Foundations of Homomorphic Secret Sharing

Elette Boyle[1], Niv Gilboa[2], Yuval Ishai[3], Huijia Lin[4], and Stefano Tessaro[4]

1 **IDC Herzliya, Herzliya, Israel**
eboyle@alum.mit.edu
2 **Ben Gurion University, Be'er Sheva, Israel**
gilboan@bgu.ac.il
3 **Technion, Haifa, Israel**
yuvali@cs.technion.ac.il
4 **University of California Santa Barbara, Santa Barbara, CA, USA**
{rachel.lin,tessaro}@cs.ucsb.edu

Abstract

Homomorphic secret sharing (HSS) is the secret sharing analogue of homomorphic encryption. An HSS scheme supports a local evaluation of functions on shares of one or more secret inputs, such that the resulting shares of the output are short. Some applications require the stronger notion of *additive* HSS, where the shares of the output add up to the output over some finite Abelian group. While some strong positive results for HSS are known under specific cryptographic assumptions, many natural questions remain open.

We initiate a systematic study of HSS, making the following contributions.

- **A definitional framework.** We present a general framework for defining HSS schemes that unifies and extends several previous notions from the literature, and cast known results within this framework.
- **Limitations.** We establish limitations on *information-theoretic* multi-input HSS with short output shares via a relation with communication complexity. We also show that *additive* HSS for non-trivial functions, even the AND of two input bits, implies non-interactive key exchange, and is therefore unlikely to be implied by public-key encryption or even oblivious transfer.
- **Applications.** We present two types of applications of HSS. First, we construct 2-round protocols for secure multiparty computation from a simple constant-size instance of HSS. As a corollary, we obtain 2-round protocols with attractive asymptotic efficiency features under the Decision Diffie Hellman (DDH) assumption. Second, we use HSS to obtain nearly optimal worst-case to average-case reductions in P. This in turn has applications to fine-grained average-case hardness and verifiable computation.

1998 ACM Subject Classification Mathematical foundations of cryptography

Keywords and phrases Cryptography, homomorphic secret sharing, secure computation, communication complexity, worst-case to average case reductions

Digital Object Identifier 10.4230/LIPIcs.ITCS.2018.21

1 Introduction

Fully homomorphic encryption (FHE) [53, 35] is a powerful cryptographic primitive that supports general computations on encrypted inputs. Despite intensive study, FHE schemes can only be based on a narrow class of cryptographic assumptions [56, 17, 36], which are all related to lattices, and their concrete efficiency leaves much to be desired.

In this paper we consider the following secret sharing analogue of FHE, referred to as *homomorphic secret sharing* (HSS) [14]. A standard (threshold) secret sharing scheme randomly splits an input x into m shares, $(x^1, \ldots, x^m)$, such that any set of t shares reveals nothing about the input. An HSS scheme supports computations on shared inputs by means of local computations on their shares. More concretely, there is a local evaluation algorithm Eval and decoder algorithm Dec satisfying the following homomorphism requirement. Given a description of a function F, the algorithm $\mathsf{Eval}(F; x^j)$ maps an input share x^j to a corresponding output share y^j, such that $\mathsf{Dec}(y^1, \ldots, y^m) = F(x)$.

An HSS scheme as above can be trivially obtained by letting Eval output (F, x^j) and Dec first reconstruct x from the shares and then compute F. Analogously to the output compactness requirement of FHE, we require that the HSS output shares be *compact* in the sense that their length depends only on the output length of F and the security parameter. In fact, it is often useful to make the more stringent requirement that Dec compute $F(x)$ as the sum $y^1 + \ldots + y^m$ in some finite Abelian group. We refer to such an HSS scheme as an *additive HSS*. We also consider a relaxed notion of *weak compactness* that allows the length of the output shares to grow sublinearly with the input size.

Finally, one can naturally consider a *multi-input* variant of HSS, where inputs $x_1, \ldots, x_n$ are independently shared, Eval locally maps the j-th shares of the n inputs to the j-th output share, and Dec outputs $F(x_1, \ldots, x_n)$. In fact, multi-input HSS is meaningful even when F is a fixed function rather than an input of Eval. For instance, one may consider additive 2-input HSS where F computes the AND of two input bits, or compact 2-input HSS where F takes an inner product of two input vectors.

HSS vs. FHE. HSS can generally be viewed as a relaxation of FHE that offers protection against bounded collusions. However, as observed in [14], in some applications of FHE it is possible to use HSS as an alternative that offers the same level of security. For instance, in the context of secure two-party computation [57, 40], using HSS to share the inputs of the two parties does not compromise security in any way, since the two parties together can anyway learn both inputs.

More importantly for this work, HSS can potentially offer several useful features that are inherently impossible for FHE. One such feature is *information-theoretic security*. Information-theoretic HSS schemes for multiplying two secrets with security threshold $t < m/2$ serve as the basis for information-theoretic protocols for secure multiparty computation [9, 20, 25]. Information-theoretic HSS schemes for certain classes of depth-2 circuits implicitly serve as the basis for the best known constructions of information-theoretic private information retrieval schemes and locally decodable codes [58, 29, 8]. Another potential feature of HSS is *optimal compactness*: if F has a single output bit, then the output shares y^j can be as short as a single bit. Indeed, special types of FHE schemes can be used to obtain additive HSS schemes with $t = m - 1$ that support general homomorphic computations with optimal compactness [27]. This feature is useful for several applications of HSS, including ones we discuss in this work.

Finally, recent works obtain HSS schemes that support rich classes of computations under the Decision Diffie Hellman (DDH) assumption [14, 16] or the security of the Paillier encryption scheme [30], which are not known to imply FHE. These constructions use very different techniques from those underlying known FHE constructions. This suggests a potential for further diversifying the assumptions and structures on which HSS can be based, which may potentially lead to more efficient substitutes for known FHE schemes.

1.1 Our Contribution

The current state of the art in HSS mostly consists of isolated positive results and leaves open some of the most basic questions. In this work we initiate a more systematic study of HSS, making the following contributions. We refer the reader to the relevant sections for a high level overview of the main ideas behind each contribution.

A definitional framework. We start, in Section 2, by presenting a general framework for HSS that unifies and extends several previous notions from the literature. In Section 3 we cast some known primitives and previous results within this framework. This includes a simple extension of a previous Learning With Errors (LWE)-based construction from [27] to the setting of multi-input HSS, whose details appear in full version.

Limitations. In Section 4 we establish two types of limitations on multi-input HSS. First, in Section 4.1, we show that weakly compact *information-theoretic* multi-input HSS schemes for security threshold $t \geq m/2$ shares do not exist for functions that have high (randomized, one-way) two-party communication complexity. This includes simple functions such as inner product or set disjointness. The high level idea is to obtain a low-communication two-party protocol from the HSS scheme by having the two parties use a common source of randomness to locally simulate the HSS input shares of both inputs, without any interaction, and then have one party send its HSS output share to the other. Second, in Section 4.2, we show that *additive* HSS for non-trivial functions, or even for computing the AND of two input bits, implies non-interactive key exchange (NIKE), a cryptographic notion which is not known to be implied by standard public-key primitives such as oblivious transfer. Loosely, two parties can simultaneously exchange HSS shares of input bits whose AND is zero, and output their HSS-evaluated output share as a shared key. This result provides some explanation for the difficulty of constructing strong types of HSS schemes from general assumptions.

Applications. In Section 5 we present two types of applications of HSS. First, in Section 5.1, we construct 2-round protocols for secure Multi-Party Computation (MPC) from a simple constant-size instance of additive HSS with $n = 3$ inputs and $m = 2$ shares, for computing $\mathsf{3Mult\text{-}Plus}((x_1, z_1), (x_2, z_2), (x_3, z_3)) = x_1x_2x_3 + z_1 + z_2 + z_3$. At a very high level, this reduction crucially relies on a randomized encoding of functions by degree-3 polynomials [2], to decompose the computation of an arbitrary function F into the computation of many degree-3 monomials. The computation of each monomial is further decomposed into many invocation of HSS for $\mathsf{3Mult\text{-}Plus}$ among only a constant number of parties. As a corollary, we can transform a previous DDH-based 2-round MPC protocol in [16] (which requires a public-key infrastructure) for only a constant number of parties, into a 2-round protocol for an arbitrary polynomial number of parties.

In the literature, 2-round MPC protocols exist *in the CRS model*, based on LWE (e.g., [3, 52]) and *in the plain model,* from indistinguishability obfuscation or witness encryption with NIZK (e.g., [32, 42]) or bilinear groups [33], or even 2-round semi-honest Oblivious Transfer (OT) protocols [34, 11]. Our protocol can be instantiated in the public-key infrastructure model under DDH, which is weaker than or incomparable to the feasibility results of other recent constructions. However, our protocols using HSS still have several advantages, in particular, they enjoy better asymptotic efficiency, and they are in the more general client-server model, where the input clients' computation can be done offline, and the output clients' computation is relatively cheap.

A second type of applications, presented in Section 5.2, is to obtaining worst-case to average-case reductions in P. Roughly speaking, the HSS evaluation function Eval for

computing F defines a function $\hat{F}$ such that computing F on any given input x can be reduced to computing $\hat{F}$ on two or more inputs that are individually pseudorandom. A similar application of FHE was already pointed out in [24]. However, an advantage of the HSS-based reductions is that they allow $\hat{F}$ to have a single bit of output. Another advantage is the potential of diversifying assumptions. We discuss applications of the reductions implied by HSS to fine-grained average-case hardness and verifiable computation. In particular, the HSS-based approach yields checking procedures for polynomial-time computations that achieve better soundness vs. succinctness tradeoffs than any other approach we are aware of.

2 General Definitional Framework for HSS

In this section we give a general definition of homomorphic secret sharing (HSS) that can be instantiated to capture different notions from the literature. We consider multi-input HSS schemes that support a compact evaluation of a function F on shares of inputs $x_1, \ldots, x_n$ that originate from different clients. More concretely, each client i randomly splits its input x_i between m servers using the algorithm Share, so that x_i is hidden from any t colluding servers. We assume $t = m - 1$ by default. Each server j applies a local evaluation algorithm Eval to its share of the n inputs, and obtains an output share y^j. The output $F(x_1, \ldots, x_n)$ is reconstructed by applying a decoding algorithm Dec to the output shares $(y^1, \ldots, y^m)$.

To make HSS useful, we require that Dec be in some sense "simpler" than computing F. The most natural simplicity requirement, referred to as *compactness*, is that the output length of Eval, and hence the complexity of Dec, depend only on the output length of F and not on the input length of F. A more useful notion of simplicity is the stronger requirement of *additive* decoding, where the decoder computes the exclusive-or of the output shares or, more generally, adds them up in some Abelian group $\mathbb{G}$. We also consider weaker notions of simplicity that are are needed to capture HSS constructions from the literature.

Finally, for some of the main applications of HSS it is useful to let F and Eval take an additional input x_0 that is known to all servers. This is necessary for a meaningful notion of single-input HSS (with $n = 1$). Typically, the input x_0 will be a description of a function f applied to the input of a single client, e.g., a description of a circuit, branching program, or low-degree polynomial. The case of single-input HSS is considerably different from the case of multi-input HSS with no server input. In particular, the negative results presented in this work do not apply to single-input HSS.

We now give our formal definition of general HSS. We refer the reader to Example 5 for an example of using this definition to describe an HSS scheme for multiplying two field elements using Shamir's secret sharing scheme. Here and in the following, we use the notation $\Pr[A_1; \ldots; A_m : E]$ to denote the probability that event E occurs following an experiment defined by executing the sequence $A_1, \ldots, A_m$ in order.

▶ **Definition 1** (HSS)**.** An n-client, m-server, t-secure homomorphic secret sharing scheme for a function $F : (\{0,1\}^*)^{n+1} \to \{0,1\}^*$, or (n, m, t)-*HSS* for short, is a triple of PPT algorithms $(\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ with the following syntax:

- $\mathsf{Share}(1^\lambda, i, x)$: On input 1^λ (security parameter), $i \in [n]$ (client index), and $x \in \{0,1\}^*$ (client input), the sharing algorithm Share outputs m input shares, $(x^1, \ldots, x^m)$. By default, we require Share to run in (probabilistic) polynomial time in its input length; however, we also consider a relaxed notion of efficiency where Share is given the total length ℓ of *all* $n + 1$ inputs (including x_0) and may run in time $\mathsf{poly}(\lambda, \ell)$.
- $\mathsf{Eval}\big(j, x_0, (x_1^j, \ldots, x_n^j)\big)$: On input $j \in [m]$ (server index), $x_0 \in \{0,1\}^*$ (common server input), and $x_1^j, \ldots, x_n^j$ (jth share of each client input), the evaluation algorithm Eval

outputs $y^j \in \{0,1\}^*$, corresponding to server j's share of $F(x_0; x_1, \ldots, x_n)$.

- $\mathsf{Dec}(y^1, \ldots, y^m)$: On input $(y^1, \ldots, y^m)$ (list of output shares), the decoding algorithm Dec computes a final output $y \in \{0,1\}^*$.

The algorithms $(\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ should satisfy the following requirements:

- **Correctness:** For any $n+1$ inputs $x_0, \ldots, x_n \in \{0,1\}^*$,

$$\Pr\left[\begin{array}{l}\forall i \in [n]\ (x_i^1, \ldots, x_i^m) \leftarrow \mathsf{Share}(1^\lambda, i, x_i) \\ \forall j \in [m]\ y^j \leftarrow \mathsf{Eval}\big(j, x_0, (x_1^j, \ldots, x_n^j)\big)\end{array} : \mathsf{Dec}(y^1, \ldots, y^m) = F(x_0; x_1, \ldots, x_n)\right] = 1.$$

  Alternatively, in a *statistically correct* HSS the above probability is at least $1 - \mu(\lambda)$ for some negligible μ and in a δ-correct HSS (or δ-HSS for short) it is at least $1 - \delta - \mu(\lambda)$. In the case of δ-HSS the error parameter δ may be given as an additional input to Eval, and the running time of Eval is allowed to grow polynomially with $1/\delta$.

- **Security:** Consider the following semantic security challenge experiment for corrupted set of servers $T \subset [m]$:
  1: The adversary gives challenge index and inputs $(i, x, x') \leftarrow \mathcal{A}(1^\lambda)$, with $|x| = |x'|$.
  2: The challenger samples $b \leftarrow \{0,1\}$ and $(x^1, \ldots, x^m) \leftarrow \mathsf{Share}(1^\lambda, i, \tilde{x})$, where $\tilde{x} = x$ if $b = 0$ and $\tilde{x} = x'$ if $b = 1$.
  3: The adversary outputs a guess $b' \leftarrow \mathcal{A}((x^j)_{j \in T})$, given the shares for corrupted T.

  Denote by $\mathsf{Adv}(1^\lambda, \mathcal{A}, T) := \Pr[b = b'] - 1/2$ the advantage of $\mathcal{A}$ in guessing b in the above experiment. For circuit size bound $S = S(\lambda)$ and advantage bound $\alpha = \alpha(\lambda)$, we say that an (n, m, t)-HSS scheme $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ is (S, α)-secure if for all $T \subset [m]$ of size $|T| \le t$, and all non-uniform adversaries $\mathcal{A}$ of size $S(\lambda)$, we have $\mathsf{Adv}(1^\lambda, \mathcal{A}, T) \le \alpha(\lambda)$. We say that Π is:
  - *computationally secure* if it is $(S, 1/S)$-secure for all polynomials S;
  - *statistically α-secure* if it is (S, α)-secure for all S;
  - *statistically secure* if it statistically α-secure for some negligible $\alpha(\lambda)$;
  - *perfectly secure* if it is statistically 0-secure.

▶ Remark 2 (Unbounded HSS). Definition 1 treats the number of inputs n as being fixed. We can naturally consider an unbounded multi-input variant of HSS where F is defined over arbitrary sequences of inputs x_i, and the correctness requirement is extended accordingly. We denote this flavor of multi-input HSS by $(*, m, t)$-HSS.

▶ Remark 3 (Robust decoding). Definition 1 allows Dec to use all output shares for decoding the output. When $t < m-1$, one can consider a stronger variant of HSS where Dec can recover the output from any $t+1$ output shares. Such a robust notion of threshold homomorphic encryption was recently considered in [44]. In this work we do not consider robust decoding.

2.1 Notions of Simple Decoding

As discussed above, to make HSS useful we impose two types of simplicity requirements on Dec. The most stringent requirement is that Dec adds its input over some Abelian group $\mathbb{G}$. We refer to such an HSS scheme as being *additive*. Note that any HSS scheme where Dec computes a fixed linear combination of the output shares (over some finite field) can be converted into an additive scheme by letting Eval multiply its outputs by the coefficients of the linear combination. See Example 5 for a relevant concrete example.

A more liberal requirement is *compactness*, which says that the length of the output shares depends only on the output length and the security parameter, independently of the input length. Finally, we also consider a further relaxation that we call *weak compactness*,

requiring that the length of the output shares be sublinear in the input length when the input length is sufficiently bigger than the security parameter and the output length. This weaker notion is needed to capture some HSS constructions from the literature, and is used for making our negative results stronger. We formalize these notions below.

▶ **Definition 4** (Additive and compact HSS)**.** We say that an (n, m, t)-HSS scheme $\Pi = (\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ for F is:

- *Additive* if Dec outputs the exclusive-or of the m output shares, or $\mathbb{G}$-additive if Dec computes addition in an Abelian group $\mathbb{G}$;
- *Compact* if there is a polynomial p such that for every $\lambda, \ell_{\mathsf{out}}$, and inputs $x_0, x_1, \ldots, x_n \in \{0,1\}^*$ such that $|F(x_0; x_1, \ldots, x_n)| = \ell_{\mathsf{out}}$, the length of each output share obtained by applying Share with security parameter λ and then Eval is at most $p(\lambda) \cdot \ell_{\mathsf{out}}$ (or $O(\ell_{\mathsf{out}})$ for perfect or statistically α-secure HSS with a constant α);
- *Weakly compact* if there is a polynomial p and sublinear function $g(\ell) = o(\ell)$, such that for every $\lambda, \ell_{\mathsf{in}}, \ell_{\mathsf{out}}$, and inputs $x_0, x_1, \ldots, x_n \in \{0,1\}^*$ of total length ℓ_{in} with $|F(x_0; x_1, \ldots, x_n)| = \ell_{\mathsf{out}}$, the length of each output share obtained by applying Share with security parameter λ and then Eval is at most $g(\ell_{\mathsf{in}}) + p(\lambda) \cdot \ell_{\mathsf{out}}$ (or $g(\ell_{\mathsf{in}}) + O(\ell_{\mathsf{out}})$ for perfect or statistically α-secure HSS with a constant α). More generally, we can specify the precise level of compactness by referring to an HSS scheme as being $g(\lambda, \ell_{\mathsf{in}}, \ell_{\mathsf{out}})$-compact.

2.2 Default Conventions

It is convenient to make the following default choices of parameters and other conventions.

- We assume $t = m - 1$ by default and write (n, m)-HSS for $(n, m, m-1)$-HSS.
- We assume computational security by default, and refer to the statistical and perfect variants collectively as "information-theoretic HSS."
- In the case of *perfectly secure* or *statistically α-secure* HSS, λ is omitted.
- For $n \geq 2$ clients, we assume by default that the servers have no input and write $F(x_1, \ldots, x_n)$, omitting the server input x_0. Note that (n, m, t)-HSS with server input can be reduced to $(n+1, m, t)$-HSS with no server input by letting the server input be shared by one of the clients.
- We consider *additive HSS* by default. This stronger notion is useful for several application of HSS, and most HSS constructions realize it.
- We will sometimes be interested in additive HSS for a constant-size (finite) function F, such as the AND of two bits; this can be cast into Definition 1 by just considering an extension $\hat{F}$ of F that outputs 0 on all invalid inputs. Note that our two notions of compactness are not meaningful for a constant-size F. We can similarly handle functions F that impose restrictions on the relation between the lengths of different inputs. Since Eval can know all inputs lengths, we can ensure that Dec outputs 0 in case of mismatch.
- As noted above, the common server input x_0 is often interpreted as a "program" P from a class of programs $\mathcal{P}$ (e.g., circuits or branching programs), and F is the universal function defined by $F(P; x_1, \ldots, x_n) = P(x_1, \ldots, x_n)$. We refer to this as *HSS for the class $\mathcal{P}$*.

2.3 HSS with Setup

When considering multi-input HSS schemes, known constructions require different forms of *setup* to coordinate between clients. This setup is generated by a PPT algorithm Setup and is reusable, in the sense that the same setup can be used to share an arbitrary number of inputs. We consider the following types of setup:

- *No setup*: This is the default notion of HSS defined above.

- *Common random string (CRS) setup*: An algorithm $\mathsf{Setup}(1^\lambda)$ is used to generate a uniformly random string σ which is given as input to Share, Eval, and Dec.
- *Public-key setup:* We consider here a strong form of public-key setup in which $\mathsf{Setup}(1^\lambda)$ outputs a public key pk and m secret evaluation keys $(\mathsf{ek}_1, \ldots, \mathsf{ek}_m)$, where each key is given to a different server. The algorithm Share is given pk as an additional input, and $\mathsf{Eval}(j, \ldots)$ is given ek_j as an additional input. The security game is changed by giving both the adversary and the challenger pk and giving to the adversary $(\mathsf{ek}_j)_{j\in T}$ in addition to $(x^j)_{j\in T}$. Following the terminology from [16], we refer to HSS with this type of setup as *public-key* $(*, m, t)$*-HSS*.

3 Constructions

In this section we present positive results on HSS that are either implicit in the literature or can be easily obtained from known results. We cast these results in terms of the general HSS framework from Section 2.

We start with a detailed example for casting Shamir's secret sharing scheme [54] over a finite field $\mathbb{F}$ as a perfectly secure, $\mathbb{F}$-additive $(2, m, t)$-HSS scheme for the function F that multiplies two field elements. Such a scheme exists if and only if $m > 2t$.

▶ **Example 5** (Additive $(2, m, t)$-HSS for field multiplication)**.** *Let m, t be parameters such that $m > 2t$, let $\mathbb{F}$ be a finite field with $|\mathbb{F}| > m$, let $\theta_1, \ldots, \theta_m$ be distinct nonzero field elements, and let $\lambda_1, \ldots, \lambda_n$ be field elements ("Lagrange coefficients") such that for any univariate polynomial p over $\mathbb{F}$ of degree at most $2t$ we have $p(0) = \sum_{j=1}^{m} \lambda_j p(\theta_j)$. Let $F : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ be the (constant-size) function defined by $F(x_1, x_2) = x_1 \cdot x_2$. A perfectly secure, additive $(2, m, t)$-HSS scheme for F is defined by the following algorithms. (Since F is a constant-size function we are not concerned with efficiency; we also omit x_0 since there is no server input and omit the security parameter λ since security is perfect.)*

1. $\mathsf{Share}(i, x)$*: pick $r_1, \ldots, r_t$ uniformly at random from $\mathbb{F}$ and let $p(Z) = x + r_1 Z + r_2 Z^2 + \ldots + r_t Z^t$ be a random polynomial of degree at most t with x as its free coefficient. Output $(p(\theta_1), \ldots, p(\theta_m))$. Note that* Share *does not depend on i (the inputs are shared the same).*
2. $\mathsf{Eval}\big(j, (x_1^j, x_2^j)\big)$*: Output $\lambda_j \cdot x_1^j x_2^j$.*
3. $\mathsf{Dec}(y^1, \ldots, y^m)$*: Output $y^1 + \ldots + y^m$.*

We now survey some other instances of HSS schemes from the literature.

- Additive m-out-of-m secret sharing over an Abelian group $\mathbb{G}$ is a $\mathbb{G}$-additive, perfectly secure $(*, m)$-HSS for the function $F(x_1, \ldots, x_n) = x_1 + \ldots + x_n$ where $x_i \in \mathbb{G}$. This is the first instance of HSS considered in the literature [10].
- Generalizing Example 5, multiplicative secret sharing [25] over a finite field $\mathbb{F}$ is an $\mathbb{F}$-additive, perfectly secure $(2, m, t)$-HSS for the function F that multiplies two field elements. Such schemes exist if and only if $m > 2t$. Multiplicative secret sharing schemes such as Shamir's scheme serve as the basis for secure multiparty computation protocols in the information-theoretic setting [9, 20]. More generally, information-theoretic $\mathbb{F}$-additive (d, m, t)-HSS for multiplying d elements of $\mathbb{F}$ exists if and only if $m > dt$ [7]. Multiplicative schemes with a smaller threshold t that work over a constant-size field (independent of m) can be based on algebraic geometric codes [21]. Efficient multiplicative schemes that support a pointwise multiplication of two vectors are considered in [31, 19].
- A 1-round k-server private information retrieval (PIR) scheme [22, 23] can be seen as a *weakly compact* $(1, k, 1)$-HSS for the selection function $F(D; \gamma) = D_\gamma$. For the 2-server case ($k = 2$), information theoretic PIR schemes provably cannot achieve our stronger

notion of compactness unless the share size is linear in $|D|$ [39, 47]. Moreover, current schemes only realize our relaxed notion of efficiency for Share, since the share size is super-polynomial in $|\gamma|$ (see [28] for the best known construction in terms of total size of input shares and output shares). In the computational case, there are in fact *additive* 2-server schemes based on the existence of one-way functions, where Share satisfies the default strict notion of efficiency (see [15] for the best known construction).

- Non-trivial instances of compact, perfectly-secure (1,3,1)-HSS for certain classes of depth-2 boolean circuits [8] implicitly serve as the basis for the best known constructions of information-theoretic 3-server PIR schemes and 3-query locally decodable codes [58, 29].
- The main result of [14] is a construction of (single-input, computationally secure, additive) $(1, 2)$-δ-HSS for branching programs under the DDH assumption. The same paper also obtains a public-key $(*, 2)$-δ-HSS variant of this result. Similar results assuming the circular security of the Paillier encryption were recently obtained in [30].
- The notion of function secret sharing (FSS) from [13] is dual to the notion of HSS for a program class $\mathcal{P}$. It can be cast as an additive $(1, m)$-HSS for the universal function $F(x; P) = P(x)$, where $P \in \mathcal{P}$ is a program given as input to the client and x is the common server input. The special case of distributed point function (DPF) [37] is FSS for the class of point functions (namely, functions that have nonzero output for at most one input). DPF can be seen as additive $(1, m)$-HSS for the function $F(x; (\alpha, \beta))$ that outputs β if $x = \alpha$ and outputs 0 otherwise. It is known that one-way functions are necessary and sufficient for DPF [37].
- We observe that additive[1] $(*, m)$-HSS for *circuits* with statistical correctness can be obtained from the Learning With Errors (LWE) assumption, by a simple variation of the FSS construction from spooky encryption of [27] (more specifically, their techniques for obtaining 2-round MPC). The share size in this construction must grow with the circuit depth, hence Share only satisfies the relaxed notion of efficiency; this dependence can be eliminated by relying on a stronger variant of LWE that involves circular security. We provide details of the underlying tools and construction in the full version.

We note that a key feature of HSS is that Dec does not require a secret key. This rules out nontrivial instances of single-server HSS. In particular, single-server PIR [49] and fully homomorphic encryption [35] cannot be cast as instances of our general definitional framework of HSS.

4 Limitations

In this section, we discuss some inherent limitations in HSS. First, in Section 4.1, we show lower bounds on the length of output shares in statistically-secure HSS using communication complexity lower bounds. In Section 4.2, we show that additive $(2, 2)$-HSS for the AND of two bits implies non-interactive key-exchange (NIKE). Given what is known about NIKE (in particular it only follows from non-generic assumptions, and it is not known to be implied directly by public-key encryption or OT), this gives a strong justification for the lack of instantiations from generic assumptions.

[1] If one settles for the weaker notion of compactness, then single-input HSS can be trivially obtained from any FHE scheme by letting Share include an encryption of the input in one of the shares and split the decryption key into m shares.

4.1 Lower Bounds for Statistically-Secure Multi-Input HSS

We show lower bounds on the length of output shares in statistically-secure multi-input HSS using lower bounds from communication complexity. The key step is to derive a public-coin two-party protocol to compute a function F from an HSS scheme for the function F, and such that the communication cost of the resulting protocol only depends on the length of the output shares.

Communication complexity refresher. We consider public-coin interactive protocols Π between two parties, Alice and Bob, who start the execution with respective inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, and common random tape R. (We can assume wlog that the protocol is otherwise deterministic, and all random coins come from R.) At any point in the execution, one of the parties can return an output value, denoted $\Pi(R, x, y)$. The cost of Π is the maximum number of bits exchanged by Alice and Bob, taken as the worst case over all possible inputs x, y, and random tapes R. We also say that such a protocol is *one-way* (or *one round*) if only one message is sent, and this goes from Alice to Bob.

We are interested in the inherent cost of a protocol Π that evaluates a function $F : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$. In particular, the *(randomized) communication complexity of F with error ϵ*, denoted $\mathsf{R}_\epsilon(F)$, is the minimum cost of a public-coin protocol Π such that $\Pr[\Pi(R, x, y) \neq F(x, y)] \leq \epsilon$ for all x, y, where the probability is over the public random string R. If we restrict ourselves to one-way protocols, then we define analogously the *one-way communication complexity of F with error ϵ*, denoted $\mathsf{R}_\epsilon^{A\to B}(F)$. It is clear that $R_\epsilon^{A\to B}(F) \geq R_\epsilon(F)$.

The following are classical examples of lower bounds on the (one-way) randomized communication complexity.

▶ **Theorem 6** (e.g., [50]). *Let* $\mathsf{IP}_\ell : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}$ *be such that* $\mathsf{IP}_\ell(x, y) = \sum_{i=1}^{\ell} x_i y_i \pmod 2$. *Then,* $\mathsf{R}_{1/3}(\mathsf{IP}_\ell) = \Omega(\ell)$.

▶ **Theorem 7** ([45]). *Let* $\mathsf{DISJ}_\ell : \{0,1\}^\ell \times \{0,1\}^\ell \to \{0,1\}$ *be such that* $\mathsf{DISJ}_\ell(x, y) = \neg \bigvee_{i=1}^{\ell}(x_i \wedge y_i)$. *Then,* $\mathsf{R}_{1/3}(\mathsf{DISJ}_\ell) = \Omega(\ell)$.

▶ **Theorem 8** ([48]). *Let* $\mathsf{INDEX}_\ell : \{0,1\}^\ell \times [\ell] \to \{0,1\}$ *be such that* $\mathsf{INDEX}_\ell(x_1 x_2 \ldots x_\ell, i) = x_i$. *Then,* $\mathsf{R}_{1/3}^{A\to B}(\mathsf{INDEX}_\ell) = \Omega(\ell)$.

Lower bounds on the length of output shares. We start with a lower bound on the length of the output shares in $(2, 2)$-HSS. (Recall that (n, m)-HSS is a shorthand for $(n, m, t = m - 1)$-HSS.) Below, we extend the technique to more general settings.

Recall that a $(2, 2)$-HSS scheme is defined for a function $F : (\{0,1\}^*)^2 \to \{0,1\}^*$. (In this section, we consider the case where the servers have no input x_0, but the results extend straightforwardly to handle server inputs.) For any two integers $\ell_{1,\mathsf{in}}, \ell_{2,\mathsf{in}}$, it is convenient to define the restriction $F^{\ell_{1,\mathsf{in}},\ell_{2,\mathsf{in}}} : \{0,1\}^{\ell_{1,\mathsf{in}}} \times \{0,1\}^{\ell_{2,\mathsf{in}}} \to \{0,1\}^*$ such that $F^{\ell_{1,\mathsf{in}},\ell_{2,\mathsf{in}}}(x_1, x_2) = F(x_1, x_2)$. Also, for a suitable function g, we say that a $(2, 2)$-HSS scheme is *g-compact* if, for security parameter λ, when the two inputs have lengths $\ell_{1,\mathsf{in}}$ and $\ell_{2,\mathsf{in}}$, respectively, the output shares have length each at most $g(\lambda, \ell_{1,\mathsf{in}}, \ell_{2,\mathsf{in}})$.

▶ Proposition 9 (Compactness lower bound). Let $(\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ be a $(2, 2)$-HSS scheme for a function $F : (\{0,1\}^*)^2 \to \{0,1\}^*$, which is statistically α-secure, g-compact, and δ-correct. Then, for all λ, and $\ell_{1,\mathsf{in}}, \ell_{2,\mathsf{in}} > 0$, $g(\lambda, \ell_{1,\mathsf{in}}, \ell_{2,\mathsf{in}}) \geq \mathsf{R}_{\delta(\lambda)+4\alpha(\lambda)}^{A\to B}(F^{\ell_{1,\mathsf{in}},\ell_{2,\mathsf{in}}})$.

We defer a proof to the full version, and only give a sketch here. It is easy to give a protocol for Alice and Bob to evaluate $F^{\ell_{1,\mathsf{in}},\ell_{2,\mathsf{in}}}$ on their respective inputs x_1, x_2: Bob runs $(x_2^1, x_2^2) \leftarrow \mathsf{Share}(1^\lambda, 2, x_2)$, and sends x_2^1 to Alice. Alice then runs $(x_1^1, x_1^2) \leftarrow \mathsf{Share}(1^\lambda, 1, x_1)$

and $y_1 \leftarrow \mathsf{Eval}(1^\lambda, 1, (x_1^1, x_2^1))$, and sends (x_1^2, y_1) to Bob. Finally, Bob computes $y_2 \leftarrow \mathsf{Eval}(1^\lambda, 2, (x_1^2, x_2^2))$, as well as the output $y \leftarrow \mathsf{Dec}(1^\lambda, y_1, y_2)$. However, we would like to make the protocol complexity independent of the input shares – this can be achieved by exploiting HSS security, as well as reverse sampling. Namely, we generate the shares x_1^2 and x_2^1 by running $\mathsf{Share}(1^\lambda, 1, 0^{\ell_{1,\mathsf{in}}})$ and $\mathsf{Share}(1^\lambda, 2, 0^{\ell_{2,\mathsf{in}}})$, respectively, and make these shares part of the common randomness. Then, when Alice and Bob need the shares x_1^1 and x_2^2, respectively, they each exploit knowlege of their respective inputs x_1 and x_2 to locally sample a share consistent with the input and the other share being equal the pre-sampled share in the common random tape. HSS security implies that the distribution of the resulting shares is close to the correct one, and the new protocol only has Alice send y_1 to Bob.

As an application, consider any statistically secure $(2,2)$-HSS scheme for inner products, i.e., for the function $\mathsf{IP} : (\{0,1\}^*)^2 \to \{0,1\}$ such that $\mathsf{IP}(x_1, x_2) = \mathsf{IP}_\ell(x_1, x_2)$ whenever $|x_1| = |x_2| = \ell$. Then, the following corollary implies that such scheme cannot be weakly compact. Similar lower bounds can be obtained for disjointness, and for the index function.

▶ **Corollary 10.** *There exists no weakly compact, statistically* $1/24$*-secure* $1/6$*-correct* $(2,2)$*-HSS scheme for* IP*.*

Proof. Apply Proposition 9 with $\ell_{1,\mathsf{in}} = \ell_{2,\mathsf{in}} = \ell$, $\delta = 1/6$, and $\alpha = \frac{1}{24}$. Regardless of the security parameter, the length of the output shares must be at least $\mathsf{R}_{1/3}^{A \to B}(\mathsf{IP}_\ell) = \Omega(\ell_{1,\mathsf{in}} + \ell_{2,\mathsf{in}})$ by Theorem 6, and this violates weak compactness. ◀

Extensions. Proposition 9 can be extended to obtain lower bounds for general (n, m, t)-HSS where $m, n \geq 2$ and $t \geq m/2$. We briefly summarize the main ideas here.

- $(n, 2)$*-HSS.* For any n-ary function $F : (\{0,1\}^*)^n \to \{0,1\}^*$, we can define a two-party function as follows. Fix $k \in \{1, \ldots, n-1\}$, as well as Alice's indices $I_1 = (a_1, \ldots, a_k)$, and Bob's indices $I_2 = (b_1, \ldots, b_{n-k})$, where $\{a_1, \ldots, a_k, b_1, \ldots, b_{n-k}\} = [n]$. Then, $F'((x_{a_1}, \ldots, x_{a_k}), (x_{b_1}, \ldots, x_{b_{n-k}})) = F(x_1, \ldots, x_n)$. The proof of Proposition 9 can be adapted to lower bound the length of the output shares in an $(n, 2)$-HSS scheme for F via $\mathsf{R}^{A \to B}(F')$, noting one would then choose the sets I_1, I_2 to maximize communication complexity of the resulting F'.
- (n, m, t)*-HSS for* $t \geq m/2$*.* A lower bound for $(n, 2)$-HSS extends straightforwardly to a lower bound for (n, m, t)-HSS where $t \geq m/2$, since the latter type of HSS implies the former type, by simply having one of the two servers in the $(n, 2)$-HSS simulate $m/2 \leq m_1 \leq t$ servers from the (n, m, t)-HSS scheme, and the other simulate the remaining $m_2 = m - m_1$ servers.
- *Simultaneous messages.* In the case of (n, m)-HSS, where $n \geq m$, we can alternatively obtain useful lower bounds via communication complexity in the *simultaneous message model* [48, 5], where m players send a message to a referee that decides the output. Roughly, a variant of the proof of Proposition 9 would build a protocol where the messages sent are exactly the m servers' output shares.

4.2 Additive Multi-Input HSS Implies Non-Interactive Key Exchange

It is known that roughly any non-trivial additive HSS (even for a single input) implies the existence of one-way functions [37, 13]; in turn, one-way functions have been shown to imply additive $(1, 2)$- and $(1, m)$-HSS for certain classes of simple functions [23, 37, 13, 15]. However, to date, all constructions of additive HSS supporting *multiple* inputs rely on a select list of heavily structured assumptions: DDH, LWE, Paillier, and obfuscation [14, 27, 30]. A clear

challenge is whether one can instantiate such an object from weaker general assumptions, such as one-way functions, public-key encryption, or oblivious transfer.

We show that this is unlikely to occur. We demonstrate the power of additive multi-input HSS by proving that even the minimal version of $(2,2)$-additive-HSS for the AND of two input bits already implies the existence of *non-interactive key exchange (NIKE)* [26], a well-studied cryptographic notion whose known constructions similarly are limited to select structured assumptions. NIKE is black-box separated from one-way functions and highly unlikely to be implied by generic public-key encryption or oblivious transfer.

On the other hand, we observe that $(2,2)$-additive-HSS for AND is unlikely to be implied *by* NIKE, as the primitive additionally implies the existence of 2-message oblivious transfer (OT) [14], unknown to follow from NIKE alone.

We first recall the definition – for a two-party protocol Π between Alice and Bob, we denote by $\mathsf{out}_A(\Pi)$ and $\mathsf{out}_B(\Pi)$ their respective outputs, and $\mathsf{Transc}(\Pi)$ the resulting transcript.

▶ **Definition 11** (NIKE)**.** A 2-party protocol Π with single-bit output is a secure *non-interactive key-exchange (NIKE)* protocol if the following conditions hold:

- **Non-Interactive:** The protocol Π consists of exchanging a single (simultaneous) message.
- **Correctness:** The parties agree on a consistent output bit: $\Pr[\mathsf{out}_A(\Pi) = \mathsf{out}_B(\Pi)] = 1$, over randomness of Π.
- **Security:** There exists a negligible function ν such that for any non-uniform polynomial-time E, for every $\lambda \in \mathbb{N}$, it holds $\Pr[b \leftarrow E(1^\lambda, \mathsf{Transc}(\Pi)) : b = \mathsf{out}_A(\Pi)] \leq 1/2 + \nu(\lambda)$, where probability is taken over the randomness of Π and E.

▶ Proposition 12. The existence of additive $(2,2)$-HSS for the AND function $F : \{0,1\}^2 \to \{0,1\}$ defined by $F(x_1, x_2) = x_1 x_2$ implies the existence of non-interactive key exchange.

Proof. Consider the candidate NIKE protocol given in Figure 1.

Communication Round:

- Alice samples shares of 0: i.e., $(x^A, x^B) \leftarrow \mathsf{Share}(1^\lambda, A, 0)$.
  Send x^B to Bob.
- Bob samples a random bit $b \leftarrow \{0,1\}$ and shares b: $(y^A, y^B) \leftarrow \mathsf{Share}(1^\lambda, B, b)$.
  Send y^A to Alice.

Output round:

- Alice outputs $z^A = \mathsf{Eval}(A, (x^A, y^A)) \in \{0,1\}$.
- Bob outputs $z^B = \mathsf{Eval}(B, (x^B, y^B)) \in \{0,1\}$.

Figure 1 NIKE protocol from any additive $(2,2)$-HSS for AND.

Non-interactive: By construction, the protocol consists of a single communication round.

Correctness: Follows by the additive decoding correctness of the $(2,2)$-HSS for AND. Namely, with probability 1, it holds $z^A + z^B = 0 \in \{0,1\}$; that is, $z^A = z^B$.

Security: Suppose there exists a polynomial-time eavesdropper E who, given the transcript of the protocol x^B, y^A succeeds in predicting Bob's output bit $z^B = \mathsf{Eval}(B(x^B, y^B))$ with advantage α: i.e.,

$$Pr\left[\begin{array}{c} b \leftarrow 0,1; \\ (x^A, x^B) \leftarrow \mathsf{Share}(1^\lambda, A, 0); \\ (y^A, y^B) \leftarrow \mathsf{Share}(1^\lambda, B, b); \\ b' \leftarrow E(1^\lambda, (x^B, y^A)) \end{array} : b' = \mathsf{Eval}(B, (x^B, y^B))\right] \geq 1/2 + \alpha(\lambda).$$

We prove in such case α must be negligible, via the following two claims.

▶ Claim 13. E must succeed with advantage α if Alice shares 1 instead of 0: Explicitly, there exists a negligible function ν_1 for which

$$Pr\left[\begin{array}{c} b \leftarrow 0,1; \\ (x^A, x^B) \leftarrow \mathsf{Share}(1^\lambda, A, 1); \\ (y^A, y^B) \leftarrow \mathsf{Share}(1^\lambda, B, b); \\ b' \leftarrow E(1^\lambda, (x^B, y^A)) \end{array} : b' = \mathsf{Eval}(B, (x^B, y^B))\right] \geq 1/2 + \alpha(\lambda) - \nu_1(\lambda).$$

Proof of Claim 13. Follows by the security of Alice's HSS execution. Namely, consider a distinguishing adversary D for the $(2,2)$-AND-HSS, who performs the following:

1: Sample a random bit $b \leftarrow \{0,1\}$, and HSS share b as $(y^A, y^B) \leftarrow \mathsf{Share}(1^\lambda, B, b)$.
2: Receive a challenge secret share x^B, generated either as $(x^A, x^B) \leftarrow \mathsf{Share}(1^\lambda, A, 0)$ or $(x^A, x^B) \leftarrow \mathsf{Share}(1^\lambda, A, 1)$.
3: Execute E on "transcript" x^B and y^A: Let $b' \leftarrow E(1^\lambda, (x^B, y^A))$.
4: Output 0 if and only if $b' = b$.

By construction, the distinguishing advantage of D is exactly the difference in the prediction advantage of E from the real protocol and the protocol in which Alice shares 1 instead of 0. Thus, this difference must be bounded by some negligible function ν_1. ◀

▶ Claim 14. The prediction advantage $\alpha(\lambda)$ of E must be negligible in λ.

Proof of Claim 14. Follows by the security of Bob's HSS execution. Namely, consider a distinguishing adversary D for the $(2,2)$-AND-HSS, who performs the following:

1: Generate HSS shares of 1, as $(x^A, x^B) \leftarrow \mathsf{Share}(1^\lambda, A, 1)$.
2: Receive challenge secret share y^A, generated as $(y^A, y^B) \leftarrow \mathsf{Share}(1^\lambda, B, b)$ for random challenge bit $b \leftarrow \{0,1\}$.
3: Execute E on "transcript" x^B and y^A: Let $b' \leftarrow E(1^\lambda, (x^B, y^A))$.
4: Output b' as a guess for b.

By construction, the distinguishing advantage of D is precisely $\alpha(\lambda) - \nu_1(\lambda)$. Thus (since ν_1 is negligible), it must be that α is negligible, as desired. ◀

This concludes the proof of Proposition 12. ◀

As a direct corollary of this result, any form of HSS which implies additive $(2,2)$-HSS for AND automatically implies NIKE as well. This includes HSS for any functionality F with an embedded AND in its truth table.

As an example, consider a form of *split* distributed point function [37], where the nonzero input value $\alpha \in \{0,1\}^\ell$ of the secret point function f_α is held split as additive shares across two clients. This corresponds to additive $(2,2)$-HSS for the function $F(x; \alpha_1, \alpha_2) = [x == (\alpha_1 \oplus \alpha_2)]$ (i.e., evaluates to 1 if and only if $x = \alpha_1 \oplus \alpha_2$). Such a notion would have applications for secure computation protocols involving large public databases, where the index α of the desired data item is not known to either party, but rather determined as the result of an intermediate computation. Unfortunately, we show that such a tool (even for inputs of length 2 bits) implies NIKE, and thus is unlikely to exist from lightweight primitives.

▶ **Corollary 15.** *The existence of "split" DPF, i.e. additive $(2,2)$-HSS for the function $F(x; \alpha_1, \alpha_2) = [x == (\alpha_1 \oplus \alpha_2)]$, implies the existence of NIKE.*

Proof. Consider the special case of 2-bit values $\alpha_0, \alpha_1 \in \{0,1\}^2$. We show evaluation of F enables evaluation of AND of clients' input bits, and thus additive $(2,2)$-HSS for AND. Indeed, for any $b_1, b_2 \in \{0,1\}$, observe that $F((0,0);(1,b_1),(b_2,1)) = [(0,0) == ((1,b_1) \oplus (b_2,1))] = [(0,0) == (b_1 \oplus 1, b_2 \oplus 1)] = b_1 \wedge b_2$. ◀

5 Applications

In this section we present two types of applications of HSS. In Section 5.1 we present an application to 2-round secure multiparty computation, and in Section 5.2 we present an application to worst-case to average-case reductions.

5.1 From $(3,2)$-HSS to 2-Round MPC

Let us define the following function over $\mathbb{Z}_2$: $\mathsf{3Mult}(x_1,\ x_2,\ x_3) = x_1x_2x_3$. In this section, we show that $(3,2)$-HSS for $\mathsf{3Mult}$ implies 2-round MPC for arbitrary functions in the client-server model. Recall that (n,m)-HSS refers to HSS with n clients, m servers, tolerating up to $m-1$ corrupted servers. An n-client m-server MPC protocol for computing an n-ary functionality F, is a standard MPC protocol with $n+m+1$ parties, including n (input) clients each holding an input x_i, m servers, and a single output client who receives the output $F(x_1,\cdots,x_n)$. A 2-round n-client m-server MPC protocol has the special communication pattern that in the first round each client sends a message (a.k.a. input share) to each server, and in the second round each server sends a message (a.k.a. output share) to the output client, who then recovers the output. Such a protocol is t-secure if secure against any passive, semi-honest, adversary corrupting any set of parties including at most t servers, according to the standard definition of semi-honest security of MPC. Below we consider the default case of $t = m-1$, and denote such MPC as (n,m)-MPC. Due to the lack of space, we refer the reader to [18, 38] for standard definitions of MPC protocols, and to the full version for more details on client-server MPC.

▶ **Theorem 16.** *Assume the existence of PRGs in* NC^1*. For any* n, m*, and any polynomial-time computable function* $F : (\{0,1\}^*)^n \to \{0,1\}$*, there is a construction of an* (n,m)*-MPC protocol that securely computes* F*, from an additive* $(3,2)$*-HSS for* $\mathsf{3Mult}$*.*

Combining this with the additive δ-HSS construction of [14] from DDH would result in (n,m)-MPC from DDH with (at best) only $1/\mathsf{poly}(\lambda)$ correctness. Fortunately, we can do better. Indeed, as an intermediate step in the proof of Theorem 16 (Lemmas 21 and 22 below), we prove that $(3,3)$-MPC for $\mathsf{3Mult}$ also suffices to imply (n,m)-MPC for general functions. A construction of $(3,3)$-MPC for general functions (in the PKI model) was shown to follow from DDH in [16] (in fact, they obtain (n,c)-MPC for any constant number of servers c). Combining this with Lemmas 21 and 22, and the fact that PRGs in NC^1 also follow from DDH, we obtain the following result. This improves directly over the 2-round MPC result of [16], by supporting an arbitrary polynomial number of servers instead of constant. (See Introduction for comparison with other recent 2-round MPC results.)

▶ **Corollary 17** (2-round MPC from DDH)**.** *For any* n, m*, and any polynomial-time computable function* $F : (\{0,1\}^*)^n \to \{0,1\}$*, there is a construction of an* (n,m)*-MPC protocol that securely computes* F *in the PKI model, assuming DDH.*

We prove Theorem 16 by combining the following steps; see full version for details.

Step 1: $(3,2)$-HSS for 3Mult-Plus. Starting from an additive $(3,2)$-HSS scheme Π_{3Mult} for the function 3Mult, thanks to the property of additive reconstruction, we can directly modify it to obtain an additive $(3,2)$-HSS for the function 3Mult-Plus (again over $\mathbb{Z}_2$) defined as

$$\mathsf{3Mult\text{-}Plus}((x_1,z_1),\ (x_2,z_2),\ (x_3,z_3)) \quad = \quad x_1x_2x_3 + z_1 + z_2 + z_3\ .$$

▶ **Lemma 18.** *There is a construction of additive $(3,2)$-HSS for the function* 3Mult-Plus *from any additive $(3,2)$-HSS for the function* 3Mult*.*

Step 2: $(3,3)$-MPC for 3Mult-Plus. From an additive $(3,2)$-HSS scheme for 3Mult-Plus, we can use the *server-emulation technique* from [16] to construct a 3-client *3-server* MPC protocol for 3Mult-Plus. In fact, the technique in [16] is way more general, it shows that from any given n-client m-server HSS for 3Mult-Plus, one can construct a n-client m^2-server MPC protocol for any n-ary function F, assuming the existence of low-depth PRGs.

▶ **Lemma 19** (Server-Emulation in [16])**.** *Assume existence of PRGs in NC^1. For any n, m and polynomial-time function $F : (\{0,1\}^*)^n \to \{0,1\}$, there is a construction of an (n, m^2)-MPC protocol Π that securely computes F, from an additive (n,m)-HSS for* 3Mult-Plus*.*

Their general lemma implies the following corollary we need, using the fact that one can reduce the number of servers by having a single server simulating multiple ones.

▶ **Corollary 20.** *Assume the existence of PRGs in NC^1. There is a construction of a $(3,3)$-MPC protocol that securely computes* 3Mult-Plus*, from an additive $(3,2)$-HSS for* 3Mult-Plus*.*

Step 3: $(3,m)$-MPC for 3Mult-Plus — Increase the number of servers. Next, from a $(3,3)$-MPC protocol for computing 3Mult-Plus, we show how to construct $(3,m)$-MPC protocol for computing the same function 3Mult-Plus, with an arbitrary number m of servers.

▶ **Lemma 21.** *For any m, there is a construction of $(3,m)$-MPC protocol that securely computes* 3Mult-Plus*, from a $(3,3)$-MPC protocol that securely computes* 3Mult-Plus*.*

Proof Overview. Let Π^3 be a $(3,3)$-MPC protocol for 3Mult-Plus; consider m servers, and three clients C_1, C_2, and C_3. Recall that each client C_d has input (x_d, z_d). If we naively let the three clients execute Π^3 with some subset of 3 servers, in the case all three servers are corrupted, the security of Π^3 no longer holds, and the inputs of all clients are potentially revealed. Thus, the challenge is ensuring that when all but one server is corrupted, the inputs of honest clients remain hidden. To achieve this, each client secret-shares its input bit $x_d = \sum_j s_j^d$; as long as server S_j is uncorrupted, the j'th share s_j^d for each honest client's input x_d remains hidden, and hence so are the inputs x^d. (As we will see shortly, the additive part of the inputs z_d can be hidden easily.) Towards this, note that multiplying x_1, x_2, x_3 boils down to computing the sum of all possible degree 3 monomials over the shares $x_1x_2x_3 = \sum_{ijk} s_i^1 s_j^2 s_k^3$. Our idea is using the protocol Π^3 to compute the each monomial $s_i^1 s_j^2 s_k^3$ hidden with some random blinding bits, and in parallel, use a protocol Π_{Add} for addition to cancel out these random blinding bits, as well as add z_1, z_2, z_3. More specifically,

- for every i, j, k, C_1, C_2, C_3 together with three appropriate servers described below run Π^3 to enable the output client to obtain $M_{ijk} = s_i^1 s_j^2 s_k^3 + t_{ijk}^1 + t_{ijk}^2 + t_{ijk}^3$, where t_{ijk}^d is a random blinding bit sampled by client C_d;
- in parallel, C_1, C_2, C_3 together with all m servers run a $(3,m)$-MPC protocol Π_{Add} to enable the output client to obtain the sum $T = T^1 + T^2 + T^3$, where $T^d = z_d - \sum_{i,j,k} t_{ijk}^d$;
- finally, the output client adds all M_{ijk} with T, which gives the correct output, i.e., $x_1x_2x_3 + z_1 + z_2 + z_3$.

The only question left is what are the three servers involved for computing M_{ijk}; they naturally should be servers S_i, S_j, S_k, since for an honest client, say C_1, if server S_i is uncorrupted, the share s_i^1 remains hidden in all computations of M_{ijk} involving this share. This allows us to argue security. One technicality is that some monomials may have form $s_i^1 s_i^2 s_j^3$ or $s_i^1 s_i^2 s_i^3$ and only correspond to two servers S_i, S_j or one S_i. In the former case, we will use the $(3,2)$-MPC protocol Π^2, and in the latter case, we directly implement a trivial protocol with one server. ◀

Step 4: (n,m)-MPC for F — Increase the number of clients and handle general function. Finally, we show how to construct MPC protocols for computing any n-ary function F, from MPC protocols for computing 3Mult-Plus, using the same number m of servers.

▶ **Lemma 22.** *Assume the existence of PRGs in* NC^1*. For any n, m, and any polynomial-time computable function $F : (\{0,1\}^*)^n \to \{0,1\}$, there is a construction of (n,m)-MPC protocol that securely computes F, from a $(3,m)$-MPC protocol that securely computes* 3Mult-Plus*.*

Proof Overview. Staring from a $(3,m)$-MPC protocol $\Pi_{\mathsf{3Mult\text{-}Plus}}$ for 3Mult-Plus, our goal is constructing a (n,m)-MPC protocol Π_F for an arbitrary F with an arbitrary number of clients. To do so, we reduce the task of computing F to the task of computing a *degree-3* randomized encoding $\mathsf{RE}_F(x_1, \cdots, x_n;\ r)$ of F. Here, having a degree of 3 means that RE_F can be represented as a degree 3 polynomial in its input *and* random bits. Such a randomized encoding scheme is constructed in [43, 1], assuming the existence of a low-depth PRG. The first question is where does the random tape r come from. Clearly, r can not be determined by any subset of clients. Therefore, the natural choice is having $r = r_1 + \cdots + r_n$ contributed by all clients. When the randomized encoding has degree 3, its computation can be expanded into a sum of degree three monomials, that is, $\mathsf{RE}_F(x_1, \cdots, x_n\ ;\ r = r_1 + \cdots + r_n) = \sum a_{ijk}^{\ell}\ v_i v_j v_k$, where each variable v_i is either a bit in some input x_l or a bit in some random tape r_l. This decomposes the computation of F into many 3-way multiplications, which can be done securely using 3Mult-Plus. More specifically, in the protocol Π_F,

- for every monomial $a_{ijk}^{\ell} v_i v_j v_k$, the three clients $C_{l_i}, C_{l_j}, C_{l_k}$ holding the variables v_i, v_j, v_k run $\Pi_{\mathsf{3Mult\text{-}Plus}}$ with all m servers to enable the output client to obtain $M_{ijk} = a_{ijk}^{\ell} v_i v_j v_k + t_{ijk}^{\ell,1} + t_{ijk}^{\ell,2} + t_{ijk}^{\ell,3}$, where the three t variables are random blinding bits sampled by the three clients respectively;
- in parallel, all clients and servers run a (n,m)-MPC protocol Π_{Add} for addition to enable the output client to obtain the sum of all t blinding elements;
- the output client adds all M_{ijk} terms, subtracts the sum of blinding elements to obtain the randomized encoding of F, and decodes the randomized encoding.

◀

5.2 Worst-Case to Average-Case Reductions

In this section we describe a simple application of HSS to worst-case to average-case reductions. We then discuss applications of these reductions to fine-grained average-case hardness and verifiable computation. These applications of HSS can be seen as more efficient or more general conditional variants of previous applications of locally random reductions that rely on arithmetization or error-correcting codes [51, 12, 4, 55, 37, 6]. In contrast to the above reductions, the HSS-based reductions can reduce any polynomial-time computable function to another polynomial-time computable function with closely related complexity.

Worst-case to average-case reductions based on fully homomorphic encryption (FHE) were previously used by Chung et al. [24] in the context of delegating computations. Compared

to the FHE-based reductions, the use of HSS has the advantages of diversifying assumptions, making only a constant number of queries to a *Boolean* function (as small as 2), and minimizing the complexity of recovering the output from the answers to the queries.

To make the discussion concrete, we focus here on the application of (computationally secure, additive[2] for the universal function $F(C;x) = C(x)$. Such HSS schemes can be based on variants of the LWE assumption (as described in the full version). Weaker versions of the following results that apply to branching programs can be based on the DDH assumption or the circular security of Paillier encryption using the HSS schemes from [14, 30].

A high level overview. The idea of using HSS for worst-case to average-case reductions is similar to previous applications of locally random reductions for this purpose, except that we apply a "hybrid HSS" technique [14] to improve the efficiency of the reduction. Concretely, the reduction proceeds as follows. Suppose for simplicity that the HSS sharing algorithm $\mathsf{Share}(1^\lambda, x)$ outputs a pair of shares (x^1, x^2) such that each share is individually pseudo-random. Moreover, suppose that the evaluation function $\mathsf{Eval}(j, C, x^j)$ does not depend on j. The evaluation of a circuit $C : \{0,1\}^n \to \{0,1\}$ on an arbitrary input $x \in \{0,1\}^n$ can then be reduced to the evaluation of an extended circuit $\hat{C}$, defined by $\hat{C}(\hat{x}) = \mathsf{Eval}(C, \hat{x})$, on the two inputs x^1, x^2. Indeed, $C(x) = \hat{C}(x^1) \oplus \hat{C}(x^2)$. Now, suppose that $\hat{C}^*$ is a polynomial-size circuit that agrees with $\hat{C}$ on all but an ϵ fraction of the inputs. Then, by the pseudo-randomness of x^1, x^2, the probability that $\hat{C}^*$ agrees with $\hat{C}$ on both inputs, and hence the reduction outputs the correct value $C(x)$, is at least $1 - 2\epsilon - \mathsf{negl}(n)$. Finally, to make the reduction run in near-linear time, we convert the given HSS into a hybrid HSS scheme in which the sharing Share' can be implemented in near-linear time. The algorithm Share' uses Share to share a short seed r for a pseudorandom generator G, and includes the masked input $G(r) \oplus x$ as part of both shares. Given a circuit C and $G(r) \oplus x$, one can efficiently compute a circuit C' such that $C'(r) = C(x)$. The algorithm Eval' of the hybrid scheme applies Eval to homomorphically evaluate C' on r.

The following theorem formalizes and generalizes the above. Here, by a "near-linear time" algorithm we refer to an algorithm whose running time is $O(n^{1+\epsilon})$ for an arbitrary $\epsilon > 0$. The proof of the theorem is deferred to the full version.

▶ **Theorem 23** (Worst-case to average-case reductions from HSS)**.** *Suppose there is a* $(1,2)$*-HSS scheme* $(\mathsf{Share}, \mathsf{Eval}, \mathsf{Dec})$ *for circuits. Then, there is a near-linear time probabilistic oracle algorithm* $Q^{[\cdot]} : \{0,1\}^* \to \{0,1\}^*$*, polynomial-time algorithm* $A : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$*, and a PPT sampling algorithm* $D(1^n)$ *with the following properties:*

- *Q makes two queries to a Boolean oracle (where the queries are computed in near-linear time and the answers are 1-bit long) and outputs the exclusive-or of the two answer bits.*
- *For any $x \in \{0,1\}^n$ and circuit $C : \{0,1\}^n \to \{0,1\}$, we have $\Pr[Q^{A(C,\cdot)}(x) = C(x)] = 1$.*
- *For any polynomial $p(\cdot)$ there is a negligible $\mu(\cdot)$ such that the following holds. For any $x \in \{0,1\}^n$ and circuits $C : \{0,1\}^n \to \{0,1\}$, $A^*_C : \{0,1\}^n \to \{0,1\}$ of size $\leq p(n)$ such that $\Pr_{\hat{x} \leftarrow D(1^n)}[A^*_C(\hat{x}) = A(C, \hat{x})] \geq 1 - \epsilon$, we have $\Pr[Q^{A^*_C(\cdot)}(x) = C(x)] \geq 1 - 2\epsilon - \mu(n)$.*

Moreover, if Share *produces pseudorandom shares then the distribution $D(1^n)$ can be replaced by the uniform distribution.*

▶ Remark 24 (Instantiating Theorem 23). The strong flavor of HSS required by Theorem 23 can be instantiated under a variant of the LWE assumption that further assumes circular

[2] The requirement of being additive can be relaxed here to small decoding complexity.

security [35, 27]. Due to the negligible decoding error of the HSS, we get a slightly weaker version of the conclusion where $\Pr[Q^{A(C,\cdot)}(x) = C(x)] \geq 1 - \mathsf{negl}(n)$. On the other hand, since the implementation of Eval has a small asymptotic overhead, we get the stronger guarantee that the oracle $A(C, \cdot)$ has roughly the same circuit size as C (rather than being polynomially bigger). One can relax the assumption to a more standard variant of LWE by using *depth-dependent* HSS for circuits, where the length of the input shares grows polynomially with the depth of the circuit C being evaluated by Eval. In this case, using an LWE-based NC^1 implementation of the PRG G, the conclusion of Theorem 23 still holds when restricted to NC-circuits C. More generally, the complexity of Q should in this case be allowed to grow with the depth of C.

We now informally discuss two types of applications of Theorem 23, which follow previous applications of such worst-case to average-case reductions from the literature.

Fine-grained average-case hardness. Theorem 23 implies, assuming HSS for circuits, that the following holds for any constants $c' > c$. For every polynomial-time computable function f there is a polynomial-time computable "extension" $\hat{f}$, such that if $\hat{f}$ has a time-$O(n^c)$ algorithm that computes it correctly on, say, 90% on the inputs, then f has a time-$O(n^{c'})$ probabilistic algorithm that computes it correctly (with overwhelming probability) on every input. This implies that if f is hard in the worst case for time $O(n^{c'})$ then $\hat{f}$ is hard in the average case for time $O(n^c)$. The same connection holds also in a non-uniform setting.

A similar result under the incomparable assumption that FHE exists is given in [24]. These results are incomparable to recent results on fine-grained average case hardness [6, 41] that obtain tighter and unconditional connections of this kind, but only for specific functions f.

Verifiable computation. The goal of program checking [12] is to reliably compute a given function f using an untrusted program or piece of hardware that purportedly computes f. We consider a variant of the problem in which a program M for computing $f : \{0,1\}^n \to \{0,1\}$ can access a purported implementation of a related function $\hat{f}$. The program M can make oracle calls to $\hat{f}$ and perform additional computations, as long as the complexity of these additional computations is significantly smaller than that of computing f from scratch. The requirements are that if $\hat{f}$ is implemented correctly, then $M^{\hat{f}}(x) = f(x)$ for all x. On the other hand, even if $\hat{f}$ is replaced by an incorrect implementation $\hat{f}^*$, the output of $M^{\hat{f}^*}(x)$ on every input x is either $f(x)$ or $\perp$ except with small failure probability ϵ. This is very similar to the traditional goal of verifiable computation, except that a malicious "prover" $\hat{f}^*$ is required to be stateless. In this setting, one can make a direct use of probabilistically checkable proofs (PCPs) for proving the correctness of $f(x)$ without any additional cryptographic machinery.

Using the HSS-based worst-case to average-case reduction from Theorem 23, we get checkers M with the following feature: after an input-independent polynomial-time preprocessing, any computation $f(x)$ can be verified with an arbitrarily small inverse polynomial error by receiving just a constant number of bits from $\hat{f}^*$. (See full version for details.) We do not know of any other approach for verifiable computation that yields such a result.

6 Conclusions and Open Problems

In this work we initiate a systematic study of homomorphic secret sharing (HSS) by providing a taxonomy of HSS variants and establishing some negative results and relations with other primitives. We also present applications of HSS in cryptography and complexity theory.

There is much left to understand about the feasibility and efficiency of HSS in different settings. In the information-theoretic setting, we have no strong negative results for *single-*

input, (weakly) *compact* HSS. This should be contrasted with *multi-input* compact HSS, for which negative results are obtained in this work, and with single-input *additive* HSS, where information-theoretic impossibility results are also known [22]. The difficulty of making progress on this question can be partially explained by its relation with information-theoretic private information retrieval and locally decodable codes [46, 8], for which proving good lower bounds is still an outstanding challenge. However, this barrier only seems to apply to special instances of the general problem. In the computational setting, the main open problems are to obtain HSS schemes for circuits under new assumptions and, more broadly, extend the capabilities of HSS schemes that do not rely on FHE.

Acknowledgements. We thank the anonymous ITCS reviewers for helpful comments.

E. Boyle was supported by ISF grant 1861/16, AFOSR Award FA9550-17-1-0069, and ERC grants 307952, 742754. N. Gilboa was supported by ISF grant 1638/15, a grant by the BGU Cyber Center by the European Union's Horizon 2020 ICT program (Mikelangelo project), and ERC grant 742754. Y. Ishai was supported by ERC grant 742754, NSF-BSF grant 2015782, BSF grant 2012366, ISF grant 1709/14, DARPA/ARL SAFEWARE award, NSF Frontier Award 1413955, NSF grants 1619348, 1228984, 1136174, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the DARPA through the ARL under Contract W911NF-15-C-0205. H. Lin was supported by NSF grants CNS-1528178, CNS-1514526, CNS-1652849 (CAREER), a Hellman Fellowship, the Defense Advanced Research Projects Agency (DARPA) and Army Research Office (ARO) under Contract No. W911NF-15-C-0236, and a subcontract No. 2017-002 through Galois. S. Tessaro was supported by NSF grants CNS-1553758 (CAREER), CNS-1423566, CNS-1719146, CNS-1528178, and IIS-1528041, and by an Alfred P. Sloan Research Fellowship. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

References

1 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *FOCS 2004*, pages 166–175, 2004.

2 Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. In *CCC*, pages 260–274, 2005.

3 Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In *EUROCRYPT*, pages 483–501, 2012.

4 László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

5 László Babai, Anna Gál, Peter G. Kimmel, and Satyanarayana V. Lokam. Communication complexity of simultaneous messages. *SIAM J. Comput.*, 33(1):137–166, 2003.

6 Marshall Ball, Alon Rosen, Manuel Sabin, and Prashant Nalini Vasudevan. Average-case fine-grained hardness. In *STOC 2017*, pages 483–496, 2017.

7 Omer Barkol, Yuval Ishai, and Enav Weinreb. On d-multiplicative secret sharing. *J. Cryptology*, 23(4):580–593, 2010.

8 Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Ilan Orlov. Share conversion and private information retrieval. In *CCC 2012*, pages 258–268, 2012.

9 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, pages 1–10, 1988.

10 Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of A secret sharing. In *CRYPTO 1986*, pages 251–260, 1986.

11 Fabrice Benhamouda and Huijia Lin. k-round MPC from k-round OT via garbled interactive circuits. Manuscript, 2017.

12 Manuel Blum and Sampath Kannan. Designing programs that check their work. In *STOC 1989*, pages 86–97, 1989.

13 E. Boyle, N. Gilboa, and Y. Ishai. Function secret sharing. In *EUROCRYPT*, pages 337–367, 2015.

14 Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In *CRYPTO*, pages 509–539, 2016. Full version: IACR Cryptology ePrint Archive 2016: 585 (2016).

15 Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *ACM CCS 2016*, pages 1292–1303, 2016.

16 Elette Boyle, Niv Gilboa, and Yuval Ishai. Group-based secure computation: Optimizing rounds, communication, and computation. In *EUROCRYPT*, pages 163–193, 2017.

17 Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM J. Comput.*, 43(2):831–871, 2014.

18 Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, pages 143–202, 2000.

19 Ignacio Cascudo Pueyo, Hao Chen, Ronald Cramer, and Chaoping Xing. Asymptotically good ideal linear secret sharing with strong multiplication over *Any* fixed finite field. In *CRYPTO*, pages 466–486, 2009.

20 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *STOC*, pages 11–19, 1988.

21 Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *CRYPTO*, pages 521–536, 2006.

22 B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

23 Benny Chor and Niv Gilboa. Computationally private information retrieval (extended abstract). In *STOC 1997*, pages 304–313, 1997.

24 Kai-Min Chung, Yael Tauman Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *CRYPTO 2010*, pages 483–501, 2010.

25 Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT*, pages 316–334, 2000.

26 Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

27 Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. In *CRYPTO*, pages 93–122, 2016.

28 Z. Dvir and S. Gopi. 2-server PIR with sub-polynomial communication. In *STOC*, pages 577–584, 2015.

29 Klim Efremenko. 3-query locally decodable codes of subexponential length. In *STOC*, pages 39–44, 2009.

30 Nelly Fazio, Rosario Gennaro, Tahereh Jafarikhah, and William E. Skeith III. Homomorphic secret sharing from paillier encryption. In *ProvSec*, pages 381–399, 2017.

31 Matthew K. Franklin and Moti Yung. Communication complexity of secure computation (extended abstract). In *STOC*, pages 699–710, 1992.

32 Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In *TCC*, pages 74–94, 2014.

33 Sanjam Garg and Akshayaram Srinivasan. Garbled protocols and two round MPC from bilinear maps. In *FOCS 2017*, 2017.

34 Sanjam Garg and Akshayaram Srinivasan. Two-round secure multiparty computation from minimal assumptions. Manuscript, 2017.

35 Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

36 Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO (1)*, pages 75–92, 2013.

37 N. Gilboa and Y. Ishai. Distributed point functions and their applications. In *Proc. EUROCRYPT '14*, pages 640–658, 2014.

38 Oded Goldreich. *Foundations of Cryptography — Basic Applications*. Cambridge University Press, 2004.

39 Oded Goldreich, Howard J. Karloff, Leonard J. Schulman, and Luca Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. *Computational Complexity*, 15(3):263–296, 2006.

40 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

41 Oded Goldreich and Guy N. Rothblum. Worst-case to average-case reductions for subclasses of p. *Electronic Colloquium on Computational Complexity (ECCC)*, 17-130, 2017.

42 S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In *CRYPTO, Part II*, pages 63–82, 2015.

43 Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *ICALP*, pages 244–256, 2002.

44 Aayush Jain, Peter M. R. Rasmussen, and Amit Sahai. Threshold fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2017:257, 2017.

45 Bala Kalyanasundaram and Georg Schintger. The probabilistic communication complexity of set intersection. *SIAM J. Discret. Math.*, 5(4), November 1992.

46 Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *STOC*, pages 80–86, 2000.

47 Iordanis Kerenidis and Ronald de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *J. Comput. Syst. Sci.*, 69(3):395–420, 2004.

48 Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.

49 E. Kushilevitz and R. Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *Proc. FOCS '97*, pages 364–373, 1997.

50 Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.

51 Richard J. Lipton. New directions in testing. In *DIMACS Workshop on Distributed Computing And Cryptography*, pages 191–202, 1989.

52 Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In *EUROCRYPT*, pages 735–763, 2016.

53 Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of secure computation*, pages 169–179. 1978.

54 Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

55 Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

56 Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proc. EUROCRYPT 2010*, pages 24–43, 2010.

57 Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

58 S. Yekhanin. Towards 3-query locally decodable codes of subexponential length. In *Proc. STOC*, pages 266–274, 2007.