Bounded Simultaneous Messages

Andrej Bogdanov
School of EECS, University of Ottawa, Canada

Krishnamoorthy Dinesh
Dept. of Computer Science and Engineering, Indian Institute of Technology, Palakkad, India

Yuval Filmus
The Henry and Marylin Taub Faculty of Computer Science, Technion, Haifa, Israel

Yuval Ishai
The Henry and Marylin Taub Faculty of Computer Science, Technion, Haifa, Israel

Avi Kaplan
The Henry and Marylin Taub Faculty of Computer Science, Technion, Haifa, Israel

Sruthi Sekar
University of California, Berkeley, CA, USA

Abstract

We consider the following question of *bounded simultaneous messages* (BSM) protocols: Can computationally unbounded Alice and Bob evaluate a function $f(x, y)$ of their inputs by sending polynomial-size messages to a computationally bounded Carol? The special case where f is the mod-2 inner-product function and Carol is bounded to AC^0 has been studied in previous works. The general question can be broadly motivated by applications in which distributed computation is more costly than local computation.

In this work, we initiate a more systematic study of the BSM model, with different functions f and computational bounds on Carol. In particular, we give evidence against the existence of BSM protocols with polynomial-size Carol for naturally distributed variants of NP-complete languages.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography; Theory of computation → Probabilistic computation

Keywords and phrases Simultaneous Messages, Instance Hiding, Algebraic degree, Preprocessing, Lower Bounds

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2023.23

Related Version *Full Version*: `https://arxiv.org/abs/2310.00334` [8]

Funding *Andrej Bogdanov*: Supported by an NSERC Discovery Grant.
Yuval Filmus: Supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 802020-ERC-HARMONIC.
Yuval Ishai: Supported by ERC Project NTSC (742754), ISF grant 2774/20, and BSF grant 2018393.
Avi Kaplan: Supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 802020-ERC-HARMONIC, and ERC Project NTSC (742754).
Sruthi Sekar: Supported by DARPA under Agreement No. HR00112020026, AFOSR Award FA9550-19-1-0200, NSF CNS Award 1936826, and research grants by the Sloan Foundation, and Visa Inc.

Acknowledgements The authors would like to thanks the anonymous reviewers for their comments.

1 Introduction

The simultaneous messages model is a model for evaluating a function $f(x, y)$ of two inputs: Alice, who holds x and Bob, who holds y, simultaneously send messages to a referee Carol, who outputs the value $f(x, y)$. This model was introduced by Yao [42] and further studied by Babai et al. [4] in the context of communication complexity, where the interest is to determine the minimum length of Alice's and Bob's messages required to specify f.[1]

We are primarily interested in Carol's *computational* complexity, assuming Alice and Bob are unbounded. Specifically, our objective is to understand which functions require large *bounded* simultaneous messages (BSM) protocols. As the BSM complexity of any function can only be smaller than its standalone computational complexity, explicit constructions of provably hard functions for this model are subject to the usual barriers [35, 1]. Nevertheless, owing to its connections to other complexity notions [33, 24, 28],[2] explicit almost-cubic lower bounds have been obtained for depth-3 AND-OR formulas [27], and almost-quadratic lower bounds for arbitrary AND-OR formulas [40], both witnessed by the "inner product modulo two" (IP) function. It remains a challenge to prove BSM lower bounds for other models in which circuit lower bounds are known such as depth-3 AND-OR circuits [32] and parities of DNF [12].

Motivated by cryptographic applications, Rothblum [36] conjectures that IP has exponential-size bounded-depth unbounded-fan-in BSM complexity. In particular, IP does not admit a BSM protocol with Carol in AC^0. Filmus et al. [16] confirmed the conjecture assuming Alice's message is not much longer than her input.

BSM and secure two-party computation

A primary motivation for revisiting the BSM model stems from its relevance to cryptographically secure two-party computation. In this model, the BSM model is the "ideal model" in which evaluation is carried out by a trusted intermediary Carol. A crowning achievement of modern cryptography is the development of secure two-party protocols that dispense off the need for a trusted Carol [43, 19, 7, 10]. However, the complexity of the ideal model is of interest for the following reasons.

First, despite tremendous progress in the design of two-party protocols for secure function evaluation, the overhead remains substantial. Moreover, certain features of trusted evaluation, such as fairness, are inherently lost by any two-party implementation. In practice, a trusted Carol is often preferred over a two-party protocol, and Carol's computational cost needs to be compensated. BSM complexity captures Carol's minimal computational cost incurred under the best-possible preprocessing of Alice's and Bob's inputs.

Second, in the design of two-party protocols, it can be desirable to preprocess the inputs so as to minimize the complexity of the interactive phase. The complexity measure to be optimized depends on the type of protocol. For instance, in protocols based on oblivious transfer [19, 25, 23], the communication complexity and round complexity of the protocol are proportional to the circuit size and circuit depth of f, respectively. In protocols that rely on fully homomorphic encryption [18, 9] and avoid an expensive "bootstrapping" technique, the computational cost is governed by the degree of f as a polynomial over some underlying finite field. The degree also determines the fraction of corrupted servers that can be tolerated in non-interactive secure computation protocols that employ a set of untrusted servers. In this context, BSM complexity captures the best-possible savings that can be obtained by endowing Alice and Bob with unbounded computational power in the preprocessing phase. Conversely, a lower bound in the BSM model indicates limits of preprocessing in a two-party protocol design.

[1] Babai et al. and others [4, 30, 34] more generally study a multi-party variant of this model.

[2] In this literature f is the adjacency function of a bipartite graph and BSM complexity is called *bipartite graph complexity*.

With these motivations in mind, we study the BSM complexity of natural classes of functions that may arise in two-party computation, including Boolean circuits, arithmetic circuits, low-degree polynomials, and time-unbounded halting Turing machines. As some of these classes do not admit explicit lower bounds, any evidence of their hardness must be conditional. One of our contributions is the development of hardness reductions and completeness results for BSM complexity.

Generic constructions and non-explicit bounds

Our primary focus is on Carol's computational complexity, with Alice's and Bob's message size (i.e., communication complexity) as a secondary parameter. Clearly every function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ has a regular circuit size of at most 2^{2n} and therefore BSM circuit size at most 2^{2n} with message size n. We are interested in the regime where Alice's and Bob's messages have a length of at least n.

It is also known that every function has a BSM circuit (parity of ANDs) of size $O(2^n)$ with message size 2^n. This is close to optimal as most functions require BSM circuits of size $\Omega(2^n)$. In contrast, most functions' circuit size is $\Omega(2^{2n}/n)$. Although BSM complexity can be much smaller than circuit complexity, every BSM of circuit size s and total message length ℓ can be simulated by a circuit of size $\ell \cdot 2^n + s$.

1.1 Our Results

Conditional impossibility of efficient BSM for NP

As our first result, we study a distributed variant of SAT, which we refer to as *split SAT*:

$$\text{SSAT} := \{(\alpha, \beta) : \alpha, \beta \text{ are CNFs and } \alpha \wedge \beta \text{ is satisfiable}\}.$$

In Section 2.1 we show that if SSAT has BSM circuit complexity $\mathsf{poly}(|\alpha| + |\beta|)$ then every language in NP can be decided by circuit families of size $O(\mathsf{poly}(n) \cdot 2^{n/2})$. In contrapositive form, no efficient BSM for SSAT exists unless NP has circuits of size $O(\mathsf{poly}(n) \cdot 2^{n/2})$.

We conjecture that the following two NP-languages meet this stringent lower bound:

1. Turing Machine Acceptance: The language ATM_c consists of all pairs (M, x) where M is a nondeterministic Turing Machine that accepts x in $|x|^c$ steps. Here c is a fixed constant; the conjecture is plausible for any $c > 1$. This language is complete in the sense that if $\text{NTIME}(n^c)$ requires circuit size $s(n)$ then ATM_c requires circuit size $s(n - O(1))$.
2. Succinct Subset Sum: The input is an element x of the finite field $\mathbb{F}_{3^m}$. The yes-instances are those x for which there exist coefficients $a_0, a_1, \cdots, a_n \in \{0,1\} \subseteq \mathbb{F}_{3^m}$ such that $a_0 + a_1 x + \cdots + a_n x^n = 0$ and not all a_i are 0, where $n = \lfloor m \log_2 3 \rfloor$.

   While the brute-force 2^n time complexity of worst-case subset-sum can be improved to $2^{n/2}$ in the RAM-model [22, 37, 29], it is unclear if these algorithms yield any savings in circuit size. In contrast, one advantage of the circuit model is that it can decide any language L in size $O(\min(|\{0,1\}^n \cap L|, |\{0,1\}^n \setminus L|))$ by memorizing all yes or no instances of a given length (whichever is smaller). We conjecture that the fraction of yes and no Succinct Subset Sum instances is at least $2^{-n/100}$ thereby rendering this attack ineffective.

As a consequence, in Corollary 5 we show BSM hardness for NP follows from the existence of very strong worst-case one-way functions. The required inversion complexity is as large as $\mathsf{poly}(n) \cdot 2^{n/2}$ under a plausible hardness assumption and $\mathsf{poly}(n) \cdot 2^{2n/3}$ unconditionally (for length-preserving functions). The best-known algorithms for generic function inversion have conjectured worst-case complexity $\mathsf{poly}(n) \cdot 2^{2n/3}$ in the RAM model with preprocessing [21]. The best unconditional upper bound is $\mathsf{poly}(n) \cdot 2^{3n/4}$ [15]. We do not know any non-trivial circuits for generic inversion.

BSM and Instance Hiding

We consider the notion of instance hiding introduced by Beaver and Feigenbaum [5]. An instance hiding (IH) scheme consists of a primary actor, called Henry, with query access to independent oracles. To compute a function on a given input, Henry may query the oracles, but in such a way that each oracle learns nothing about the input, meaning that the query distribution each oracle sees depends only on the input length. Beaver and Feigenbaum allow interactive oracle queries to multiple oracles; we specialize to non-interactive queries and two oracles.

Fortnow and Szegedy [17] asked whether languages in NP can have an instance hiding scheme where Henry is a polynomial-sized circuit. They showed that SAT cannot have $\mathsf{poly}(n)$-sized IH scheme where the two oracles return 1-bit answers under a standard complexity-theoretic assumption. However, a similar statement with oracles returning 2 or more bits is unknown. The hardness assumption about IH which we make here is that languages in NP do not have a $\mathsf{poly}(n)$-sized instance hiding scheme.

In Proposition 10 we show that if BSM for SSAT has circuit complexity $\mathsf{poly}(|\alpha| + |\beta|)$ then every language in NP has a polynomial-size instance hiding scheme. We interpret this as additional evidence against the existence of efficient BSM for NP. The same holds for the split variants of 3-coloring (Proposition 11) and partition (Proposition 12):

$$\text{SCOL} := \{(A,B) : A, B \text{ are graphs on the same vertex set, } A \cup B \text{ is 3-colorable}\},$$
$$\text{SPRT} := \left\{(A,B) : \begin{array}{c} A, B \text{ are sets of non-negative integers,} \\ A \cup B \text{ can be partitioned into two sets of equal sum} \end{array}\right\}.$$

We achieve this via a notion of a reduction that we call *split-hide* reduction (Definition 7). For a language A, we define a language B which is a "split variant" of A, such that if A split-hide reduces to B and B has a BSM protocol with polynomial sized Carol, then A has IH schemes where Henry is polynomial sized (Lemma 8). In addition, if A is NP-complete, then every language in NP has polynomial sized IH schemes (Corollary 9)

Having established this relation between instance hiding and BSM (in Section 2.3), we also obtain a connection between instance hiding and Private Information Retrieval [11] and use this connection (in the form of reduction between the models) to obtain universal lower bounds on IH schemes (Appendix A.1 [8]). This, in turn, gives a two-way reduction chain between BSM and locally decodable codes (Appendix A.2 [8]). Both of these connections could be of independent interest.

BSM and algebraic polynomials

We now move on to the setting where Carol is an algebraic polynomial over some ring. As explained before in the introduction, this setting is motivated by the design of secure two-party protocols that rely on fully homomorphic encryption [18, 9] and avoids expensive bootstrapping (where the computational cost is dependent on the degree of Carol). Every function $f\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ can be represented by a polynomial of degree at most $2n$, and this is also tight. We ask whether this bound can be reduced if we allow preprocessing. To this end, we define $d_f(n,m)$ to be the minimal degree of a polynomial over $\mathbb{F}_2$ such that there exists a BSM protocol for f with preprocessing length of m bits and Carol computing the polynomial, and we define

$$d(n,m) := \max\left\{d_f(n,m) : f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}\right\}.$$

We prove a tight bound of $\Theta(n/\log n)$ for the case $m = \mathsf{poly}(n)$. The upper bound was obtained by Beaver et al. [6]. We show a lower bound for the equality function (Proposition 15).

Going ahead, we ask if there are any savings (in terms of degree) in computing the equality function if we allow Carol to be a polynomial over $\mathbb{Z}_k$ where k is composite, followed by a Boolean decision predicate $P : \mathbb{Z}_k \to \{0,1\}$. We show that this is indeed the case for $k = 6$ (Proposition B.27 [8]) using known constructions of matching vectors [20]. In particular, we give a constant degree Carol, but with superpolynomial ($\exp(\widetilde{O}(\sqrt{n}))$) message length over $\mathbb{Z}_6$ which should be contrasted with the BSM protocol over $\mathbb{F}_2$ (Proposition 14) of polynomial message length but with near linear degree.

BSM, Matrix multiplication and RE languages

We now turn to the setting where Carol is an arithmetic circuit as well as a Turing machine. We start with the setting of an arithmetic circuit. The problem of interest is the matrix multiplication over reals. A remarkable result of Strassen [39] showed a non-trivial algorithm for matrix multiplication of two $n \times n$ matrices over reals in $O(n^{\log_2 7})$ time. Since then, there has been a flurry of improvements (notably [13, 14, 2]) where the best value of the exponent in the runtime is denoted by ω. These results also imply an arithmetic circuit performing $O(n^\omega)$ additions and multiplications over reals computing the matrix product. A natural question is: can there be a BSM protocol where Carol is an arithmetic circuit of size $o(n^\omega)$ such that the preprocessing can have a non-trivial saving in size of Carol? We answer this question in negative in Theorem 13.

Finally, we turn to the setting where Carol is a Turing machine. In Proposition 17, we show that any recursively enumerable $f \in \mathsf{RE}$ (viewed as a language) has a BSM protocol where Carol computes a decidable language (in R).

Summary of results

The results are summarized in Table 1 according to Carol's computational power.

- $\mathsf{BSM}(m,s)$ is the class of all functions $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ that have a two-party SM protocol of size s (namely, Carol is implemented by a circuit of size s) and preprocessing of length m (namely, Alice and Bob send messages of length m; we place a "$\cdot$" symbol in cases where the result is independent of the corresponding parameter).
- $\mathsf{IH}(a,s)$ is the class of all functions $f : \{0,1\}^n \to \{0,1\}$ that have a two-oracle Instance Hiding scheme in which Henry's circuit size is s and each oracle answer is of length a.

We use RE and R to denote the recursively enumerable and recursive languages respectively.

1.2 BSM protocols implicit in literature

In Appendix B.1 [8] we discuss BSM protocols for a subclass of Boolean functions that can be obtained readily from known results in the literature. In particular, we consider the bitwise-combined functions which are functions of the form $f(x,y) = g(x * y)$ where g is a Boolean function on n bits and $* : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a *combiner function* that takes in two n bit strings and output an n bit string. Babai *et.al* [4] considers SM protocols, where $*$ is the bitwise XOR combiner function, with a preprocessing length of $O(2^{0.92n})$. Ambainis and Lokam [3] improved this to obtain SM protocols for bitwise XOR combined

Table 1 Summary of Results.

Bound	Result	Reference
Size	SSAT $\in \mathsf{BSM}(\cdot, \mathsf{poly}(n)) \implies \mathsf{NP} \subseteq \mathsf{SIZE}(\mathsf{poly}(n) \cdot 2^{n/2})$	Thm. 1
	$\implies$ No $\widetilde{O}(2^{2n/3})$-secure length preserving OWF	Cor. 5
	SSAT/SCOL/SPRT $\in \mathsf{BSM}(\cdot, \mathsf{poly}(n)) \implies \mathsf{NP} \subseteq \mathsf{IH}(\cdot, \mathsf{poly}(n))$	Thm. 6
	$g(x \oplus y) \in \mathsf{BSM}(\cdot, s) \implies g \in \mathsf{SIZE}((s/n) \cdot 2^{n/2})$	Prop. B.2 [8]
	$g(x \vee y), g(x \wedge y) \in \mathsf{BSM}(2^{0.729n}, \cdot)$ for all functions $g\colon \{0,1\}^n \to \{0,1\}$	Thm. B.25 [8]
Arithmetic	Matrix multiplication with preprocessing requires $\Omega(n^\omega)$ operations	Thm. 13
Degree	$d(n, m) = O(n/\log(m/n))$ for $n < m$	Prop. 14
	$d_{\mathrm{EQ}}(n, m) = \Omega(n/\log m)$ for $n \leq m$	Prop. 15
	$d(n, n^{C+1}) = \Theta(n/\log n)$ ($C > 0$ constant)	Cor. 16
	$d_{\mathrm{EQ}}(n, m) = 2$ for $m = 2^{\widetilde{\Theta}(\sqrt{n})}$ (over $\mathbb{Z}_6$)	Prop. B.27 [8]
Recursive	$\mathsf{RE} \subseteq \mathsf{BSM}(2n + 1, \mathsf{R})$	Prop. 17

functions where the preprocessing length is bounded by $O(2^{0.729n})$. A standard counting argument shows that most XOR combined functions require a preprocessing length of $\Omega(2^{0.5n})$. (Proposition B.8 [8]). The XOR combined functions are interesting since an efficient BSM protocol computing f would give an efficient IH scheme for g (Proposition B.1 [8]). Using this connection (in Appendix B.1.2 [8]), we could rule out bitwise-XOR combined version of NP-hard languages from having $\mathsf{poly}(n)$ sized BSM protocols with message complexity of $n+1$ unless the polynomial hierarchy collapse and bitwise-XOR combined version PSPACE-hard languages having $\mathsf{poly}(n)$ sized BSM protocols with a message complexity of $n + O(\log n)$ unless $\mathsf{PSPACE} \subseteq \mathsf{PP}^{\mathsf{NP}}/\mathsf{poly}$. These conclusions rely on structural complexity results about instance hiding due to Fortnow and Szegedy [17] and Tripathi [41].

We continue the study where the bitwise combiner function is a bitwise OR as well as bitwise AND function and obtain SM protocols with a preprocessing length of $O(2^{0.729n})$ matching the result of Ambainis and Lokam (Theorem B.25 [8]) for XOR combined functions [3].

1.3 Paper organization

The overview for the rest of the paper is as follows. We study the BSM model under computational restrictions on Carol. In Section 2, we restrict the Carol to be a polynomial-sized circuit, and show that the Split-SAT problem (SSAT) is unlikely to have polynomial sized BSM protocols and we extend the same to two other distributed variants of NP-complete problems. In Section 3, we restrict Carol to be an arithmetic circuit and in Section 4, we restrict Carol to be an algebraic polynomial over $\mathbb{F}_2$. In Section 5, we discuss the setting where Carol is a Turing machine and show that it is possible to decide any RE language (including the Halting problem) with preprocessing.

The full version [8] of the paper includes appendices omitted in this version due to space constraints. In Appendix A [8], we provide connections of BSM to related models. In particular, we exploit this connection to argue that most Boolean functions do not have efficient instance hiding schemes. In Appendix B [8], we discuss a few BSM protocols that are implicit in the existing literature.

2 Bounded Simultaneous Messages for NP problems

In this section we show that the existence of BSM for split variants of certain NP complete problems yields unlikely consequences, including better than brute-force circuits for all NP languages (Section 2.1), one-way functions (Section 2.2), and instance hiding schemes for all of NP (Section 2.3).

2.1 Split SAT and NP languages

▶ **Theorem 1.** *If* SSAT *has a* 2*-party BSM protocol of size* $s(n)$ *and message length* $\ell(n)$ *then every* L *in* NP *has circuit size* $O(\ell(p(n)) \cdot 2^{n/2}) + s(p(n))$ *for some polynomial* p*.*

Proof. We will assume without loss of generality that L rejects all inputs of odd length. This can be enforced using the encoding

$$L = \{0x\colon x \text{ is a yes instance of odd length}\} \cup \{11x\colon x \text{ is a yes instance of even length}\}.$$

Consider the following algorithm for L:

Preprocessing. Given an even input length n:

1. Construct the Cook-Levin CNF ϕ which accepts (x, y, z) for some z if and only if $xy \in L$ with $|x| = |y| = n/2$.
2. For every x with $|x| = n/2$, construct the CNF
   $$\alpha_x(u, v, z) = \text{“}(x = u) \wedge \phi(x, v, z)\text{”}, \qquad \text{where } |u| = |v| = n/2,$$
   then determine and store the message a_x that Alice sends in the BSM SSAT protocol when her input is α_x.
3. For every y with $|y| = n/2$, construct the CNF
   $$\beta_y(u, v, z) = \text{“}(y = v) \wedge \phi(u, y, z)\text{”}, \qquad \text{where } |u| = |v| = n/2,$$
   then determine and store the message b_y that Bob sends in the BSM SSAT protocol when his input is β_y.

Execution. On input xy with $|x| = |y| = n/2$, simulate Carol on messages a_x and b_y and output her answer.

Under the assumptions of the theorem this algorithm decides L because $\alpha_x \wedge \beta_y$ is satisfiable if and only if $\phi(x, y, \cdot)$ is. If $\phi(x, y, z)$ accepts then so does $\alpha_x(x, y, z) \wedge \beta_y(x, y, z)$, while if $\alpha_x(u, v, z) \wedge \beta_y(u, v, z)$ accepts it must be that $x = u$ and $y = v$ so $\phi(x, y, z)$ must also accept.

The circuit representation of this algorithm consists of two tables of size $\ell(p(n)) \cdot 2^{n/2}$, a proportional amount of hardware to select the messages a_x and b_y, and a copy of Carol's circuit, giving the bound on circuit size. ◀

▶ **Corollary 2.** *If there exists a language in* NP *that is not computable by any circuit family of size* $\widetilde{O}(2^{n/2})$*, then* SSAT *doesn't have a* 2*-party,* $\mathsf{poly}(n)$*-size BSM protocol.*

2.2 BSM hardness from worst-case one-way functions

We derive consequences of Corollary 2 to the worst-case insecurity of one-way functions.

A function $f\colon \{0,1\}^* \to \{0,1\}^*$ is length-preserving if $|f(x)| = |x|$ for all sufficiently long x. It is non-poly-shrinking if $|f(x)| \geq |x|^\epsilon$ for all sufficiently long x and some constant ϵ.

▶ **Proposition 3.** *If every* NP *problem has circuit size* $\mathsf{poly}(n) \cdot 2^{n/2}$ *then every efficient length-preserving function can be inverted on every output in size* $\mathsf{poly}(n) \cdot 2^{2n/3}$*.*

Proof. Let L be the language

$$L = \{(y, a)\colon \text{there exists } x \text{ such that } f(x) = y \text{ and } a \text{ is a prefix of } x\}.$$

By the assumption L has circuits C of size $\mathsf{poly}(|y| + |a|) \cdot 2^{(|y|+|a|)/2}$, which is at most $\mathsf{poly}(n) \cdot 2^{2n/3}$ assuming $|y| = n$ and $|a| \leq n/3$.

Given y in the image of f, the circuit C can be applied iteratively to find the first $n/3$ bits of a preimage x. Once those are found, the other $2n/3$ bits can be computed by brute-force search. The resulting preimage finder has circuit size $\mathsf{poly}(n) \cdot 2^{2n/3}$. ◀

▶ **Proposition 4.** *If every* NP *problem has circuit size* $\mathsf{poly}(n) \cdot 2^{n/2}$ *but* $\mathsf{E}_{\|}^{\mathsf{NP}}$ *requires exponential size nonadaptive SAT-oracle circuits then every efficient non-poly-shrinking function can be inverted on every output of length n in size* $\mathsf{poly}(n) \cdot 2^{n/2}$*.*

Proof. Klivans and van Melkebeek [26] (see also the discussion in [38]) show that under the second assumption there is an efficient deterministic algorithm that on input y of length n produces a list of circuits $V_1, \cdots, V_{\mathsf{poly}(n)}$ such that at least one of the circuits V_i accepts a *unique* x for which $f(x) = y$, assuming y is in the image of the function f. Let

$$L = \{(y, i, j)\colon \text{there exists } x \text{ for which } V_i(x) \text{ accepts and } x_j = 1\}.$$

As L is an NP language, by the first assumption, it has circuits C of size $\mathsf{poly}(|y| + |i| + |j|) \cdot 2^{(|y|+|i|+|j|)/2} = \mathsf{poly}(n) \cdot 2^{n/2}$. Using C, a preimage of y can be found among the rows of the table $x_{ij} = C(y, i, j)$. ◀

▶ **Corollary 5.** *If SSAT has a 2-party,* $\mathsf{poly}(n)$*-size BSM protocol then every efficiently computable function can be inverted on length-n outputs in (1) size* $\mathsf{poly}(n) \cdot 2^{2n/3}$ *assuming it is length-preserving; (2) size* $\mathsf{poly}(n) \cdot 2^{n/2}$ *assuming it is non-poly-shrinking and* $\mathsf{E}_{\|}^{\mathsf{NP}}$ *requires exponential size nonadaptive SAT-oracle circuits.*

2.3 Instance hiding and NP languages

A two-query *instance hiding (IH) scheme* [5] consists of size circuit (Henry) that receives an input and its output using queries to two oracles (Alice and Bob) such that the distribution of each query depends only on Henry's input length.

The instance hiding scheme is efficient if the size of Henry is polynomial in its input length. Our main result in this section is the following:

▶ **Theorem 6.** *If there is an efficient BSM protocol to one of the languages* SSAT*,* SCOL*,* SPRT*, then every language in* NP *has an efficient IH scheme.*

The connection between IH an BSM will be established using the following notion of reduction.

▶ **Definition 7.** *Let L_1, L_2 be a languages. A* split-hide reduction *from L_1 to L_2 is a pair of polynomial-time computable randomized mappings $a, b\colon \{0,1\}^n \to \{0,1\}^{\mathsf{poly}(n)}$ satisfying*

Correctness: $x \in L_1 \iff a(x)b(x) \in L_2$.

Privacy: The marginal distributions of $a(x)$ and $b(x)$ depend only on the length of x.

If such a reduction exists, we say that L_1 split-hide reduces *to L_2, and denote $L_1 \leq_p^{\mathsf{sh}} L_2$.*

▶ **Lemma 8.** *Let L_1, L_2 be two languages. If $L_1 \leq_p^{\mathsf{sh}} L_2$ and L_2 has an efficient BSM protocol, then L_1 has an efficient IH scheme with a single nonadaptive pair of queries.*

Proof. Suppose that L_2 has a 2-party efficient BSM protocol implemented by Alice, Bob, and Carol, and that $L_1 \leq_p^{\mathsf{sh}} L_2$ using mappings a and b. In the IH for L_1, on input x, Henry submits queries $a(x)$ and $b(x)$ to Alice and Bob, respectively, forwards their answers to Carol, and produces her output. ◀

▶ **Corollary 9.** *Let L_1, L_2 be a languages, and suppose that L_1 is* NP*-hard. If $L_1 \leq_p^{\mathsf{sh}} L_2$ and L_2 has an efficient BSM protocol, then every language in* NP *has an efficient IH scheme with a single nonadaptive pair of queries.*

Here, NP-hardness is assumed to hold under reductions that map all instances of a given length n into instances of the same length $m(n)$.

Proof. In the IH scheme for L_1, Henry implements the reduction from L_1 to L_2 and then runs the IH for L_2 from Lemma 8. ◀

Theorem 6 follows from Corollary 9 and the propositions below which show that SAT, 3COL, and PARTITION each split-hide reduce to their split variant.

▶ **Proposition 10.** SAT $\leq_p^{\mathsf{sh}}$ SSAT.

Proof. By the standard reduction from SAT to 3SAT we can represent the input instance by a 3CNF $\varphi(x_1, \dots, x_n)$, where n is chosen large enough to embed all SAT instances of a given size. We describe the randomized mappings a and b that yield CNFs α and β, respectively, in variables $x_1, \dots, x_n$ and additional auxiliary variables $y_1, \dots, y_{2N}$, where $N = 8\binom{n}{3}$. Let $C_1, \dots, C_N$ be a list of all possible clauses of width at most 3 over the n variables $x_1, \dots, x_n$. Let π be a random permutation of $[2N]$. We define

$$\alpha = \bigwedge_{i \in [N]} (C_i \vee y_{\pi(i)}).$$

Let $I = \{i : C_i \in \varphi\}$ and $J = \{N + i : C_i \notin \varphi\}$. We define

$$\beta = \bigwedge_{i \in I \cup J} \neg y_{\pi(i)}.$$

Correctness: We argue that φ is satisfiable if and only if $\alpha \wedge \beta$ is satisfiable.

- Suppose that φ is satisfiable. Then, we can satisfy α by taking the satisfying assignment of φ to satisfy each clause $(C_i \vee y_{\pi(i)})$ for $i \in I$, and assign a true value to $y_{\pi(i)}$ for every $i \in [N] \setminus I$. To satisfy β, assign a false value to $y_{\pi(i)}$ for every $i \in I$, which will not affect α's satisfiability, and assign a false value to $y_{\pi(i)}$ for every $i \in J$.
- Suppose that $\alpha \wedge \beta$ is satisfiable. Since β is satisfiable, it follows that the satisfying assignment assigns false to $y_{\pi(i)}$ for every $i \in I$, which implies that α can be satisfied by the same assignment only if C_i is satisfied for every $i \in I$, namely φ itself is satisfied.

Privacy: The y-variables in both α and β are random length N subsequences of $(y_1, \dots, y_{2N})$. ◀

▶ **Proposition 11.** 3COL $\le_p^{\mathsf{sh}}$ SCOL.

Proof. Given a graph $G = (V, E)$ with vertices $V = \{1, \dots, n\}$ the reduction produces the following two graphs A and B on the same vertex set $\tilde{V}$. Let $N = \binom{n}{2}$ and π be a random permutation of $[2N]$. We identify $[N]$ with ordered pairs of vertices in V.

- Vertices $\tilde{V}$: For each $v \in V$ there is a vertex v in $\tilde{V}$, and for each $j \in [2N]$ there are vertices $a_j, b_j, c_j, a'_j, b'_j, c'_j$.
- Edges in A: For each pair $(v < v') \in V$, let $j = \pi(v, v')$. We connect v to a_j, b_j, v' to a'_j, b'_j, and add triangles (a_j, b_j, c_j) and (a'_j, b'_j, c'_j).
- The edges in B are $\{(c_{\pi(e)}, c'_{\pi(e)}) : e \in E\} \cup \{(c_{\pi(N+e)}, c'_{\pi(N+e)}) : e \notin E\}$.

Correctness: If G is 3-colorable, the 3-coloring can be extended to a valid coloring of $A \cup B$ by coloring c_j and c'_j for $j = \pi(v, v')$ with the colors of v and v', respectively, and coloring a_j, b_j, a'_j, b'_j with the remaining colors. The vertices indexed by j that are not in the image of $\pi([N])$ are assigned fixed consistent colors, e.g., $(a_j, b_j, c_j, a'_j, b'_j, c'_j) \to (\mathtt{R}, \mathtt{B}, \mathtt{G}, \mathtt{B}, \mathtt{G}, \mathtt{R})$. Conversely, if $A \cup B$ is 3-colorable, v and c_j must have the same color and so must v' and $c_{j'}$; by the definition of B the coloring on V is valid for G.

Privacy: The graph A consists of N pairs of "diamonds" v, a_j, b_j, c_j and v', a'_j, b'_j, c'_j, one for each vertex-pair $v < v'$, where j is randomly assigned one of N distinct values in $[2N]$. The graph B is a matching chosen at random among those that match N out of the $2N$ pairs $(c_1, c'_1), \dots, (c_{2N}, c'_{2N})$. Both distributions depend on G only through n. ◀

▶ **Proposition 12.** PARTITION $\le_p^{\mathsf{sh}}$ SPRT.

Proof. The input to PARTITION problem consists of a multi-set $S = \{x_1, \dots, x_n\}$ of n non-negative integers (not necessarily distinct) of n-bits forming a multi-set S.

Reduction: Set $p = n2^n + 1$. For each $i \in \{0, 1, \dots, n-1\}$, pick a y_i uniformly at random from $\{0, 1, \dots, p-1\}$ and let z_i be the unique residue $(x_i - y_i) \mod p$. Define

$$\begin{aligned} Y_i &= y_i \cdot 8^n + 1 \cdot 8^i \\ Z_i &= z_i \cdot 8^n + 2 \cdot 8^i \\ W_i &= 3 \cdot 8^i \end{aligned}$$

Alice's and Bob's inputs are the sets

$$\begin{aligned} A &= \{Y_0, \dots, Y_{n-1}\} \\ B &= \{Z_0, \dots, Z_{n-1}\} \cup \{W_0, \dots, W_{n-1}\} \cup E, \qquad E = \{2^j \cdot p8^n \mid 0 \le j \le \log n\}. \end{aligned}$$

Correctness: We show that $U = A \cup B$ has a partition $\sum C = \sum D$ if and only if S has a partition $\sum P = \sum Q$, where $\sum X$ is the sum of all elements in a multi-set X. Let $U' = \{Y_i, Z_i, W_i \mid 0 \le i \le n-1\}$ so that $U' \cup E = U = A \cup B$. We show the following three statements are equivalent. Correctness follows from the equivalence of 1 and 3.

1. S has a partition $\sum P = \sum Q$
2. U' has a partition $\sum C' \equiv \sum D' \bmod p8^n$
3. U has a partition $\sum C = \sum D$.

$1 \to 2$: Assuming $\sum P = \sum Q$, For each $x_i \in P$, place Y_i, Z_i in C' and W_i in D'. For each $x_i \in Q$, place W_i in C' and Y_i, Z_i in D'. This ensures that $\sum C' - \sum D'$ is a multiple of 8^n and moreover $\sum C' - \sum D' = (\sum P - \sum Q)/8^n$. With P and Q of equal sums, $\sum C' - \sum D' \equiv 0 \bmod p8^n$, we can conclude that C', D' is the desired partition of U'.

$2 \to 1$: Assume $\sum C' \equiv \sum D' \bmod p8^n$. Then for every i (1) $Y_i, Z_i \in C'$ and $W_i \in D'$, or (2) $Y_i, Z_i \in D'$ and $W_i \in C'$, or (3) Y_i, Z_i, W_i do not appear in C' and D'. In all other cases the i-th least significant entry in the base-8 representation of $\sum C'$ and $\sum D'$ cannot match.

Assign every number y_i, z_i, w_i to P or Q depending on whether Y_i, Z_i, W_i is in C' or D'. Then $\sum C' \equiv 8^n \sum P + \sum 3 \cdot 8^i$ and $\sum D' \equiv 8^n \sum Q + \sum 3 \cdot 8^i$ modulo $p8^n$, where the second summation is over those indices i that satisfy (1) or (2). Therefore $\sum P \equiv \sum Q$ modulo p. By our choice of p, $\sum P = \sum Q$.

$2 \to 3$: If $\sum C' \equiv \sum D' \bmod p8^n$, then $\sum C' - \sum D' = kp8^n$ for some $-n \le k \le n$. Let E_k be the (unique) subset of E that sums to $|k|\, p8^n$. If $k > 0$ set $C = C'$, $D = D' \cup E_k$. Otherwise, set $C = C' \cup E_k$, $D = D'$. In either case $\sum C = \sum D$.

$3 \to 2$: C' and D' are obtained from C and D by dropping the elements in E.

Privacy: The marginal distributions of y_i as well as z_i are independent of the input instance S and depends only on n. The set A consists of Y_i and B consists of Z_i, W_i and some fixed additional elements none of which depend on the set S. Hence, the marginal distributions A and B are also independent of S and depends only on n. ◀

3 Arithmetic circuits for Matrix Multiplication

An *arithmetic BSM protocol* for a polynomial $p(X, Y)$ is a decomposition of the form $p(X, Y) = \text{Carol}(\text{Alice}(X), \text{Bob}(Y))$. The complexity of the protocol is the smallest possible arithmetic circuit complexity of Carol.

We show that BSM protocols do not help for circuit multiplication. Recall that the *tensor rank* $R(n)$ of $n \times n$ matrix multiplication is the smallest possible number of terms in a decomposition of XY as a sum of products of linear functions in X and Y, respectively. Asymptotically, $\log_n R(n)$ converges to the matrix multiplication exponent ω.

▶ **Theorem 13.** *In any arithmetic BSM protocol for multiplying $n \times n$ matrices Carol must use at least $R(n)/2$ multiplication gates.*

As matrix multiplication can be realized in complexity $O(R(n))$ without preprocessing, BSM does not offer any savings in this setting.

Proof. We prove the contrapositive: A BSM protocol in which Carol uses t multiplication gates yields a representation of XY of tensor rank at most $2t$.

For a polynomial P in the entries of both X and Y, let $c(P)$ be the constant part, $a(P)$ be the linear part involving entries from X, $b(P)$ be the linear part involving entries from Y, and $ab(P)$ be the bilinear part, where each monomial is a product of an entry of X and an entry of Y.

We construct a circuit that computes the low-degree part $\ell(C) = (c(C), a(C), b(C), ab(C))$ using at most $2t$ multiplication gates inductively over the size of Carol's circuit C. For the base case, Carol has size zero its output must come either from Alice of from Bob, so ab must be zero and $q(C)$ is a linear function of X or Y requiring no multiplications. For the inductive step, the following rules show that computing $\ell(P + Q)$ from $\ell(P)$ and $\ell(Q)$ takes no extra multiplications, while computing $\ell(PQ)$ takes at most two extra multiplications (underlined):

$$\begin{array}{l|l}
c(P+Q) = c(P) + c(Q) & c(PQ) = c(P)c(Q) \\
a(P+Q) = a(P) + a(Q) & a(PQ) = c(P)a(Q) + a(P)c(Q) \\
b(P+Q) = b(P) + b(Q) & b(PQ) = c(P)b(Q) + b(P)c(Q) \\
ab(P+Q) = ab(P) + ab(Q) & ab(PQ) = c(P)ab(Q) + ab(P)c(Q) + \underline{a(P) \cdot b(Q)} + \underline{b(P) \cdot a(Q)}
\end{array}$$

As Carol's output is some linear combination of its multiplication gates, $\ell(XY)$ can be computed using at most twice the number of multiplications used by Carol as desired. ◀

4 Polynomials of bounded degree

Every Boolean function $f\colon \{0,1\}^N \to \{0,1\}$ can be represented by a multilinear polynomial of degree at most N over the reals. In this section we ask whether we can reduce the degree by means of preprocessing. Given a function $f\colon \{0,1\}^{2n} \to \{0,1\}$, let $d_f(n,m)$ be the least degree of a polynomial Carol for which $f(x,y) = \mathrm{Carol}(\mathrm{Alice}(x), \mathrm{Bob}(y))$ and Alice, Bob output at most m bits each.

This type of representation provides dramatic savings for $\mathrm{AND}(x,y) = x_1 \cdot\dots\cdot x_n \cdot y_1 \cdot\dots\cdot y_n$, which has degree n, yet $d_{\mathrm{AND}}(n,1) \le 2$. This separation is best possible because the only functions with $d_f(n,m) = 1$ are direct sums.

Let $d(n,m) := \max\left\{d_f(n,m)\colon f\colon \{0,1\}^{2n} \to \{0,1\}\right\}$. Beaver at al. [6] showed (in a different context) that for any constant $C > 0$,

$$d(n, n^{C+1}) \le \frac{2n}{C \log n}.$$

We extend their argument to obtain the following bound.

▶ **Proposition 14.** *For $n < m < 2^n$, it holds that $d(n,m) = O(n/\log(m/n))$.*

Proof. Let $f\colon \{0,1\}^{2n} \to \{0,1\}$, and let p be a multilinear polynomial of degree $2n$ that computes f:

$$p(z_1, \dots, z_{2n}) = \sum_{S \subseteq [2n]} c_S \cdot z^S.$$

Consider now the following strategy to reduce the degree of p: split the input z into n/t disjoints set of coordinates, each of size t, for some chosen $t \le n$. For each coordinate subset $T \subseteq [2n]$ of size T, we define $2^t - 1$ variable as follows: for each $\emptyset \ne J \subseteq T$:

$$Z_J = \prod_{i \in J} z_i.$$

It follows that we can express p as a polynomial of degree $\lceil 2n/t \rceil$ by taking each monomial z^S and replacing it with a product of Z_J's for proper choices of J's.

We can now establish a BSM protocol for f with Carol computing a degree-$(2n/t)$ polynomial by letting Alice and Bob compute the Z_J's and send the results to Carol. This requires preprocessing output length of at most $\lceil n/t \rceil \cdot (2^t - 1)$. Choosing

$$t = \lceil \log(m/(4n)) \rceil \le \log(m/(4n)) + 1 = \log(m/n) - 1 \le \log(2^n/n) < n$$

implies preprocessing of length at most

$$\lceil n/t \rceil \cdot (2^t - 1) \le (n/t + 1) \cdot 2^t \underset{t \le n}{\le} (2n/t) \cdot 2^t = (2n/t) \cdot 2^{\lceil \log(m/(4n)) \rceil} \le m/t \underset{t \ge 1}{\le} m.$$

Thus,

$$d(n,m) = \lceil 2n/t \rceil \le 2n/t + 1 = \frac{2n}{\lceil \log(m/(4n)) \rceil} + 1 \le \frac{2n}{\log(m/(4n))} + 1 = O(n/\log(m/n)). \quad ◀$$

The following proposition about the equality function shows that this bound is almost tight.

▶ **Proposition 15.** *For $n \le m$, $d_{\mathrm{EQ}}(n,m) = \Omega(n/\log m)$.*

Proof. Let us start by considering the case in which the $\mathbb{F}_2$-polynomial has *individual degree* 1, where individual degree-1 means that each variable taken from either Alice's message or Bob's message occurs at most once in any monomial computed by Carol. This means that it is of the form

$$P(z,w) = \sum_{ij} c_{ij} z_i w_j + \sum_i d_i z_i + \sum_j e_j w_j + r.$$

Now consider the n-bit equality function EQ. Suppose that it can be calculated using preprocessing of length m. That is, there exist functions $A, B \colon \{0,1\}^n \to \{0,1\}^m$ and a polynomial P as above such that for all $x, y \in \{0,1\}^n$:

$$\mathrm{EQ}(x,y) = P(A(x), B(y)).$$

For each $x \in \{0,1\}^n$, $Q_x(y) = P(A(x), B(y))$ is an affine form on m variables. Any $m+2$ such forms must be linearly dependent. Hence if $m + 2 \le 2^n$, we can find $x_0, \dots, x_m$ such that

$$Q_{x_0}(y) = Q_{x_1}(y) + \cdots + Q_{x_m}(y).$$

Substituting $y = x_0$, we obtain a contradiction, since the left-hand side should be 1, while the right-hand side is a sum of zeroes. We conclude that $m \ge 2^n - 1$. (This should be compared to the obvious upper bound in which $m = 2^n$.)

Now suppose that $\mathrm{EQ}(x,y) = P(A(x), B(y))$, where P is an arbitrary polynomial of degree d, and $|A(x)| = |B(x)| = m$. Let $A'(x)$ consist of all ANDs of up to d elements from $A(x)$, and define $B'(x)$ similarly. Thus, $|A'(x)| = |B'(x)| = \binom{m}{\le d}$. We can find a polynomial P' of individual degree 1 such that $P'(A'(x), B'(y)) = P(A(x), B(y))$. Therefore,

$$2^n - 1 \le \binom{m}{\le d} < (m+1)^d \implies d \ge \frac{n}{\log(m+1)}. \qquad ◀$$

Combining the last two propositions, we have a tight bound when m is polynomial in n:

▶ **Corollary 16.** *For every constant $C > 1$, $d(n, n^C) = \Theta(n/\log n)$.*

5 Computability with preprocessing

Language $L \subseteq \{0,1\}^* \times \{0,1\}^*$ is BSM-computable with message size $m(n)$ if there is a BSM protocol for L in which

1. Carol is a Turing Machine that receives $m(|x|)$ and $m(|y|)$-bit messages from Alice and Bob on input (x, y)
2. Carol halts on all inputs of the form Alice(x), Bob(y).

In this setting Alice and Bob do not know each other's input length.

▶ **Proposition 17.** *Every recursively enumerable language is BSM-computable with message size $2n + 1$.*

As BSM-computability is closed under complement, the proposition extends to co-recursively enumerable languages.

Proof. Let M be a Turing Machine that recognizes L, i.e., it halts on yes-instances only. The following BSM protocol computes L:

- On input x, Alice sends Carol x and the number of pairs (x, y') with $|y'| \le |x|$ such that $xy' \in L$. On input y, Bob sends Carol y and the number of pairs (x', y) with $|x'| \le |y|$ such that $x'y \in L$.

- If $|x| \leq |y|$, Carol runs M on all inputs xy' for $|y'| \leq |x|$ in parallel and halts whenever the number of accepted instances reaches the value sent by Alice. Carol accepts iff (x, y) is among them. If $|x| < |y|$, Alice's and Bob's roles are reversed. ◀

If condition 2 in the definition of BSM-computability were replaced with the stronger requirement that Carol halts on all inputs, Proposition 17 would no longer be true: It is possible to construct a recursively enumerable language that is not BSM-computable in this stronger sense whenever $m(n) = o(2^{n/2})$ by diagonalizing against every possible Carol [31].

6 Conclusion and Open Problems

In this work, we initiate a systematic study of the BSM complexity of several natural classes of functions with different assumptions on Carol: size-bounded, degree-bounded, and arithmetic (summarized in Table 1). Our work suggests several natural open problems.

- *Size-bounded Carol.* Can preprocessing help improve the matrix multiplication time when Carol is a Boolean circuit? Our negative result only considered an arithmetic Carol over the reals (Theorem 13). To argue against the possibility of BSM with polynomial-size Carol for arbitrary NP languages, we suggested two explicit candidates for NP languages with the highest possible circuit complexity: Turing Machine Acceptance and Succinct Subset Sum (see Section 1.1 for the exact conjectures). Studying these conjectures and proposing other natural candidates for maximally hard NP languages may be of independent interest. A final open question is proving an analog of Theorem 1 for SCOL, SPRT or split-versions of some other NP complete languages, where the natural Cook-Levin approach does not seem to apply.
- *Depth-bounded Carol.* We do not have any results pertaining to a depth-bounded Carol. However, the prior result of [16] which shows a negative result for a BSM protocol for the mod-2 inner product function by AC^0 circuits, leaves a non-trivial open question to solve. Can we prove even a weak unconditional lower bound for the mod-3 inner product by AC^0 circuits with mod-2 gates?

References

1 Scott Aaronson and Avi Wigderson. Algebrization: a new barrier in complexity theory. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 731–740, 2008. doi:10.1145/1374376.1374481.

2 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10–13, 2021*, pages 522–539, 2021. doi:10.1137/1.9781611976465.32.

3 Andris Ambainis and Satyanarayana V. Lokam. Improved upper bounds on the simultaneous messages complexity of the generalized addressing function. In *LATIN 2000: Theoretical Informatics, 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000, Proceedings*, volume 1776 of *Lecture Notes in Computer Science*, pages 207–216, 2000. doi:10.1007/10719839_21.

4 László Babai, Anna Gál, Peter G. Kimmel, and Satyanarayana V. Lokam. Communication complexity of simultaneous messages. *SIAM J. Comput.*, 33(1):137–166, 2003. doi:10.1137/S0097539700375944.

5 Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In *STACS 90, 7th Annual Symposium on Theoretical Aspects of Computer Science, Rouen, France, February 22-24, 1990, Proceedings*, pages 37–48, 1990. doi:10.1007/3-540-52282-4_30.

6 Donald Beaver, Joan Feigenbaum, Joe Kilian, and Phillip Rogaway. Locally random reductions: Improvements and applications. *J. Cryptol.*, 10(1):17–36, 1997. doi:10.1007/s001459900017.

7 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10, 1988. doi:10.1145/62212.62213.

8 Andrej Bogdanov, Krishnamoorthy Dinesh, Yuval Filmus, Yuval Ishai, Avi Kaplan, and Sruthi Sekar. Bounded simultaneous messages, 2023. arXiv:2310.00334.

9 Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. *SIAM J. Comput.*, 43(2):831–871, 2014. doi:10.1137/120868669.

10 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19, 1988. doi:10.1145/62212.62214.

11 Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 41–50, 1995. doi:10.1109/SFCS.1995.492461.

12 Gil Cohen and Igor Shinkar. The complexity of DNF of parities. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 47–58, 2016. doi:10.1145/2840728.2840734.

13 Don Coppersmith and Shmuel Winograd. On the asymptotic complexity of matrix multiplication (extended summary). In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 82–90, 1981. doi:10.1109/SFCS.1981.27.

14 A. M. Davie and A. J. Stothers. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh Section A: Mathematics*, 143(2):351–369, 2013. doi:10.1017/S0308210511001648.

15 Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM Journal on Computing*, 29(3):790–803, 2000. doi:10.1137/S0097539795280512.

16 Yuval Filmus, Yuval Ishai, Avi Kaplan, and Guy Kindler. Limits of preprocessing. In *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 17:1–17:22, 2020. doi:10.4230/LIPIcs.CCC.2020.17.

17 Lance Fortnow and Mario Szegedy. On the power of two-local random reductions. In *Advances in Cryptology – ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, volume 739 of *Lecture Notes in Computer Science*, pages 346–351, 1991. doi:10.1007/3-540-57332-1_29.

18 Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 – June 2, 2009*, pages 169–178. ACM, 2009. doi:10.1145/1536414.1536440.

19 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, 1987. doi:10.1145/28395.28420.

20 Vince Grolmusz. Superpolynomial size set-systems with restricted intersections mod 6 and explicit ramsey graphs. *Comb.*, 20(1):71–86, 2000. doi:10.1007/s004930070032.

21 M. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980. doi:10.1109/TIT.1980.1056220.

22 Ellis Horowitz and Sartaj Sahni. Computing partitions with applications to the knapsack problem. *J. ACM*, 21(2):277–292, April 1974. doi:10.1145/321812.321823.

23 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer – efficiently. In David A. Wagner, editor, *Advances in Cryptology – CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008. doi:10.1007/978-3-540-85174-5_32.

24 Stasys Jukna. On graph complexity. *Comb. Probab. Comput.*, 15(6):855–876, 2006. doi:10.1017/S0963548306007620.

25 Joe Kilian. Founding cryptography on oblivious transfer. In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31. ACM, 1988. doi:10.1145/62212.62215.

26 Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002. doi:10.1137/S0097539700389652.

27 Satyanarayana V. Lokam. Graph complexity and slice functions. *Theory Comput. Syst.*, 36(1):71–88, 2003. doi:10.1007/s00224-002-1068-0.

28 Satyanarayana V. Lokam. Complexity lower bounds using linear algebra. *Found. Trends Theor. Comput. Sci.*, 4(1-2):1–155, 2009. doi:10.1561/0400000011.

29 Jesper Nederlof and Karol Wegrzycki. Improving Schroeppel and Shamir's algorithm for subset sum via orthogonal vectors. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 1670–1683, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3406325.3451024.

30 Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM Journal on Computing*, 22(1):211–219, 1993. doi:10.1137/0222016.

31 P. Odifreddi. *Classical Recursion Theory: The Theory of Functions and Sets of Natural Numbers.* ISSN. Elsevier Science, 1992. URL: https://books.google.co.in/books?id=ahBLqshTFMwC.

32 Ramamohan Paturi, Michael E. Saks, and Francis Zane. Exponential lower bounds for depth three boolean circuits. *Comput. Complex.*, 9(1):1–15, 2000. doi:10.1007/PL00001598.

33 Pavel Pudlák, Vojtech Rödl, and Petr Savický. Graph complexity. *Acta Informatica*, 25(5):515–535, 1988. doi:10.1007/BF00279952.

34 Pavel Pudlák, Vojtech Rödl, and Jirí Sgall. Boolean circuits, tensor ranks, and communication complexity. *SIAM Journal on Computing*, 26(3):605–633, 1997. doi:10.1137/S0097539794264809.

35 Alexander A. Razborov and Steven Rudich. Natural proofs. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 204–213, 1994. doi:10.1145/195058.195134.

36 Guy N. Rothblum. How to compute under AC^0 leakage without secure hardware. In *Advances in Cryptology – CRYPTO 2012 – 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 552–569, 2012. doi:10.1007/978-3-642-32009-5_32.

37 Richard Schroeppel and Adi Shamir. A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems. *SIAM Journal on Computing*, 10:456–464, 1981.

38 Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *computational complexity*, 15(4):298–341, 2006.

39 Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, August 1969. doi:10.1007/bf02165411.

40 Avishay Tal. Formula lower bounds via the quantum method. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1256–1268, 2017. doi:10.1145/3055399.3055472.

41 Rahul Tripathi. On the computational power of locally random reductions. *A preliminary version appeared in MFCS 2007*, 2010. URL: https://api.semanticscholar.org/CorpusID:17801736.

42 Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 – May 2, 1979, Atlanta, Georgia, USA*, pages 209–213, 1979. `doi:10.1145/800135.804414`.

43 Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167, 1986. `doi:10.1109/SFCS.1986.25`.