1}),\mathsf{Sim}_j(f_j(x)),\ldots,\mathsf{Sim}_c(f_c(x)))$$

and the tuple

$$(\hat{f}_1(x;r_1),\ldots,\hat{f}_j(x;r_j),\mathsf{Sim}_{j+1}(f_j(x)),\ldots,\mathsf{Sim}_c(f_c(x))).$$

Now, we can define an adversary $\mathcal{B}$ that ε-distinguishes $\hat{f}_j(x;r_j)$ from $\mathsf{Sim}_j(f_j(x))$. Given a challenge $\hat{y}_j$, the adversary $\mathcal{B}$ samples $(\hat{f}_i(x;r_i))_{i<j}$ and $(\mathsf{Sim}_i(f_i(x)))_{i>j}$ with complexity $c \cdot s$, and invokes $\mathcal{A}$ on the resulting vector with the challenge planted in the j-th position. This gives rise to a (t,ε)-adversary, contradicting our hypothesis. □

Fact A.3 (Composition). *Suppose that:*

- *$g(x;r_g)$ is a (t_1,ε_1)-encoding of $f(x)$ with decoder Dec_g and simulator Sim_g, and*
- *$h((x,r_g);r_h)$ is a (t_2,ε_2)-encoding of the function $g(x,r_g)$, viewed as a single-argument function, with decoder Dec_h, simulator Sim_h and complexity s.*

Then the function $\hat{f}(x;(r_g,r_h)) = h((x,r_g);r_h)$ is a $(\min(t_1-s,t_2),\varepsilon_1+\varepsilon_2)$-encoding of $f(x)$ where (r_g,r_h) are its random inputs and the simulator and decoder are $\mathsf{Sim}(y) = \mathsf{Sim}_h(\mathsf{Sim}_g(y))$ and $\mathsf{Dec}(\hat{y}) = \mathsf{Dec}_g(\mathsf{Dec}_h(\hat{y}))$.

Proof. To prove perfect correctness note that $\Pr_{r_g,r_h}[\mathsf{Dec}(\hat{f}(x;r_g,r_h)) \neq f(x)]$ is upper-bounded by

$$\Pr_{r_g,r_h}[\mathsf{Dec}(h(x,r_g;r_h)) \neq g(x,r_g)] + \Pr_{r_g}[\mathsf{Dec}(\hat{g}(x;r_g)) \neq f(x)] = 0.$$

We prove privacy by noting that $\mathsf{Sim}_g(f(x))$ is (t_1,ε_1)-indistinguishable from $g(x;r_g)$. Hence, $\mathsf{Sim}_h(\mathsf{Sim}_g(f(x)))$ is (t_1-s,ε_1) indistinguishable from $\mathsf{Sim}_h(g(x;r_g))$. However, the latter distribution is (t_2,ε_2)-indistinguishable from $h((x,r_g);r_h)$, and so $h(x;(r_g,r_h))$ is $(\min(t_1-s,t_2),\varepsilon_1+\varepsilon_2)$-indistinguishable from $\mathsf{Sim}_g(f(x))$. □