Lower bounds for the depth of modular squaring

Benjamin Wesolowski, Ryan Williams

Lower bounds for the depth of modular squaring

Benjamin Wesolowski[1,2] and Ryan Williams*[3]

[1] Univ. Bordeaux, CNRS, Bordeaux INP, IMB, UMR 5251, F-33400, Talence, France
[2] INRIA, IMB, UMR 5251, F-33400, Talence, France
[3] Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139

Abstract. The modular squaring operation has attracted significant attention due to its potential in constructing cryptographic time-lock puzzles and verifiable delay functions. In such applications, it is important to understand precisely how quickly a modular squaring operation can be computed, even in parallel on dedicated hardware. We use tools from circuit complexity and number theory to prove concrete numerical lower bounds for squaring on a parallel machine, yielding nontrivial results for practical input bitlengths. For example, for $n = 2048$, we prove that every logic circuit (over AND, OR, NAND, NOR gates of fan-in two) computing modular squaring on all n-bit inputs (and any modulus that is at least 2^{n-1}) requires depth (critical path length) at least 12. By a careful analysis of certain exponential Gauss sums related to the low-order bit of modular squaring, we also extend our results to the average case. For example, our results imply that every logic circuit (over any fan-in two basis) computing modular squaring on at least 76% of all 2048-bit inputs (for any RSA modulus that is at least 2^{n-1}) requires depth at least 9.

Keywords: Verifiable delay function · circuit · modular squaring · RSA

1 Introduction

Which computational problems are *inherently sequential*, and cannot be efficiently parallelized? This is the intuitive question behind the P versus NC problem, one of the major open problems in computational complexity theory and computer science. It is widely conjectured that $\mathsf{P} \neq \mathsf{NC}$, which would show that there are (natural) problems solvable in $n^{O(1)}$ time in a sequential computational model that cannot be solved in $(\log n)^{O(1)}$ parallel time (a.k.a. $(\log n)^{O(1)}$ *depth*) in a parallel computational model with $n^{O(1)}$ processors. Very little is known about the depth complexity of functions in P: the best known depth lower bounds for any function in P (or NP for that matter) only show that depth must be at least $c \log n$ where $c \leq 3$.

In cryptography, certain problems have been proposed as candidates for being inherently sequential. In this paper, we focus on the problem of integer modular squaring, which was first proposed by Rivest, Shamir, and Wagner [RSW96] in the context of time-lock puzzles. It has recently received renewed attention, when it was shown in [Wes19, Pie19] that it allows to build verifiable delay functions [BBBF18] (or *VDF*), a cryptographic primitive key to a wide variety of exciting applications, notably for secure decentralised systems.

Repeated Modular Squaring (RMS): Given an n-bit x, a parameter T, and an n-bit modulus $m \in [2^{n-1}, 2^n - 1]$, calculate $x^{2^T} \bmod m$.

*Email: rrw@mit.edu.

Typically, we think of T as "large". If m is prime, one can calculate x^{2^T} mod m very efficiently by first reducing the exponent $k = 2^T \bmod m - 1$, then computing x^k mod m for $k < m$. More generaly, knowing $\phi(m)$ (or equivalently, the factorisation of m) allows to compute x^{2^T} mod m in about $\log_2(T)$ multiplications in $\mathbf{Z}/m\mathbf{Z}$. However, if the factorisation of m is unknown, one cannot reduce the exponent, and there is no known method to compute x^{2^T} mod m faster than squaring T times sequentially. In this situation, Rivest, Shamir and Wagner suggest that the depth of computing x^{2^T} mod m is roughly T times the depth of squaring modulo m, when $T \leq 2^{n^{o(1)}}$ (when T is sufficiently large, it becomes advantageous to factor m first).

The parallel setting is critical here: for applications of VDFs, it is important to have a good estimate of the *wall-clock* time required for computing RMS: what is the *depth* of the shallowest circuit an adversary can build? Naively, one might conjecture that every (reasonably defined) parallel machine computing RMS requires at least $\tilde{\Omega}(T + \log n)$ depth, where the $\tilde{\Omega}$ omits division by polylogarithmic factors. However, noting that the input length to RMS is only $m \leq O(\log T + \log n)$, such a conjecture would imply a $2^{\Omega(m)}$ circuit depth lower bound, but it is well-known that *every* function on m-bit input has a (non-uniform) circuit of 2^m size and $O(m)$ depth. So for such a conjecture to make sense, we also need to bound the size/work of the parallel computation as well, to be (for example) polynomial in T and $\log n$.

Other constructions with lower depth are known, such as [BS07], but require models whose practicality has been called into question [MOS20], with high fan-in and fan-out. To this day, the most successful attempts at building fast hardware for RMS still boil down to efficient modular squaring [Özt20, MOS20].

In this article, we focus on modular squaring, and ask the relaxed question: *what unconditional lower bounds* **can** *be proved for modular integer squaring?* Can it be rigorously proved that even computing certain bits of x^2 already requires an interesting level of depth complexity?

1.1 Our Results

In this paper, we focus on the following problem.

> **Modular Squaring (MS):** Given an n-bit x and an n-bit modulus $m \in [2^{n-1}, 2^n]$, calculate x^2 mod m.

Our goal is to prove unconditional, *concrete* depth lower bounds for MS. That is, we want to be able to plug in values of n arising in practice, and obtain unconditionally true facts about the required time delays of any logic circuit computing MS on n-bit numbers. To this end, we prove interesting worst-case lower bounds and average-case lower bounds on the depth complexity of MS. We state our results in terms of Boolean logic circuits over OR, AND, NAND, and NOR, each of fan-in two, with "free" NOT gates having cost 0. (We believe this choice of gates is reasonable for practice, because the other possible two-bit gates XOR and EQUALS[1] in modern hardware are, to the best of our knowledge, always implemented in terms of the other gates.) In circuit complexity terminology, this corresponds to proving circuit lower bounds over the basis U_2 (all unate functions on two bits [Weg87, Juk12]).

Worst-Case Lower Bounds. Our primary worst-case lower bound is the following.

[1] An EQUALS gate takes two bits x and y and outputs 1 if and only if $x = y$. EQUALS is simply the complement of XOR.

Theorem 1 (Worst-Case Lower Bound)**.** *For every n, every logic circuit (over the basis U_2) computing MS on n-bit integers requires depth at least $2\log_2(n-1)-2\log_2(\log_2(n-1)-1)-4$.*

In particular, we show that "middle bits" of the output of MS require such depth complexity. The depth complexity of Theorem 1 is stated rather precisely, so that it is possible to plug in concrete numbers and yield interesting results. For example, when $n = 1024$ and $n = 2048$, in which case Theorem 1 yields a depth lower bounds of at least 10 and 12, respectively (note that the depth must be an integer). Thus, Theorem 1 allows us to make concrete statements about the delays of logic circuits computing MS, such as the following:

> *Every logic circuit with a* 10 *picosecond delay per gate must take at least* 1200 *picoseconds to compute a modular squaring operation on* 2048*-bit integers.*

The lower bound of Theorem 1 proceeds by an efficient reduction from the SUM problem from n bits to $O(\log n)$ bits, defined as:

$$\mathsf{SUM}(x_1, \ldots, x_n) := \sum_i x_i.$$

That is, SUM simply outputs a binary representation of the sum of the 1s in its input. It is known that SUM has circuits of depth at most $4.14\log_2(n)$ [Ser16] and every circuit for SUM must have depth at least $2\log_2(n)$ [Kra70]. Our reduction can be stated as follows.

Theorem 2. *If* SUM *requires depth at least $c\log_2(n)$ for constant $c \geq 2$, then MS on n-bit integers requires depth at least $c\log_2(n-1) - c\log_2(\log_2(n-1)-1) - 2c$.*

As a corollary, under the reasonable hypothesis that SUM requires depth $4\log_2(n)$, MS on 2048-bit integers has depth complexity at least 23.

Average-Case Lower Bounds. For cryptographic reasons, it is interesting to ask if the above lower bounds hold for circuits that are only required to compute MS on a decent fraction of n-bit inputs. In this case we can also prove non-trivial depth lower bounds for squaring with an RSA modulus, but they are weaker than what we can prove in the worst case.

Theorem 3. *Let $n \geq 38$, and let m be the product of two primes in $[2^{n/2-1}, 2^{n/2})$. Every logic circuit computing $(x^2 \bmod m)$ correctly on at least 76% of all n-bit integers x (over the basis U_2 or B_2) requires depth at least $\log_2(n/4 - 7.2)$.*

Corollary 1. *Every logic circuit (even with XOR and EQUALS gates) computing $(x^2 \bmod m)$ correctly on at least 76% of all n-bit integers x requires depth at least 9, when m is the product of two 1024-bit primes, both of which are at least 2^{1023}.*

An interesting aspect of Theorem 3 is that the depth lower bound holds even over circuits from B_2, the *full binary basis*: our gates may be any possible function that takes two bits as input and outputs a bit. (This point is not necessarily relevant to practice; it just turns out that the proof works in that case.)

We believe that the depth complexity of MS in the average case should be roughly as hard as the worst case. However, our worst-case proofs reasoning about the "middle bit" of MS do not extend to the average case (they modify the input too much to yield "typical" inputs), so we take a different approach. We show that computing the *low-order bit* of MS is difficult on average. This is interesting, because the low-order bit of normal *integer squaring*, without modular reduction, is trivial (the low-order bit of the output only depends on the low-order bit of the input). Thus modular reduction plays a crucial role in the average-case lower bound of Theorem 3.

In order to establish Theorem 3, we prove a result regarding the sensitivity of modular squaring on random inputs, and show how such sensitivity results yield size and depth lower bounds. (Here in the introduction, we will keep the discussion at an informal level.) The sensitivity of a function f at a binary input x measures how many distinct input bits could be "flipped" such that the value of f changes. When f is the low-order bit of the MS function on n-bit integers, we use bounds on exponential sums to show (in Theorems 10 and 11) that there are at least $n/4$ input bits x_i such that, when an n-bit x is chosen uniformly at random, the probability that the value of $f(x)$ flips is at least $1/2 - o(1)$. We show how such sensitivity results can be applied to prove lower bounds on the depths of all circuits computing MS even on a decent fraction of inputs, culminating in Theorem 3.

2 Preliminaries

Notation and Definitions. Let $n \geq 1$ be an integer, and let $[n] := \{1, \ldots, n\}$. We define the function $\mathsf{IS} : \{0,1\}^\star \to \{0,1\}^\star$ (Integer Squaring) to map n-bit integers x to $2n$-bit integers x^2.

Let $m \leq 2^n - 1$ be an odd integer, and let $\mathbf{Z}/m\mathbf{Z} = \{0, 1, \ldots, m-1\}$. We view $(x \bmod m)$ as a function from $\mathbf{Z}$ to the set $\mathbf{Z}/m\mathbf{Z}$. Define

$$\text{MS-MOD2}_{n,m} : \{0,1\}^n \to \{0,1\}$$

as the function which construes its n-bit input x as an integer in $[0, 2^n - 1]$, and outputs $((x^2 \bmod m) \bmod 2)$. This is the least significant bit of the function $MS(x) = (x^2 \bmod m)$ as defined in the introduction.

We use basic notions from circuit complexity [Weg87, Juk12]. Let $\mathcal{B}$ be a set of functions of the form $f : \{0,1\}^k \to \{0,1\}$ where $k \in \{1,2\}$. Typical choices are B_2, the set of all functions on at most two inputs, and U_2, the set of all unate functions on at most two inputs (a unate function g is a monotone function with signs on some of its inputs, so for example $g(x,y,z) = f(\neg x, y, \neg z)$ is unate when f is monotone). It is easy to verify that $B_2 = U_2 \cup \{XOR, EQUALS\}$.

As is standard, an n-input, m-output, s-size circuit over the basis $\mathcal{B}$ is represented as a directed acyclic graph with n sources and s other nodes which include m sinks. Each node is called a *gate*, has indegree at most two (a.k.a. fan-in at most two), and is labelled with some $f \in \mathcal{B}$.

More background on circuit complexity will be covered in Section 3.

Prior Work. As far as we can tell, there is scant literature on proving *numerical* and unconditional lower bounds for circuit depth. Meyer and Stockmeyer [SM02] proved such results for circuit size: concretely, they showed that deciding the truth of formulas of length at most 610 from the language WS1S (a problem known to require extremely high asymptotic time complexity) requires circuits containing at least 10^{125} gates.

On a technical level, the work most related to ours concerns depth lower bounds for PRAMs (parallel random access machines). It has long been known that sensitivity is tied to running time for parallel machines (a.k.a. circuit depth) [Weg87]. For example, the work of Cook, Dwork, and Reishuk [CDR86] implies that for so-called CREW (Concurrent Read Exclusive Write) PRAMs, the minimal running time for computing a function f is at least $\Omega(\log S(f))$ where $S(f)$ is the sensitivity of f, and the recently celebrated proof of the Sensitivity Conjecture by Huang [Hua19] implies (among other results) that CREW PRAM running time is bounded from above by $O(\log S(f))$. Among this literature, the reference most related to our work is that of Shlarpinski and von zur Gathen [vzGS00], who used bounds on exponential sums and sensitivity to show that modular inversion requires depth $\Omega(\log_2(n))$ on n-bit integers in the worst case, with a leading constant at

most 1/2 in the $\Omega(\cdot)$. To contrast, our results for modular squaring are worst-case with a leading constant of (at least) 2, and average-case with a leading constant 1 in front of the $\log_2(n)$ term, allowing us to conclude interesting numerical lower bounds.

3 Useful Notions from Circuit Complexity

In this section, we generalize some known concepts and theorems from circuit complexity which will be useful in our average-case depth lower bounds. Two good general references on the subject are [Weg87, Juk12]. In the following, let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. For a point $x \in \{0,1\}^n$ and index $i \in [n]$, we define $x^{(i)}$ to be x with the i-th bit flipped to the opposite value. All of the following results hold over any basis of fan-in two Boolean functions, including B_2.

3.1 Non-Degeneracy

Roughly speaking, a function f is non-degenerate if it depends on all of its inputs.

Definition 1. A function f is *non-degenerate* if for all $i \in [n]$, there is an $a \in \{0,1\}^n$ such that $f(a) \neq f(a^{(i)})$.

That is, f is non-degenerate if, for all i, there is at least one input a such that flipping the i-th bit of a changes the value of f. A function f is *degenerate* if it is not non-degenerate. Intuitively, a degenerate function has at least one variable that does not affect the output: there is an $i \in [n]$ such that for *all* assignments a, $f(a) = f(a^{(i)})$. Therefore the i-th bit of input does not affect the output.

It is not difficult to prove the following relation between non-degeneracy and circuit size. (We omit the proof, as we will prove a stronger result in a moment.)

Proposition 1 (Folklore, [Weg87])**.** *Every non-degenerate f has circuit complexity at least $n-1$, over the basis of all Boolean functions on two inputs.*

Corollary 2. *The AND, OR, and PARITY functions on n-bit inputs require circuits of size at least $n-1$.*

Proposition 2 (Folklore)**.** *Every non-degenerate f has depth complexity at least $\log_2(n)$, over the basis of all Boolean functions on two inputs.*

Proof. The proof is by contrapositive. Consider any circuit C of depth $d < \log_2(n)$. Since C has fan-in two and depth d, the circuit C can be converted into an equivalent formula F, with at most $2^d \leq n-1$ leaves. This equivalent formula F can read at most $n-1$ distinct inputs, so the function computed by F is degenerate (the missing input cannot affect the output of f). □

Corollary 3. *The AND, OR, and PARITY functions on n-bit inputs require circuits of depth at least $\log_2(n)$.*

In the proofs of average-case lower bounds, we will require a natural generalization of the classic non-degeneracy concept, and its application to circuit complexity.

Definition 2. Let $k \in \{0,1,\ldots,n\}$. A function f is *degenerate in k variables* if there are k distinct $i_1,\ldots,i_k \in [n]$ such that for all $a \in \{0,1\}^n$ and $j \in [k]$, $f(a) = f(a^{(i_j)})$.

That is, f is degenerate in k variables if there are k different input indices i such that flipping the i-th bit of a does not change the value of f, for all inputs a. The following generalizes Proposition 1.

Proposition 3. *Let $k \in \{0, 1, \ldots, n\}$. Every minimal fan-in two circuit with at most $(n-1)-k$ gates is degenerate in (at least) k variables.*

Proof. We prove the contrapositive. Let s be the number of gates in a minimal circuit for an f that is **not** degenerate in k variables. First, there must be at least $n-k$ inputs in the circuit that have at least one outgoing wire. Otherwise, f would trivially be degenerate in (at least) k variables: k of the inputs have no outgoing wire, so flipping any of their values cannot change the circuit's output on *any* input. Note that each of the s gates have at least one outgoing wire: otherwise, that gate could be removed without changing the circuit's functionality, contradicting minimality.

Therefore the number of wires in a minimum circuit for f is at least $n-k+s-1$: there are at least $n-k$ wires coming out of the inputs, and $s-1$ wires coming out of the interior gates (subtracting one for the output gate, which has no outgoing wire). Since each gate also has fan-in two (two edges coming into it), the number of wires in the circuit is at most $2s$. Therefore $2s \geq n-k+s-1$, implying $s \geq (n-1)-k$. □

The following is analogous to the proof of Proposition 2.

Corollary 4. *Every f that is degenerate in at most k variables has depth complexity at least $\log_2(n-k)$, over the basis of all Boolean functions on two inputs.*

3.2 Sensitivity

We also use the classical concept of the sensitivity of a function.

Definition 3 ([Weg87])**.** The *sensitivity of f at $x \in \{0,1\}^n$* is the number of indices $i \in [n]$ such that $f(x^{(i)}) \neq f(x)$.

For concrete examples, note the PARITY function on n-bit inputs has sensitivity n at every point (flipping any bit changes the parity), and the AND function has sensitivity n at the all-ones input but sensitivity 0 at the all-zeroes input. A basic result about sensitivity is the following proposition:

Proposition 4 ([CDR86, Weg87])**.** *Suppose a function $f : \{0,1\}^n \to \{0,1\}$ has sensitivity at least k at some point x. Then f requires at least $k-1$ gates and $\log_2(k)$ depth, even over B_2.*

Proof. We prove the contrapositive. Suppose f has a minimal-size circuit C with at most $k-2$ gates. By the same argument as in Proposition 3, there must be at least v input nodes in C that do not have an outgoing wire, where $k-2 = (n-1)-v$. Noting that $v = n-k+1$, this means that circuit C's output only ever depends on at most $n-v = k-1$ of the inputs. Therefore f has sensitivity at most k at **every** point. The depth lower bound of at least $\log_2(k)$ follows analogously to the proof of Proposition 2. □

4 Worst-Case Lower Bounds for Modular Squaring

In this section, we present worst-case depth lower bounds for squaring n-bit integers. We study the "middle bits" of the output, show they can efficiently compute symmetric Boolean functions, and use this information to prove an overall depth lower bound. Our approach proceeds by adapting a reduction of Furst-Saxe-Sipser [FSS81], who showed how to reduce the PARITY function to MULTIPLICATION in constant depth. Recall that IS maps n-bit integers x to $2n$-bit integers x^2. We will show that low-depth circuits for IS can be used to obtain somewhat-low-depth circuits for the function $\mathsf{SUM}(x_1, \ldots, x_\ell) \mapsto \sum_i x_i$ for $\ell \geq \Omega(n/\log n)$.

Let $m \in \mathbf{Z}_{>0}$ be our modulus. If we only consider integers $x \in [0, \lfloor\sqrt{m-1}\rfloor]$, then clearly $(x^2 \bmod m) = \mathsf{IS}(x)$, and our problem reduces to proving a depth lower bound for IS on integers x which are b bits long, where $b = \lfloor \log(m-1)/2 \rfloor$. Let us record this simple observation.

Proposition 5. *Let $m \geq 2$ be an integer. If IS requires depth-d circuits on b-bit input, where $b = \lfloor \log(m-1)/2 \rfloor$, then computing $(x^2 \bmod m)$ on b-bit input requires depth-d as well.*

Now we turn to the reduction. Let $\ell \in \mathbf{Z}_{>0}$. Consider the bits $x_i \in \{0,1\}$ defined by $x = \sum_{i=0}^{\ell-1} x_i 2^i$. Define the degree-$(2\ell-1)$ polynomial

$$p(x_0, \ldots, x_{\ell-1}, y) := \sum_{i=0}^{\ell-1} y^i + \sum_{i=0}^{\ell-1} x_i \cdot y^{\ell+i}.$$

In the following, we will construe p as a univariate polynomial in y, with coefficients which are polynomials in the x_i.

Claim. *The degree-$(2\ell-1)$ coefficient of $p(x_0, \ldots, x_{\ell-1}, y)^2$ equals $2\sum_{i=0}^{\ell-1} x_i$.*

Proof. For notational simplicity, let c_m be the degree-m coefficient of $p(x_0, \ldots, x_{\ell-1}, y)$. The degree-$m$ coefficient is the sum of all products $c_i c_j$ over ordered pairs $(i,j) \in \{0,1,\ldots,k\}^2$ such that $i+j=m$. Now let $m = 2\ell-1$. WLOG, consider the case where $i < j$. Then for each j there is exactly one $i < j$ such that $i + j = 2\ell - 1$, and $i < \ell < j$. Thus $c_i c_j = x_{j-\ell}$, and the coefficient is $2\sum_{j=\ell}^{2\ell-1} x_{j-\ell} = 2\sum_i x_i$ by double-counting the pairs (i,j) and (j,i). □

Set $k := 2 + \lceil \log_2(\ell) \rceil$, and consider the polynomial

$$q(x_0, \ldots, x_{\ell-1}) := p(x_0, \ldots, x_{\ell-1}, 2^k).$$

We claim that, for all $a \in \{0,1\}^\ell$, the binary representation of $q(a)^2$ contains $\sum_i a_i$ as a substring. By Claim 4, $2\sum_i x_i$ is the coefficient of $y^{2\ell-1}$ in p^2. For $j = 0, \ldots, 2\ell-1$, note that each coefficient of y^j in $p(a,y)^2$ is at most $j+1$, since the coefficients of $p(a,y)$ (in the variable y) are in $\{0,1\}$. Thus by setting $y := 2^k > 2\ell$, each coefficient c_j of y^j in $p(a,y)^2$ is less than y, for $j = 0, \ldots, 2\ell-1$. It follows that each such coefficient c_j appears as a binary substring in the bit representation of $q(a)^2$. In particular, the sum of all terms of degree less than $2\ell-1$ in $p(a,y)^2$ contributing to $q(a)^2$ is

$$\sum_{j=0}^{2\ell-2} c_j y^j \leq \sum_{j=0}^{2\ell-2} (j+1) \cdot 2^{jk} < 2^{(2\ell-1)k}.$$

Moreover, all terms of degree greater than $2\ell-1$ in $p(a,y)^2$ appearing in $q(a)^2$ are multiplied by $y^{2\ell} = 2^{2\ell k} > 2\ell \cdot 2^{(2\ell-1)k} \geq (2\sum_i x_i) y^{2\ell-1}$. Therefore $2\sum_i a_i$ appears as a substring in the output, i.e., the substring representing $\sum_i a_i$ followed by a 0.

We have shown the output of $\mathsf{IS}(q(a))$ contains the quantity $\sum_i a_i$ in binary. Given a circuit C for integer squaring on b bits, we can "implement" $q(a)$ by simply plugging in appropriate zeroes into the inputs of C. We have proven the following.

Theorem 4. *Let $\ell \in \mathbf{Z}_{>0}$. Given a depth-d circuit for IS on integers of bitlength $b := 2\ell \cdot k = 4\ell + 2\ell \log_2(\ell)$, we can construct a depth-d circuit for $\mathsf{SUM}(a_1, \ldots, a_\ell) = \sum_i a_i$ on ℓ-bit inputs.*

Thus, lower bounds for SUM imply lower bounds for IS. We will utilize the following classical lower bound of Krapchenko [Weg87, p.258-262].

Proposition 6 (Krapchenko 1972)**.** *Every circuit for* SUM *on ℓ-bits requires depth at least $2\log_2(\ell)$.*

Proof. The least significant bit of SUM is the PARITY function on ℓ bits, for which Krapchenko showed a $2\log_2(\ell)$ depth lower bound. □

Applying Proposition 6 and Theorem 4 yields a worst-case lower bound for IS.

Theorem 5. *For every circuit with AND and OR gates (and NOT gates for free),* IS *on n-bit integers requires depth at least $2\log_2(n) - 2\log_2\log_2(n) - 2$.*

Proof. Given any depth-d circuit for IS on integers of bitlength $n := 4\ell + 2\ell\log_2(\ell)$, Theorem 4 and its corollary show that we get a depth-d circuit for SUM on ℓ-bit inputs. But the least significant bit of SUM is precisely PARITY, so we also obtain a depth-d circuit for PARITY. Observe that for all $\ell \geq 1$, we have $\ell \geq n/(2\log_2(n))$. Therefore by Proposition 6, we must have $d \geq 2\log_2(\ell) \geq 2(\log_2(n) - \log_2\log_2(n) - 1)$. □

Combining Theorem 5 and Proposition 5, we obtain the desired worst-case lower bounds for MS.

Reminder of Theorem 1. *For every n, every logic circuit over U_2 computing MS on n-bit integers requires depth at least $2\log_2(n-1) - 2\log_2(\log_2(n-1) - 1) - 4$.*

Corollary 5. *For every circuit with AND and OR gates (and NOT gates for free), MS on 2048-bit integers requires depth at least 12.*

A Generalization. The above approach gives a reduction from IS to SUM, and appeals to lower bounds on computing SUM. The best-known depth lower bound for SUM (in fact, for any symmetric function) is only $2\log n$ (see [GTTN18]) and this has been the state-of-the-art since the 1970s [Kra70].

We observe a generalization of the above reduction that strongly suggests the possibility of larger depth lower bounds. Recall that a *symmetric Boolean function* $f : \{0,1\}^n \to \{0,1\}$ has the property that its output is determined solely by the number of 1's in its input. In other words, f is symmetric if there is a "spectrum" function $g : \{0, 1, \ldots, n+1\} \to \{0,1\}$ such that for all $a \in \{0,1\}^n$, $f(a) = g(\sum_i a_i)$. Classic examples of symmetric functions include OR, AND, MAJORITY, PARITY, and MODm (which outputs 1 if and only if the sum of its inputs is divisible by m).

Theorem 6. *Suppose there is a symmetric Boolean function $f : \{0,1\}^n \to \{0,1\}$ with depth complexity at least $c\log_2(n)$, such that its spectrum function g has a depth-$d(n)$ circuit on input of length $1 + \lceil\log_2(n)\rceil$. Then,* IS *has depth complexity at least $c(\log_2(n) - \log_2\log_2(n) - 2) - d(n/(2\log n))$.*

The depth $d(n)$ of the spectrum function is typically very small, as the input to the spectrum function is only about $\log(n)$ bits. For example, with the PARITY function, the spectrum function g has a *depth-zero* circuit: we only have to output the least significant bit of the input. For the MAJORITY function, when the number of input bits is a power of two, we only have to output the most significant bit of the input. (When the number of inputs is not a power of two, we have to check that the input number is at least a fixed number, which can be done in $O(\log\log n)$ depth, with small constant in the big-O.) In general, we expect $d(n)$ to be *at most* $O(\log\log n)$ for natural symmetric functions.

Proof of Theorem 6. Given any circuit of depth D for IS on n-bit inputs, we first obtain a depth D circuit for SUM on ℓ-bit inputs, for $\ell \geq n/(2\log_2 n)$ as described above. The output of SUM is $1 + \lceil\log_2(\ell)\rceil$ bits, encoding an integer in $\{0, 1, \ldots, \ell\}$. If the spectrum function g has depth $d(\ell)$, then we can compute f from the SUM in only $d(\ell)$ additional depth. Thus we get a circuit for f on ℓ-bit inputs of depth $D + d(\ell)$. Assuming f has depth complexity at least $c\log_2(\ell)$, we have $D \geq \log_2(\ell) - d(\ell)$. □

For SUM, the lowest-depth circuits over AND/OR/NOT known still have depth larger than $4\log_2(n)$; when XOR gates are included, the depth is still greater than $3\log_2(n)$. In particular, it is known how to construct circuits for SUM with depth $4.14\log_2(n)$ (over AND/OR/NOT) and $3.02\log_2(n)$ (over AND/OR/NOT/XOR); the current best known result seems to be by Sergeev [Ser16]. Thus there is still a fairly wide gap ($2\log n$ versus $4.14\log n$) between lower bounds and upper bounds on the depth complexity of SUM, and we might expect to improve our concrete lower bound for MS by another factor of two, as follows.

Corollary 6. *If* SUM *on n bits requires depth $4\log n$ over AND/OR/NOT, then* IS *on n bits requires depth $4\log_2(n) - 4\log\log(n) - 4$. For $n = 2048$, the depth lower bound is at least 26.*

The following is immediate, using the reduction from IS to MS of Proposition 5.

Reminder of Theorem 2. *Suppose* SUM *requires depth at least $c\log_2(n)$ for a constant $c > 2$. Then MS on n-bit integers requires depth at least $c\log_2(n-1) - c\log_2(\log_2(n-1)-1) - 2c$.*

4.1 Discussion

The above results show how to lower bound the depth of integer squaring (IS) using a lower bound on SUM. One might think there could be significant "loss" in reducing IS to a simple-looking problem like SUM, but surprisingly, this is not actually the case. It is also known that the depth of multiplication can be *upper bounded* by low-depth circuits for SUM, so the two problems are in fact tightly correlated in terms of depth complexity.

Theorem 7 ([Kra70] as cited in [Ser16])**.** *Given a depth-d circuit for* SUM *on n-bit input, there is a circuit for multiplication of two n-bit numbers of depth at most $d + \log_2(n)$.*

Applying the results of [Ser16] to this theorem, the lowest-depth circuits known for multiplication (and thus squaring) have depth $4.02\log_2(n)$ over all fan-in two gates, and depth $5.14\log_2(n)$ over AND/OR/NOT gates.

The arguments of this section proving for worst-case depth lower bounds do not look directly helpful for trying to prove an *average case* lower bound against IS or modular squaring. In particular, when we reduce from SUM to IS, the resulting distribution of inputs is far from nice and uniform: starting from an n-bit input to a symmetric Boolean function, we insert many zeroes in specific places in the corresponding input to IS. We will need a different sort of argument to prove average-case lower bounds.

5 Average-Case Lower Bounds

We now turn to proving average-case lower bounds for MS, where our discussion will culminate in the proof of Theorem 3.

Reminder of Theorem 3. *Let $n \geq 38$, and let m be the product of two primes in $[2^{n/2-1}, 2^{n/2})$. Every logic circuit computing $(x^2 \bmod m)$ correctly on at least 76% of all n-bit integers x (over the basis U_2 or B_2) requires depth at least $\log_2(n/4 - 7.2)$.*

Our proof will rely critically on sensitivity arguments, as defined and described in Section 3. As a warm-up, let us begin with a short argument that the n-th output of IS (integer squaring without modular reduction) requires about $\log(n)$ depth in the *worst case*, via a sensitivity argument. Note that this bound is worse than that proved in the previous section, but it will be motivating for the average case.

Theorem 8. *There is an n-bit x such that the n-th output bit of* IS *has sensitivity at least $n/2-1$ at the point x.*

Proof. For simplicity assume n is even, and let $x := \sum_{j=n/2+1}^{n-1} 2^j$. Observe x^2 has a 0 in the n-th output of IS, because $x^2 = \sum_{j,k \geq n/2+1} 2^{j+k}$, and $j+k > n$ for all j,k.

For every $i = 0, \ldots, n/2-2$, there is exactly one $j' \in [n/2+1, n-1]$ such that $i + j' = n-1$. Therefore

$$(x^{(i)})^2 = (x+2^i)^2 = \sum_{j,k \geq n/2+1} 2^{j+k} + \sum_{j=n/2+1}^{n-1} 2^{i+j+1} + 2^{2i}$$

has a 1 in the n-th output (all $i+j+1$'s are distinct, larger than $2i$, and smaller than the $j+k$'s). □

Applying Proposition 4 directly, we get a (weak) depth lower bound for IS.

Corollary 7. *The n-th output bit of* IS *on n-bit integers requires depth at least* $\log_2(n/2-1) \geq \log_2(n) - 2$.

We would like to extend such an argument to hold for the average case, showing that any circuit that can approximately compute IS still requires an interesting level of depth. The first obvious problem is that the "bad" input x being chosen in the proof of Theorem 8 is not at all "random"! What we would like to show is that sensitivity of IS on *random* points is high. Given such a result, we could then use it to prove that all small circuits cannot *approximate* IS well in the average case.

The Framework. Let $\delta \in (0,1]$ and $\varepsilon \in (0,1/2)$ in the following.

Definition 4 ((δ,ε)-sensitivity)**.** A function $f : \{0,1\}^n \to \{0,1\}$ is (δ,ε)-sensitive if there are at least δn distinct values $i \in \{0,1,\ldots,n-1\}$ such that

$$\Pr_{x \sim \{0,1\}^n} \left[f(x) \neq f\left(x^{(i)}\right) \right] \geq 1/2 - \varepsilon.$$

For small $\varepsilon > 0$, (δ,ε)-sensitivity means that there are at least δn input indices such that, for approximately half of the possible inputs x, f is sensitive at x on all of those input indices. For our lower bounds, we will want ε to be as small (close to 0) as possible, and we want δ to be as large (close to 1) as possible.[2]

The utility of (δ,ε)-sensitivity is captured by the following theorem.

Theorem 9. *Let $\varepsilon \in (0,1/2)$ and $\delta \in (0,1]$. Assume f is (δ,ε)-sensitive. Then for all $n \geq 1$, every fan-in two circuit with at most $\delta n - 2$ gates fails to compute f on at least a $(1/4 - \varepsilon/2)$-fraction of all n-bit inputs. A similar statement hold for any (fan-in two) circuit with depth at most* $\log_2(\delta n - 1)$.

Proof. Let C be any n-input circuit with $\delta n - 2$ gates. First of all, note if $\delta n - 2 < 1$ then the theorem is trivial. Without loss of generality, C is minimal (otherwise, let us take a minimum-size circuit equivalent to C). Recall that Proposition 3 (Section 3) shows that every minimal fan-in two circuit with at most $(n-1)-k$ gates is degenerate in at least k variables. Thus our circuit C is degenerate in at least $n+1-\delta n$ variables (for example, if $\delta = 1$, then C is degenerate in at least one variable). Thus there are variable indices $i_1, \ldots, i_{n+1-\delta n} \in [n]$ such that

$$\text{For every } x \in \{0,1\}^n \text{ and } j = 1, \ldots, n+1-\delta n,\ C(x) = C(x^{(i_j)}). \tag{1}$$

[2] Please note that the (δ,ε)-sensitivity of f is not at all the same as the concept in the literature of the "average sensitivity of f", which is the *expected value* of the sensitivity of f at uniform random input.

We show that, under the hypotheses, that C fails to compute f correctly on at least a $1/4 - \varepsilon/2$ fraction of all n-bit inputs. The key observation is that, if f is (δ, ε)-sensitive, then there are at least δn distinct $i \in [n]$ such that

$$\Pr_{x \sim \{0,1\}^n}[f(x) \neq f(x^{(i)})] \geq 1/2 - \varepsilon, \tag{2}$$

but there are also at least $n+1-\delta n$ values in $[n]$ such that $C(x) = C(x^{(i_j)})$ is true for all x. By the pigeonhole principle, there must be an index $i \in [n]$ such that both $C(x) = C(x^{(i)})$ holds for all x, and (2) is true.

Fix such an i, and pair up all inputs in $\{0,1\}^n$ into the 2^{n-1} disjoint pairs $(x, x^{(i)})$. Define

$$S := \{(x, x^{(i)}) \mid f(x) \neq f(x^{(i)})\}.$$

Since f is (δ, ε)-sensitive, there are at least $(1/2 - \varepsilon) \cdot 2^n$ distinct $x \in \{0,1\}^n$ such that the pair $(x, x^{(i)})$ has $f(x) \neq f(x^{(i)})$. This implies $|S| \geq (1/2 - \varepsilon) \cdot 2^{n-1}$.

By our choice of i, we also have $C(x) = C(x^{(i)})$ for all x. Because $f(x) \neq f(x^{(i)})$ for all $(x, x^{(i)}) \in S$, it follows that C computes f correctly on exactly *one* string in each pair $(x, x^{(i)}) \in S$. Therefore there are at least $(1/2 - \varepsilon) \cdot 2^{n-1} = (1/4 - \varepsilon/2) \cdot 2^n$ inputs y such that $C(y) \neq f(y)$, so the circuit C errs on at least a $1/4 - \varepsilon/2$ fraction of all inputs.

The depth lower bound follows analogously to the proof of Proposition 2. □

Applying the Framework. We would like to apply Theorem 9 to obtain Theorem 3. This requires us to study the (δ, ε)-sensitivity of bits of modular squaring. On a number-theoretic level, it is easier to work with the situation of picking x at random from $\mathbf{Z}/m\mathbf{Z}$, but for the purposes of applying Theorem 3, we need to consider the case where x is a uniform random n-bit string (viewed as an integer in $\{0, 1, \ldots, 2^n - 1\}$). To this end, consider the experiment when we pick $x \in \{0,1\}^n$ at random, and let $x^{(i)}$ be the i-th bit flipped. More formally, if $x = (x_{n-1}, \ldots, x_0) \in \{0,1\}^n$, we can define

$$x^{(i)} := x + (1 - 2x_i)2^i.$$

Note that under this definition, $x^{(i)} = x + 2^i$ if the i-th bit of x_i is 0, and otherwise equals $x - 2^i$. We observe an easy proposition.

Proposition 7. *The distribution of outputs of* $(x \bmod m)$ *over* $x \sim \mathbf{Z}/m\mathbf{Z}$ *and the distribution of* $(x \bmod m)$ *over* $x \sim \{0, 1, \ldots, 2^n - 1\}$ *are identical, up to additive* $1/2^n$ *factors in the probabilities.*

Proof. Consider the distribution $\mathcal{D}$ which chooses a uniform random $x \in \{0, 1, \ldots, 2^n - 1\}$, and outputs $(x \bmod m) \in \{0, \ldots, m-1\}$. This distribution $\mathcal{D}$ partitions $\{0, \ldots, 2^n - 1\}$ of cardinality 2^n into m equivalence classes, where each class contains at least $2^n/m - 1$ elements and at most $2^n/m + 1$ elements. Thus for all $t \in \mathbf{Z}/m\mathbf{Z}$, we have

$$\Pr_{x \sim \{0,1,\ldots,2^n-1\}}[(x \bmod m) = t] \in [1/m - 1/2^n, 1/m + 1/2^n].$$

Hence $\left|\Pr_{x \sim \{0,1,\ldots,2^n-1\}}[(x \bmod m) = t] - \Pr_{x \sim \mathbf{Z}/m\mathbf{Z}}[(x \bmod m) = t]\right| \leq 1/2^n$. □

Therefore, the two distributions of outputs of $(x^2 \bmod m) \bmod 2$ (over $x \sim \mathbf{Z}/m\mathbf{Z}$ and $x \sim \{0, 1, \ldots, 2^n - 1\}$) are also identical up to $1/2^n$ as well, so proving (δ, ε)-sensitivity results for x chosen at random from $\mathbf{Z}/m\mathbf{Z}$ imply analogous results for x chosen at random from $\{0,1\}^n$ (construed as a non-negative integer).

The following two results are proved in Section 6 and Section 6.1. Keeping in mind Proposition 7, these results are immediate corollaries of Theorem 12 and Theorem 13, respectively.

Theorem 10 (Squaring Sensitivity Theorem 1)**.** *For all odd $2^{n-1} \leq m \leq 2^n - 1$, and all $\varepsilon > 0$, the function MS-MOD2$_{n,m}$ is $\left(1/4 + \frac{\log_2(\varepsilon)}{n} + o(1), \varepsilon\right)$-sensitive, where $o(1)$ does not depend on ε and $\lim_{n\to\infty} o(1) = 0$.*

To obtain concrete lower bounds, one should explicitly estimate the $o(1)$ term appearing in Theorem 10. When m is an RSA modulus (the case of interest in all modular squaring applications we are aware of) we obtain the following explicit result.

Theorem 11 (Squaring Sensitivity Theorem 2)**.** *For all $n \geq 38$, let p, q be primes in $[2^{n/2-1}, 2^{n/2})$. For all $\varepsilon > 0$, the function MS-MOD2$_{n,pq}$ is $\left(1/4 + \frac{\log_2(\varepsilon) - \log_2(5.8)}{n}, \varepsilon\right)$-sensitive.*

Putting the pieces together, we can complete the proof of Theorem 3.

Proof of Theorem 3. Let C be a circuit computing MS correctly on at least 76% of all n-bit integers, modulo an RSA modulus m as specified in the theorem. By Theorem 11, the least significant output bit of C is $\left(1/4 + \frac{\log_2(\varepsilon) - \log_2(5.8)}{n}, \varepsilon\right)$-sensitive. By Theorem 9, every fan-in two circuit with at most $\delta n - 2$ gates fails to compute f on at least a $(1/4 - \varepsilon/2)$-fraction of all n-bit input, where $\delta = 1/4 + \frac{\log_2(\varepsilon) - \log_2(5.8)}{n}$. Setting $\varepsilon = 1/50$, the fraction becomes 24%, and $\delta n = n/4 + \log_2(1/50) - \log_2(5.8) > n/4 - 8.2$. Therefore the circuit size lower bound is at least $n/4 - 8.2$, and the resulting depth lower bound is at least $\log_2(n/4 - 7.2)$ (by an analogous argument as the proof of Proposition 2). □

5.1 Discussion

Before we move to the proofs of the Squaring Sensitivity Theorems, it is worth noting why we had to introduce a notion of (δ, ε)-sensitivity, to prove what we did. It would be natural to conjecture (and we originally did) that for *all* indices $i \in [n]$, the low-order bit of $(x^2 \bmod m)$ differs from the low-order bit of $((x^{(i)})^2 \bmod m)$ with probability approximately $1/2$, over a uniform random choice of $x \sim \mathbf{Z}/m\mathbf{Z}$.

Conjecture 1 (A Bad Squaring Sensitivity Conjecture)**.** *For all fixed constant $\varepsilon > 0$, for all large enough n, for all odd (square-free) $2^{n-1} \leq m \leq 2^n - 1$, and for* **all** *$i \in \{0, 1, \ldots, n-1\}$,*

$$\Pr_{x \sim \mathbf{Z}/m\mathbf{Z}}[(x^2 \bmod m) \bmod 2 \neq ((x^{(i)})^2 \bmod m) \bmod 2] \geq 1/2 - \varepsilon.$$

That is, Conjecture 1 posits that the parity of $(x^2 \bmod m)$ differs from the parity of $((x^{(i)})^2 \bmod m)$ with probability very close to $1/2$, when x is chosen uniformly at random from $\mathbf{Z}/m\mathbf{Z}$. It turns out that Conjecture 1 is **false** as stated (reference omitted here to preserve anonymity) and we had to resort to weaker claims where a δ-fraction of the variables are sensitive on random inputs for some $\delta < 1$.

6 The Least Significant Bit of Modular Squaring

Let m be any odd, positive integer. Recall we identify $\mathbf{Z}/m\mathbf{Z}$ with $\{0, \ldots, m-1\}$, so that $x \bmod m$ denotes the least positive residue of x modulo m. For the rest of this section, let $f : \mathbf{Z}/m\mathbf{Z} \to \mathbf{C}^\times$ be the function induced by MS-MOD2$_{n,m}$, i.e.,

$$f(x) = \begin{cases} 0 & \text{if } x \bmod m \text{ is even,} \\ 1 & \text{if } x \bmod m \text{ is odd,} \end{cases}$$

For any integer i, and any $x \in \mathbf{Z}/m\mathbf{Z}$, let $x^{(i)}$ be the class of x mod m with the i-th bit flipped. The goal of this section to prove the following theorem, which states that for any integer $i \in [0, \log_2(m)(1/4+o(1))]$, the least significant bits of x^2 and $(x^{(i)})^2$ are essentially independent, when $x \in \mathbf{Z}/m\mathbf{Z}$ is uniformly distributed.

Theorem 12. *We have*

$$\left| \frac{\#\{x \in \mathbf{Z}/m\mathbf{Z} \mid f(x^2) \neq f((x^{(i)})^2)\}}{m} - \frac{1}{2} \right| = \frac{2^{2i}}{m^{1/2+o(1)}}.$$

Notation. For any $x \in \mathbf{R}$, let $e(x) = e^{2\pi i x}$. For any integer a, define the character χ_a of $\mathbf{Z}/m\mathbf{Z}$ as $\chi_a(x) = e\left(\frac{ax}{m}\right)$. For any integer x, let $[\![x]\!]_m = \min(x \bmod m, (-x) \bmod m)$ be the distance of x to the closest multiple of m. The function log is the natural logarithm, in base e.

We now prove Theorem 12, and in Section 6.1, we specialise it to the case of RSA moduli and obtain explicit bounds. Let $f^{(i)}(x)$ be the i-th least significant bit of x mod m. In particular, for our function f we have $f = f^{(0)}$.

These functions $f^{(i)}$ are useful to study how flipping the i-th bit affects the output of any function $g : \mathbf{Z}/m\mathbf{Z} \to \mathbf{C}$, since

$$g\left(x^{(i)}\right) = f^{(i)}(x)g(x - 2^i) + (1 - f^{(i)}(x))g(x + 2^i). \tag{3}$$

In particular, these bit functions can be used to compute the size of the set in Theorem 12.

Lemma 1. *We have* $\#\{x \in \mathbf{Z}/m\mathbf{Z} \mid f(x^2) \neq f((x^{(i)})^2)\} = 2A - 4C + \varepsilon$, *where* $|\varepsilon| \leq 2^{i+2}$,

$$A = \sum_x f(x^2), \text{ and } C = \sum_x f^{(i)}(x)f(x^2)f((x - 2^i)^2).$$

Proof. We have

$$\#\{x \in \mathbf{Z}/m\mathbf{Z} \mid f(x^2) \neq f((x^{(i)})^2)\} = \sum_x f(x^2) + \sum_x f((x^{(i)})^2) - 2\sum_x f(x^2)f((x^{(i)})^2).$$

Applying Equation (3) to the function $g : x \mapsto f(x^2)$, we have

$$f((x^{(i)})^2) = f^{(i)}(x)f((x - 2^i)^2) + f((x + 2^i)^2) - f^{(i)}(x)f((x + 2^i)^2).$$

Let

$$A = \sum_x f(x^2), \qquad B_\alpha = \sum_x f^{(i)}(x)f((x + \alpha)^2),$$
$$C_\alpha = \sum_x f^{(i)}(x)f(x^2)f((x + \alpha)^2), \quad D_\alpha = \sum_x f(x^2)f((x + \alpha)^2),$$

for $\alpha = \pm 2^i$. Then,

$$\#\{x \in \mathbf{Z}/m\mathbf{Z} \mid f(x^2) = f((x^{(i)})^2)\} = 2A - (B_{2^i} - B_{-2^i}) + 2(C_{2^i} - C_{-2^i}) - 2D_{2^i}.$$

Let us estimate $|B_{2^i} - B_{-2^i}|$. With $S = \{x \mid f^{(i)}(x + 2^{i+1}) \neq f^{(i)}(x)\}$, we have

$$\begin{aligned} B_{-2^i} &= \sum_x f^{(i)}(x)f((x - 2^i)^2) = \sum_x f^{(i)}(x + 2^{i+1})f((x + 2^i)^2) \\ &= \sum_x f^{(i)}(x)f((x + 2^i)^2) + \sum_{x \in S} (f^{(i)}(x + 2^{i+1}) - f^{(i)}(x))f((x + 2^i)^2) \\ &= B_{2^i} + \sum_{x \in S} (1 - 2f^{(i)}(x))f((x + 2^i)^2). \end{aligned}$$

Since $\#S \leq 2^{i+1}$, we deduce $|B_{2^i} - B_{-2^i}| \leq 2^{i+1}$. Now, with $T = \{x \mid f^{(i)}(x+2^i) \neq 1 - f^{(i)}(x)\}$,

$$\begin{aligned}
C_{-2^i} &= \sum_x f^{(i)}(x) f(x^2) f((x-2^i)^2) \\
&= \sum_x f^{(i)}(x+2^i) f((x+2^i)^2) f(x^2) \\
&= \sum_x (1 - f^{(i)}(x)) f((x+2^i)^2) f(x^2) + \sum_{x \in T} (2f^{(i)}(x) - 1) f((x+2^i)^2) f(x^2) \\
&= D_{2^i} - C_{2^i} + \sum_{x \in T} (2f^{(i)}(x) - 1) f((x+2^i)^2) f(x^2).
\end{aligned}$$

Since $\#T \leq 2^i$, we deduce that $|C_{2^i} + C_{-2^i} - D_{2^i}| \leq 2^i$.

$$2A + (B_{-2^i} - B_{2^i}) + 2(C_{2^i} - C_{-2^i}) - 2D_{2^i} = 2A - 4C_{-2^i} + \varepsilon,$$

where $\varepsilon = (B_{-2^i} - B_{2^i}) + 2(C_{2^i} + C_{-2^i} - D_{2^i})$, so $|\varepsilon| \leq 2^{i+2}$. □

Fourier decomposition of the bit functions. It remains to estimate the terms A and C from Lemma 1. To do so, we decompose the bit functions $f^{(i)}$ as character sums.

Proposition 8. *We have the Fourier decomposition*

$$f(x) = \frac{1}{m} \sum_{\chi \in \widehat{\mathbf{Z}/m\mathbf{Z}}} c_\chi \chi(x),$$

with $c_{\chi_0} = \frac{m-1}{2}$ *and* $c_\chi = \frac{-1}{1+\chi(1)}$ *for any* $\chi \neq \chi_0$.

Proof. We have $f(x) = \frac{1}{m} \sum_\chi c_\chi \chi(x)$, where $c_\chi = \sum_{x \in \mathbf{Z}/m\mathbf{Z}} f(x)\chi(x)$. Let $T = \{x \in \mathbf{Z}/m\mathbf{Z} \mid f(x) = 1\}$. We have $c_{\chi_0} = |T| = \frac{m-1}{2}$. Now,

$$c_\chi = \sum_{x \in T} \chi(x) = \sum_{x \in T} \chi(x+2) + \chi(1) - \chi(m) = c_\chi \chi(2) + \chi(1) - 1.$$

We deduce that $c_\chi = -(1-\chi(1))/(1-\chi(2)) = -1/(1+\chi(1))$ for any $\chi \neq \chi_0$. □

The following lemma allows to bound the Fourier coefficients c_χ, as well as the coefficients of the functions $f^{(i)}$, in Proposition 9.

Lemma 2. *For any integers* a *and* b, *we have* $\frac{1}{|1+\chi_a(b)|} \leq \frac{m}{2(m-2[\![ab]\!]_m)}$.

Proof. Observe that, for any $x \in [0,1]$, we have $1 + \cos(2\pi x) \geq 2(2x-1)^2$. So for any $x \in [0,1]$, we have $\frac{1}{|1+e(x)|} = \frac{1}{\sqrt{2+2\cos(2\pi x)}} \leq \frac{1}{2|2x-1|}$. Therefore, for any integers a and b, $\frac{1}{|1+\chi_a(b)|} \leq \frac{m}{2|2(ab \bmod m) - m|} = \frac{m}{2(m-2[\![ab]\!]_m)}$. □

Proposition 9. *For all* $i \in [n]$, *we have the Fourier decomposition*

$$f^{(i)}(x) = \frac{1}{m} \sum_{\chi \in \widehat{\mathbf{Z}/m\mathbf{Z}}} c_\chi^{(i)} \chi(x),$$

with $\left| c_{\chi_0}^{(i)} - \frac{m}{2} \right| \leq 2^{i-1}$ *and* $\left| c_{\chi_a}^{(i)} \right| \leq \frac{m2^i}{2(m-2[\![a2^i]\!]_m)}$.

Proof. Let $T^{(i)} = \{x \in \mathbf{Z}/m\mathbf{Z} \mid f^{(i)}(x) = 1\}$. First, we have $c^{(i)}_{\chi_0} = |T^{(i)}|$, and $\left||T^{(i)}| - \frac{m}{2}\right| \leq 2^{i-1}$. Now, for any character χ,

$$c^{(i)}_\chi = \sum_{x\in T^{(i)}} \chi(x) = \sum_{x\in T^{(i)}} \chi(x+2^{i+1}) + \sum_{x=2^i}^{2^{i+1}-1} \chi(x) - \sum_{x\in T^{(i)} | x+2^{i+1}\geq m} \chi(x+2^{i+1}).$$

Let $[z,z'] = [0,2^i-1] \cap \{x+2^{i+1}-m \mid x \in T^{(i)} \text{ and } x+2^{i+1} \geq m\}$. For any non-trivial χ, we have $\chi(2^{i+1}) \neq 1$ (recall that m is odd), so

$$c^{(i)}_\chi = \frac{1}{1-\chi(2^{i+1})}\left(\sum_{x=z}^{z'}(\chi(x+2^i)-\chi(x))\right) = \frac{-1}{1+\chi(2^i)}\left(\sum_{x=z}^{z'}\chi(x)\right).$$

It follows that $\left|c^{(i)}_{\chi_a}\right| \leq \frac{m2^i}{2(m-2[\![a2^i]\!]_m)}$. □

Notation 1. *In the rest of the paper, we use the notation c_χ and $c^{(i)}_\chi$ for the Fourier coefficients of f and $f^{(i)}$ given in Propositions 8 and 9 respectively.*

Gauss sums. Observe that $A = \sum_x f(x^2) = \frac{1}{m}\sum_\chi c_\chi \sum_x \chi(x^2)$, and $\sum_x \chi(x^2)$ is a Gauss sum. Generalised Gauss sums also appear in C, and play an important role in our estimates of these two quantities.

Definition 5. For any integers a, b, m, the *generalised Gauss sum* is

$$G(a,b,m) = \sum_{x\in\mathbf{Z}/m\mathbf{Z}} e\left(\frac{ax^2+bx}{m}\right).$$

Lemma 3. *We have* $|G(a,b,m)| = \begin{cases} \sqrt{(a,m)m} & \text{if } (a,m) \mid b, \\ 0 & \text{otherwise.} \end{cases}$.

Proof. This is a classical result; we include a proof for completeness. We have

$$\left|\sum_x e\left(\frac{ax^2+bx}{m}\right)\right|^2 = \sum_{x,y} e\left(\frac{ax^2+bx}{m}\right) e\left(-\frac{ay^2+by}{m}\right)$$
$$= \sum_{x,y} e\left(\frac{a(x+y)+b}{m}\right)^{x-y} = \sum_s \sum_t e\left(\frac{as+b}{m}\right)^t$$

Since $\sum_t e\left(\frac{as+b}{m}\right)^t$ is m if $as+b \equiv 0 \mod m$ and 0 otherwise. We conclude by counting the number of solutions of $as+b \equiv 0 \mod m$. □

Estimating A. Here we estimate the term A from Lemma 1.

Lemma 4. *For any odd positive integer h, $\sum_{k=0}^{h-1} \frac{1}{h-2[\![k]\!]_h} \leq \log(h) + 2.31$.*

Proof. The result is immediate if $h = 1$. For $h > 1$, we have

$$\sum_{k=0}^{h-1} \frac{1}{h-2[\![k]\!]_h} = \sum_{k=1}^{(h-1)/2} \frac{2}{2k-1} + \frac{1}{h} \leq \left(2 + \sum_{k=1}^{(h-1)/2} \frac{1}{k}\right) \leq \log(h-1) + 3 - \log(2)$$
$$\leq \log(h) + 2.31,$$

where the penultimate inequality uses $\sum_{k=1}^n \frac{1}{k} \leq \log(n) + 1$. □

Proposition 10. *We have*

$$\left|A - \frac{m+1}{2}\right| \leq \frac{\sqrt{m}}{2} \sum_{g|m, g\neq m} \frac{1}{\sqrt{g}} \left(\log(m/g) + 2.31\right).$$

Proof. For any integer a, define the character $\chi_a(x) = e(ax/m)$. We get

$$A = \sum_x f(x^2) = \frac{1}{m} \sum_a c_{\chi_a} \sum_x \chi_a(x^2) = \frac{1}{m} \sum_a c_{\chi_a} G(a, 0, m) = \frac{m+1}{2} + \frac{1}{m} \sum_{a\neq 0} \frac{G(a,0,m)}{1+\chi_a(1)}.$$

Applying Lemma 3, we get

$$\left|A - \frac{m+1}{2}\right| = \left|\frac{1}{m} \sum_{a\neq 0} \frac{G(a,0,m)}{1+\chi_a(1)}\right| \leq \frac{1}{\sqrt{m}} \sum_{a\neq 0} \frac{\sqrt{(a,m)}}{|1+\chi_a(1)|} \tag{4}$$

$$= \frac{1}{\sqrt{m}} \sum_{g|m, g\neq m} \sqrt{g} \sum_{k\in(\mathbf{Z}/(m/g)\mathbf{Z})^\times} \frac{1}{|1+\chi_{gk}(1)|}. \tag{5}$$

With $h = m/g$, and applying Lemmata 2 and 4, we have

$$\sum_{k\in(\mathbf{Z}/h\mathbf{Z})^\times} \frac{1}{|1+\chi_{gk}(1)|} \leq \sum_{k=0}^{h-1} \frac{h}{2(h - 2[\![k]\!]_h)} \leq \frac{h}{2} \left(\log(h) + 2.31\right),$$

We conclude by applying the above inequality to Equation (5). □

Estimating C. We turn to estimating the term C from Lemma 1. We have

$$C = \sum_x f^{(i)}(x) f((x-2^i)^2) f(x^2) = \frac{1}{m^3} \sum_{a,b,c} c^{(i)}_{\chi_a} c_{\chi_b} c_{\chi_c} \sum_x \chi_a(x) \chi_b((x-2^i)^2) \chi_c(x^2).$$

With $d = b + c = kg$ where $g = \gcd(d, m)$, and $f = a - b2^{i+1} = g\ell$, we get

$$C = \frac{1}{m^3} \sum_{a,b,c} c^{(i)}_{\chi_a} c_{\chi_b} c_{\chi_c} e\left(\frac{b2^{2i}}{m}\right) G(b+c, a-2b2^i, m)$$

$$= \frac{1}{m^3} \sum_{b,d,e} c^{(i)}_{\chi_{f+b2^{i+1}}} c_{\chi_b} c_{\chi_{d-b}} e\left(\frac{b2^{2i}}{m}\right) G(d, f, m) = \sum_{g|m} \sum_{b\in\mathbf{Z}/g\mathbf{Z}} E(g, b),$$

where

$$E(g,b) = \frac{1}{m^3} \sum_{k\in(\mathbf{Z}/(m/g)\mathbf{Z})^\times} \sum_{n,\ell\in\mathbf{Z}/(m/g)\mathbf{Z}} c^{(i)}_{\chi_{\ell g+b2^{i+1}}} c_{\chi_{ng+b}} c_{\chi_{kg-b}} e\left(\frac{b2^{2i}}{m}\right) G(kg, \ell g, m).$$

Applying Lemma 3, we get

$$|E(g,b)| \leq \frac{\sqrt{m}}{m^3} \sum_{k\in(\mathbf{Z}/(m/g)\mathbf{Z})^\times} \sum_{n,\ell\in\mathbf{Z}/(m/g)\mathbf{Z}} |c^{(i)}_{\chi_{\ell g+b2^{i+1}}} c_{\chi_{ng+b}} c_{\chi_{kg-b}}| \sqrt{g}$$

Fixing a divisor g of m, and a non-zero element $b \in \mathbf{Z}/g\mathbf{Z}$, the quantity $|E(g,b)|$ is at most

$$\frac{\sqrt{mg}2^i}{8} \left(\sum_{\ell=0}^{m/g-1} \frac{1}{m - 2[\![(\ell g + b2^{i+1})2^i]\!]_m}\right) \left(\sum_{n=0}^{m/g-1} \frac{1}{m - 2[\![ng+b]\!]_m}\right) \left(\sum_{k=0}^{m/g-1} \frac{1}{m - 2[\![kg-b]\!]_m}\right).$$

Lemma 5. *For any divisor g of m and any integer b, we have*

$$\sum_{k=0}^{m/g-1} \frac{1}{m - 2[\![kg+b]\!]_m} \leq \frac{1}{g - 2[\![b]\!]_g} + \frac{\log(m/g) + 2.31}{g}.$$

Proof. Without loss of generality, $b = [\![b]\!]_g \leq (g-1)/2$. There is at most one k-value such that $m - 2[\![kg-b]\!]_m < g$. It is the value $k = (m/g-1)/2$, for which we have $m - 2[\![kg-b]\!]_m = g - 2[\![b]\!]_g$. Splitting the sum around this value, we have

$$\begin{aligned}
&\sum_{k=0}^{m/g-1} \frac{1}{m - 2[\![kg+b]\!]_m} \\
&= \frac{1}{g - 2[\![b]\!]_g} + \sum_{k=0}^{(m/g-1)/2-1} \frac{1}{m - 2[\![kg+b]\!]_m} + \sum_{k=(m/g-1)/2+1}^{m/g-1} \frac{1}{m - 2[\![kg+b]\!]_m} \\
&\leq \frac{1}{g - 2[\![b]\!]_g} + \sum_{k=0}^{(m/g-1)/2-1} \frac{1}{m - 2[\![(k+1)g]\!]_m} + \sum_{k=(m/g-1)/2+1}^{m/g-1} \frac{1}{m - 2[\![kg]\!]_m} \\
&= \frac{1}{g - 2[\![b]\!]_g} + \frac{1}{g} \sum_{k=1}^{m/g-1} \frac{1}{(m/g) - 2[\![(k+1)]\!]_{m/g}} \leq \frac{1}{g - 2[\![b]\!]_g} + \frac{\log(m/g) + 2.31}{g},
\end{aligned}$$

where the last inequality comes from Lemma 4. □

Let $L(g) = \frac{\log(m/g)+2.31}{g}$ for $g \neq m$ and $L(m) = 0$.

Proposition 11. *For any $g \mid m$, we have*

$$\sum_{b=1}^{g-1} |E(g,b)| \leq \sqrt{mg} 2^{i-3} (\mathcal{L}(g) + 2.83),$$

where $\mathcal{L}(g) = L(g)\left(3\log(g) + 6.93 + 3(g-1)L(g) + (g-1)L(g)^2\right)$. If $2^{2i} < g/12$, we have

$$\sum_{b=1}^{g-1} |E(g,b)| \leq \sqrt{mg} 2^{i-3} \left(\mathcal{L}(g) + \frac{5.66 + 2^{2i+3}}{g}\right).$$

Proof. We have

$$\begin{aligned}
|E(g,b)| &\leq \frac{\sqrt{mg}2^i}{8} \left(\frac{1}{g - 2[\![b2^{2i+1}]\!]_g} + L(g)\right) \left(\frac{1}{g - 2[\![b]\!]_g} + L(g)\right)^2, \\
&\leq \frac{\sqrt{mg}2^i}{8} \left(L(g)\left(\frac{3}{g - 2[\![b]\!]_g} + 3L(g) + L(g)^2\right) + \frac{1}{g - 2[\![b2^{2i+1}]\!]_g} \frac{1}{(g - 2[\![b]\!]_g)^2}\right).
\end{aligned}$$

Now let us sum these terms for all non-zero values of $b \in \mathbf{Z}/g\mathbf{Z}$. Applying Lemma 4,

$$\sum_{b=1}^{g-1} \left(\frac{3}{g - 2[\![b]\!]_g} + 3L(g) + L(g)^2\right) \leq \left(3(\log(g) + 2.31) + 3(g-1)L(g) + (g-1)L(g)^2\right)$$

It remains to estimate

$$\sum_{b=1}^{g-1} \frac{1}{g - 2[\![b2^{2i+1}]\!]_g} \frac{1}{(g - 2[\![b]\!]_g)^2} = 2 \sum_{b=1}^{(g-1)/2} \frac{1}{g - 2[\![(2b-1)2^{2i}]\!]_g} \frac{1}{(2b-1)^2}.$$

First, we have

$$2\sum_{b=1}^{(g-1)/2}\frac{1}{g-2[\![(2b-1)2^{2i}]\!]_g}\frac{1}{(2b-1)^2}\le 2\sum_{b=1}^{(g-1)/2}\frac{1}{(2b-1)^2}=2\sum_{b=0}^{(g-1)/2-1}\frac{1}{(2b+1)^2}$$
$$\le 2\left(1+\frac{1}{4}\sum_{b=1}^{\infty}\frac{1}{b^2}\right)=2+\frac{\pi^2}{12}\le 2.83.$$

Finally, suppose that $2^{2i}<g/12$. Let κ the largest integer such that $b\le\kappa$ implies $(2b-1)2^{2i}<g/4$. On one hand,

$$\sum_{b=1}^{\kappa}\frac{1}{g-2[\![(2b-1)2^{2i}]\!]_g}\frac{1}{(2b-1)^2}\le\sum_{b=1}^{\kappa}\frac{2}{g}\frac{1}{(2b-1)^2}\le\frac{2.83}{g}.$$

On the other hand,

$$\sum_{b=\kappa+1}^{(g-1)/2}\frac{1}{g-2[\![(2b-1)2^{2i}]\!]_g}\frac{1}{(2b-1)^2}\le\sum_{b=\kappa+1}^{\infty}\frac{1}{(2b-1)^2}\le\frac{1}{4}\frac{1}{\kappa-1}\le\frac{2^{2i+2}}{g},$$

where the last two inequalities use the fact that $2^{2i}<g/12$ (in particular, $\kappa>1$). □

Corollary 8. *We have* $\sum_{g|m}\sum_{b=1}^{g-1}|E(g,b)|=m^{1/2+o(1)}2^{2i}$.

Proof. By Proposition 11, we have

$$\sum_{g|m}\sum_{b=1}^{g-1}|E(g,b)|=O\left(\left(\tilde\sigma_0(m)+\log(m)\tilde\sigma_{-1/2}(m)+\log(m)^2\tilde\sigma_{-3/2}(m)\right)m^{1/2}2^{2i}\right),$$

where $\sigma_x(m)=\sum_{d|m}d^x$ is the divisor function and $\tilde\sigma_x(m)=\sigma_x(m)-1$. The corollary follows from $\tilde\sigma_0(m)+\log(m)\tilde\sigma_{-1/2}(m)+\log(m)^2\tilde\sigma_{-3/2}(m)=m^{o(1)}$. □

It remains to deal with the terms where $b=0$.

Proposition 12. *We have* $\left|E(m,0)-\frac{m}{8}\right|\le\frac{2^i}{8}$, *and for any* $g\mid m$, $g\ne m$,

$$|E(g,0)|\le\frac{g-1}{2\sqrt{mg}}L(g)\left(\frac{m}{2}+2^{i-1}+m2^iL(g)\right).$$

Proof. First, we have $E(m,0)=\frac{c_{\chi_0}^{(i)}c_{\chi_0}^2}{m^2}$. Since $c_{\chi_0}=\frac{m-1}{2}$ and $\left|c_{\chi_0}^{(i)}-\frac{m}{2}\right|\le 2^{i-1}$, we get $\left|\frac{c_{\chi_0}^{(i)}c_{\chi_0}^2}{m^2}-\frac{m}{8}\right|\le\left|\frac{(m+2^i)}{8}-\frac{m}{8}\right|\le\frac{2^i}{8}$, and we deduce the first part of the proposition. Now suppose that $g\ne m$. We have

$$|E(g,0)|\le\frac{\sqrt{mg}}{m^3}\left(\sum_{\ell\in\mathbf{Z}/(m/g)\mathbf{Z}}|c_{\chi_{\ell g}}^{(i)}|\right)\left|\sum_{n\in\mathbf{Z}/(m/g)\mathbf{Z}}c_{\chi_{ng}}\right|\left(\sum_{k\in(\mathbf{Z}/(m/g)\mathbf{Z})^\times}|c_{\chi_{kg}}|\right).$$

Using the fact that $\mathrm{Re}(c_\chi)=-1/2$ and that $c_{\chi_{-a}}=\overline{c_{\chi_a}}$ for any a, we get

$$\sum_{n\in\mathbf{Z}/(m/g)\mathbf{Z}}c_{\chi_{ng}}=c_{\chi_0}-\sum_{n=1}^{(m/g-1)/2}(c_{\chi_a}+c_{\chi_{-a}})=\frac{m-1}{2}-\frac{m/g-1}{2}=\frac{m(g-1)}{2g}.$$

In particular, $E(1,0) = 0$. From Lemma 4, we have

$$\sum_{k=1}^{m/g-1} |c_{\chi_{kg}}| \leq m \sum_{k=1}^{m/g-1} \frac{1}{m - 2[\![kg]\!]_m} \leq \frac{m}{g} \left(\log(m/g) + 2.31\right) = mL(g).$$

Similarly, $\sum_{\ell=0}^{m/g-1} |c^{(i)}_{\chi_{\ell g}}| \leq |c^{(i)}_{\chi_0}| + m2^i L(g) \leq \frac{m}{2} + 2^{i-1} + m2^i L(g)$, and the second part of the proposition follows. □

Proof of Theorem 12. By Proposition 12, we have $\sum_{g|m, g\neq m} |E(g,0)| = m^{1/2+o(1)}2^i$. Together with Corollary 8, we obtain $\left|C - \frac{m}{8}\right| = m^{1/2+o(1)}2^{2i}$. By Proposition 10, we have $\left|A - \frac{m}{2}\right| = m^{1/2+o(1)}$. Recall from Lemma 1 that

$$\#\{x \in \mathbf{Z}/m\mathbf{Z} \mid f(x^2) \neq f((x^{(i)})^2)\} = 2A - 4C + \varepsilon,$$

where and $|\varepsilon| \leq 2^{i+2}$. Theorem 12 follows. □

6.1 Application to RSA moduli

Theorem 13. *Let $p, q \in [2^{b-1}, 2^b)$ be two prime numbers of b bits, and suppose that $b \geq 19$. Then, for any non-negative integer $i \leq b/2$,*

$$\left|\frac{\#\{x \in \mathbf{Z}/m\mathbf{Z} \mid f(x^2) \neq f((x^{(i)})^2)\}}{m} - \frac{1}{2}\right| \leq 5.8 \cdot \frac{2^i}{2^{b/2}}.$$

Proof. Recall that

$$\left|\frac{\#\{x \in \mathbf{Z}/m\mathbf{Z} \mid f(x^2) \neq f((x^{(i)})^2)\}}{m} - \frac{1}{2}\right| \leq \frac{2^{i+2}}{m} + \frac{2}{m}\left|A - \frac{m}{2}\right| + \frac{4}{m}\left|C - \frac{m}{8}\right|.$$

Let us bound each of the three terms. First, $\frac{2^{i+2}}{m} \leq \frac{2^{b/2+2}}{2^{2b-2}} \leq \frac{1}{2^{b/2}} \cdot \frac{1}{2^{15}}$. By Proposition 10, we have

$$\left|A - \frac{m}{2}\right| \leq \frac{1}{2} + \frac{\sqrt{m}}{2}\left(\log(m) + 2.31 + \frac{1}{\sqrt{p}}(\log(q) + 2.31) + \frac{1}{\sqrt{q}}(\log(p) + 2.31)\right).$$

Since $b \geq 19$, we have $\frac{1}{\sqrt{p}}(\log(q) + 2.31) \leq 0.05$. It follows that for $b \geq 19$,

$$\frac{2}{m}\left|A - \frac{m}{2}\right| \leq \frac{1}{\sqrt{m}}\left(\log(m) + 2.41\right) \leq \frac{1}{2^{b-1}}\left(2b\log(2) + 2.41\right) \leq \frac{1}{2^{b/2}} \cdot 0.08.$$

Now, we have

$$\begin{aligned}\frac{4}{m}\left|C - \frac{m}{8}\right| &\leq \frac{4}{m}\left(\left|E(m,0) - \frac{m}{8}\right| + \sum_{b=0}^{p-1}|E(p,b)| + \sum_{b=0}^{q-1}|E(q,b)| + \sum_{b=1}^{m-1}|E(m,b)|\right) \\ &\leq \frac{2^{i-1}}{m} + \frac{4}{m}\left(\sum_{b=0}^{p-1}|E(p,b)| + \sum_{b=0}^{q-1}|E(q,b)| + \sum_{b=1}^{m-1}|E(m,b)|\right).\end{aligned}$$

First, $\frac{2^{i-1}}{m} \leq \frac{1}{2^{b/2}} \cdot \frac{1}{2^{18}}$. Then, by Proposition 11, for any $g \mid m$, we have $\sum_{b=1}^{g-1}|E(g,b)| \leq \sqrt{mg}2^{i-3}(\mathcal{L}(g) + 2.83)$. Moreover, for any integer x of at least 19 bits, we have $\mathcal{L}(x) \leq 0.01$. Therefore,

$$\frac{4}{m}\left(\sum_{b=1}^{p-1}|E(p,b)| + \sum_{b=1}^{q-1}|E(q,b)|\right) \leq \frac{4}{\sqrt{m}}(\sqrt{p} + \sqrt{q})2^{i-3}2.84 \leq \frac{2^i \cdot 5.68}{2^{b/2}}.$$

We also have for $g \in \{p, q\}$ that

$$\begin{aligned}\frac{4}{m}|E(g,0)| &\leq \frac{2(g-1)}{m\sqrt{mg}}L(g)\left(\frac{m}{2} + 2^{i-1} + m2^iL(g)\right)\\ &\leq \frac{\log(m/g)+2.31}{\sqrt{mg}}\left(1+\frac{2^i}{m}+\frac{2^{i+1}(\log(m/g)+2.31)}{g}\right)\\ &\leq \frac{1}{2^{b/2}}\frac{b\log(2)+2.31}{2^{b-1.5}}\left(1+\frac{1}{2^{3b/2-2}}+\frac{4(b\log(2)+2.31)}{2^{b/2}}\right) \leq \frac{1}{2^{b/2}}\cdot 0.0001.\end{aligned}$$

Since $i \leq b/2$ and $b \geq 19$, we have $2^{2i} < m/12$, therefore

$$\begin{aligned}\frac{4}{m}\sum_{b=1}^{m-1}|E(m,b)| &\leq 2^{i-1}\left(\mathcal{L}(m)+\frac{5.66+2^{2i+3}}{m}\right)\\ &\leq 2^{i-1}\left(L(m)\left(3\log(m)+6.93+3mL(m)+mL(m)^2\right)+\frac{5.66+2^{2i+3}}{m}\right)\\ &\leq \frac{2^{i-1}}{m}\left(2.31\left(3\log(m)+6.93+3\cdot 2.31+2.31^2/m\right)+5.66\right)+\frac{2^{3i+2}}{m}\\ &\leq \frac{1}{2^{b/2}}\frac{1}{2^{b-1}}\left(2.31\left(6b\log(2)+13.86+2.31^2/2^{36}\right)+5.66\right)+\frac{1}{2^{b/2}}\frac{2^{i+4}}{2^{b/2}}\\ &\leq \frac{0.0009+2^i\cdot 0.03}{2^{b/2}}\end{aligned}$$

We deduce that $\frac{4}{m}\left|C-\frac{m}{8}\right| \leq \frac{2^{-18}+2^i\cdot 5.68+0.0002+0.0009+2^i\cdot 0.03}{2^{b/2}} \leq \frac{5.71\cdot 2^i+0.0012}{2^{b/2}}$. Finally,

$$\left|\frac{\#\{x \in \mathbf{Z}/m\mathbf{Z} \mid f(x^2) \neq f((x^{(i)})^2)\}}{m} - \frac{1}{2}\right| \leq \frac{1}{2^{b/2}}\left(\frac{1}{2^{15}}+0.08+5.71\cdot 2^i+0.0012\right),$$

and the result follows. □

7 Conclusion

We have proven concrete time-delay results for modular squaring computations. Of course, one would like to prove more: that repeated squarings should also require higher depth. As discussed in the introduction, this seems extremely difficult to prove without making some unproven conjectures. Very recently, several papers [RS20, KLX20] have made significant progress on conditional lower bound results for repeated modular squaring. Still, there are several interesting open issues to consider, which do not seem too arduous to tackle. Below, we discuss two in detail.

Extending the Average-Case Lower Bounds. Considering Theorems 1 and 3 together, they suggest that for proving stronger average-case lower bounds, it may be fruitful to try proving an average-case lower bound for the "middle bits" of modular squaring, rather than a lower bound on the least significant bit of modular squaring, as is done in this paper. Perhaps a different (more involved) case analysis than the above can yield a $2\log_2(n)$ depth lower bound for modular squaring in the average case, for all moduli m? It is known that the PARITY function on n bits requires depth $2\log(n) - O(1)$ to compute, even in the average case of uniform random input (see the discussion in [KR13]); this should be useful.

Modular Squaring in Other Representations. We have shown lower bounds for modular and integer squaring in the binary representation. In practice, one can convert a given integer into a different representation where repeated squaring is easier. Can similar lower bounds be proved for modular squaring in other number representations, such as redundant representations (as is used in some state-of-the-art modular multipliers [Özt20]) or the Chinese Remainder Representation?

Let us consider the latter, and sketch the challenges involved. In CRR, inputs are vectors $x = (x_1, \ldots, x_t)$ where each $x_i \in \mathbf{Z}/p_i\mathbf{Z}$ is written in binary for distinct co-prime p_i, $t \leq O(n/\log n)$, each p_i is $O(\log n)$ bits (so that $\prod_i p_i = \Omega(2^n)$). Letting $int(x)$ be the non-negative integer representation of x, we wish to know the depth complexity of computing $(int(x)^2 \bmod m)$ in Chinese Remainder Representation (CRR). This output would be a vector $y = (y_1, \ldots, y_t)$ where each $y_i \in \mathbf{Z}/p_i\mathbf{Z}$ is written in binary. Now, $x < \sqrt{\prod_i p_i}$ can be squared, by squaring each component $x_i \bmod p_i$ individually, which can be done in $O(\log \log n)$ depth because each p_i is only $O(\log n)$ bits. Correspondingly, the sensitivity of each output bit of integer squaring is only $O(\log n)$ for all integers in $\{0, 1, \ldots, \lfloor\sqrt{\prod_i p_i}\rfloor\}$. However, for large x, we believe the modular reduction back to $\{0, 1, \ldots, m-1\}$ should be very sensitive to the CRR. For example, if some $p_i = 2$, our CRR must track the parity of $f(x) := (int(x)^2 \bmod m)$. It is natural to conjecture that f is highly sensitive even for x written in CRR: for each component x_i of x, there should be many bits of x_i such that flipping any of them changes the parity of some outputs, just as it changes x itself. (This is in line with the results we have proven about the parity of squaring modulo m.) If we can show that a random input has many sensitive indices (say, $n/4$) with nonzero probability, we will obtain interesting average-case depth lower bounds.

7.1 Open Problem Bounties

The Ethereum Foundation is offering first-come-first-served bounties for four specific open problems. If you believe you have a solution and wish to claim the bounty, contact information can be found at https://rsa.cash, or through the authors.

1. **$5,000** Prove that for all $n \geq 128$, SUM on n-bit inputs requires depth at least $c\log_2(n)$ for some $c > 2$. (That is, improve upon Krapchenko's lower bound for SUM [Kra70].)

2. **$5,000** Prove that for all $n \geq 128$, SUM on n-bit inputs requires depth at least $4\log_2(n)$. (That is, prove the "reasonable hypothesis" stated immediately after Theorem 2. This $5,000 bounty is in addition to the $5,000 bounty above.)

3. **$2,000** Prove that there is a $c < 4$ such that for all $n \geq 128$, SUM has circuits of depth at most $c\log_2(n)$. (That is, refute the "reasonable hypothesis", and do so for all large enough input lengths n.)

4. **$3,000** Improve the average-case depth lower bound (Theorem 3) to $c\log_2(n)$ for some $c > 1$, for any algorithm computing MS-MOD2 on at least 51% of the inputs.

Acknowledgements. We are grateful to Justin Drake and Dankrad Feist for approaching us, motivating us to work on this problem, and for useful discussions and references, as well as Simon Peffers and Erdinc Ozturk. Benjamin Wesolowski was supported by the Ethereum Foundation Grants program, through Grant FY19-0151. Ryan Williams was supported by NSF CCF-1909429 "Average-Case Fine-Grained Complexity" and the Ethereum Foundation.

References

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology — CRYPTO 2018*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, 2018.

[BS07] Daniel Bernstein and Jonathan Sorenson. Modular exponentiation via the explicit chinese remainder theorem. *Mathematics of Computation*, 76(257):443–454, 2007.

[CDR86] Stephen A. Cook, Cynthia Dwork, and Rüdiger Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J. Comput.*, 15(1):87–97, 1986.

[FSS81] Merrick Furst, James Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984. See also FOCS'81.

[GTTN18] Anna Gál, Avishay Tal, and Adrian Trejo Nuñez. Cubic formula size lower bounds based on compositions with majority. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[Hua19] Hao Huang. Induced subgraphs of hypercubes and a proof of the sensitivity conjecture. *Annals of Mathematics*, 190(3):949–955, 2019.

[Juk12] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.

[KLX20] Jonathan Katz, Julian Loss, and Jiayu Xu. On the security of time-locked puzzles and timed commitments. Cryptology ePrint Archive, Report 2020/730, 2020. https://eprint.iacr.org/2020/730.

[KR13] Ilan Komargodski and Ran Raz. Average-case lower bounds for formula size. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, STOC 2013, pages 171–180, 2013.

[Kra70] VM Krapchenko. Asymptotic estimation of addition time of parallel adder. *Syst. Theory Res.*, 19:105–122, 1970.

[MOS20] Ahmet Can Mert, Erdinc Ozturk, and Erkay Savas. Low-latency asic algorithms of modular squaring of large integers for vdf evaluation. Cryptology ePrint Archive, Report 2020/480, 2020. https://eprint.iacr.org/2020/480.

[Özt20] Erdinç Öztürk. Design and implementation of a low-latency modular multiplication algorithm. *IEEE Trans. Circuits Syst.*, 67-I(6):1902–1911, 2020.

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019*, volume 124 of *LIPIcs*, pages 60:1–60:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[RS20] Lior Rotem and Gil Segev. Generically speeding-up repeated squaring is equivalent to factoring: Sharp thresholds for all generic-ring delay functions. Cryptology ePrint Archive, Report 2020/812, to appear in Advances in Cryptology — CRYPTO 2020, 2020. https://eprint.iacr.org/2020/812.

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. *Unpublished manuscript*, Revised March 10, 1996.

[Ser16] Igor Sergeevich Sergeev. Complexity and depth of formulas for symmetric boolean functions. *Moscow University Mathematics Bulletin*, 71(3):127–130, 2016.

[SM02] Larry Stockmeyer and Albert R Meyer. Cosmological lower bound on the circuit complexity of a small problem in logic. *Journal of the ACM (JACM)*, 49(6):753–784, 2002.

[vzGS00] Joachim von zur Gathen and Igor E. Shparlinski. The CREW PRAM complexity of modular inversion. *SIAM J. Comput.*, 29(6):1839–1857, 2000.

[Weg87] Ingo Wegener. *The complexity of Boolean functions*. BG Teubner. URL: https://eccc.weizmann.ac.il/resources/pdf/cobf.pdf, 1987.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology — EUROCRYPT 2019*, volume 11478 of *Lecture Notes in Computer Science*, pages 379–407. Springer, 2019.