A Tight Lower Bound on the TdScrypt Trapdoor Memory-Hard Function

Jeremiah Blocki and Seunghoon Lee

Purdue University, West Lafayette, IN, 47906, USA
{jblocki,lee2856}@purdue.edu

Abstract. A trapdoor Memory-Hard Function is a function that is memory-hard to evaluate for any party who does not have a trapdoor, but is substantially less expensive to evaluate with the trapdoor. Biryukov and Perin [BP17] introduced the first candidate trapdoor Memory-Hard Function called Diodon which modifies a Memory-Hard Function called Scrypt by replacing a hash chain with repeated squaring modulo a composite number $N = pq$. The trapdoor, which consists of the prime factors p and q, allows one to compute the function with significantly reduced cumulative memory cost (CMC) $O(n \log n \log^2 N)$ where n denotes the running time parameter, e.g., the length of the hash chain or repeated squaring chain. By contrast, the best-known algorithm to compute Diodon without the trapdoor has the CMC $O(n^2 \log N)$. Auerbach et al. [AGP24] provided the first provable lower bound on the CMC of TdScrypt — a specific instantiation of Diodon. In particular, in idealized models, they proved that the CMC of TdScrypt is $\Omega(\frac{n^2}{\log n} \log N)$ which almost matches the upper bound $O(n^2 \log N)$ but is off by a multiplicative $\log n$ factor. In this work, we show how to tighten the analysis of Auerbach et al. [AGP24] and eliminate the gap. In particular, our results imply that TdScrypt has the CMC at least $\Omega(n^2 \log N)$.

1 Introduction

A Memory-Hard Function (MHF) [Per09] is a cryptographic primitive that requires significant memory resources for evaluation. MHFs are used to design egalitarian Proofs of Work and to help protect low-entropy secrets such as user passwords against brute-force attacks in password hashing. After the initial construction of an MHF called Scrypt by Percival [Per09], there has been a line of work [FLW14, BDK16, BCS16, ABH17, BHK+19, BH22] on the construction of MHFs. Those MHF constructions are all *symmetric*, which means that any party will have the same cost to evaluate the function.

Biryukov and Perin [BP17] were the first to consider an *asymmetry* in MHFs by introducing a trapdoor, i.e., a secret piece of information that allows the function to be evaluated much more efficiently by the party who knows the secret information. The MHFs that satisfy this property were first called *asymmetrically Memory-Hard Functions* [BP17] and then later called *trapdoor Memory-Hard*

Functions (TMHFs) by Auerbach et al. [AGP24][1]. Biryukov and Perin proposed the first candidate TMHF called Diodon [BP17] which modifies Scrypt by replacing a hash chain with repeated squaring modulo an unknown large composite number N. The trapdoor is the number N and it allows one to compute the function with significantly less cost than the one without having the trapdoor (i.e., not knowing the order N).

An important cost metric to understand the memory-hardness of (T)MHFs is *cumulative memory cost (CMC)* [AS15] which refers to the total amount of memory required to evaluate the function over a sequence of operations. Biryukov and Perin [BP17] showed that the best-known algorithm to compute Diodon without the trapdoor has the CMC $O(n^2 \log N)$ where n denotes the running time parameter, i.e., the number of repeated squaring. With the trapdoor, the CMC of Diodon is significantly reduced to $O(n \log n \log^2 N)$.

Auerbach et al. [AGP24] provided the first provable lower bound of the CMC of Diodon. In particular, they considered a specific instantiation of Diodon which mostly resembles Scrypt, which they named TdScrypt.

Review: Description of TdScrypt. TdScrypt [AGP24] is a concrete instantiation of Diodon [BP17] by setting $M := n, L := n$, and $\eta := 1$, where M is a parameter to control the memory-hardness of Diodon and the parameters L and η decide its time complexity, following the same notation from Biryukov and Perin [BP17]. TdScrypt is defined over the quadratic residues $\mathsf{QR}_{N'} := \{q \in \mathbb{Z}^*_{N'} : \exists x \text{ s.t. } x^2 \equiv q \mod N'\}$ and consists of the following three algorithms.

- $\mathsf{Setup}(1^\lambda)$: the setup algorithm first samples two distinct primes p', q' with the same bit length and a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\omega_\mathsf{H}}$. Then it returns public parameters $\mathsf{pp} := (N' := p'q', \mathsf{H}, n)$ and the trapdoor $\mathsf{td} := N$ where $N := (p'-1)(q'-1)/4$.[2]
- $\mathsf{Eval}(\mathsf{pp}, W)$: the deterministic evaluation algorithm (without trapdoor) on input $W \in \mathsf{QR}_{N'}$ first computes W^{2^n} by repeated squaring, i.e., it sets $W_0 := W$ and computes $W_i := W_{i-1}^2$ for $i \in [n]$. Now, the algorithm sets $S_0 := \mathsf{H}(W_n, 0^{\omega_\mathsf{H}})$ and computes $S_i := \mathsf{H}(W_{j_i}, S_{i-1})$ for $i \in [n]$, where $j_i := S_{i-1} \mod n$. Then the algorithm returns $S_n \in \{0,1\}^{\omega_\mathsf{H}}$.
- $\mathsf{TDEval}(\mathsf{pp}, \mathsf{td}, W)$: the deterministic trapdoor evaluation algorithm on input $W =: W_0$ with the trapdoor $\mathsf{td} = N$ could compute S_n efficiently (without storing all the W_i's) using the knowledge of $N = (p'-1)(q'-1)/4 = |\mathsf{QR}_{N'}|$. In particular, it first computes $m := 2^n \mod N$ and $W_n := W_0^m$. Then it computes $S_0 := \mathsf{H}(W_n, 0^{\omega_\mathsf{H}})$. Similar to $\mathsf{Eval}(\mathsf{pp}, W)$, the algorithm then

[1] Biryukov and Lombard-Platet [BLP23] also used the word "trapdoor" to illustrate functions with the property we described, but they did not explicitly used the term "trapdoor MHFs".

[2] Note that p' and q' are sampled such that both $p := (p'-1)/2$ and $q := (q'-1)/2$ are also primes and therefore the trapdoor $N = (p'-1)(q'-1)/4 = pq$ is also the product of two distinct primes. In this case, it is known that $\mathsf{QR}_{N'} = \langle g \rangle$ is a cyclic group of order N.

computes $S_i := \mathsf{H}(W_{j_i}, S_{i-1})$ for $i \in [n]$, where $j_i := S_{i-1} \mod n$, but instead of storing all the W_{j_i}'s prior, it computes $m_i := 2^{j_i} \mod N$ and $W_{j_i} := W_0^{m_i}$ for $i \in [n]$. Finally, the algorithm returns $S_n \in \{0,1\}^{\omega_{\mathsf{H}}}$.

Review: CMC Lower Bound of TdScrypt. Auerbach et al. [AGP24] proved that without the trapdoor the CMC of evaluating TdScrypt is $\Omega(\frac{n^2}{\log n} \log N)$, which almost matches the upper bound $O(n^2 \log N)$ but off by a multiplicative factor of $\log n$. They proved the CMC lower bound in the Generic Group Model (GGM) [Sho97] and the Random Oracle Model (ROM) [BR93]. In particular, they considered the *parallel oracle model* where a polynomial-time algorithm $\mathcal{A}^{\mathcal{G},\mathsf{H}}$ has access to the generic group oracle $\mathcal{G}$ (which performs group operations) and the random oracle H. They also analyzed the CMC of TdScrypt in the *preprocessing setting*, i.e., we view an algorithm as a pair of deterministic polynomial-time algorithms $\mathcal{A}^{\mathcal{G},\mathsf{H}} = (\mathcal{A}^{\mathcal{G},\mathsf{H}}_{\mathsf{pre}}, \mathcal{A}^{\mathcal{G},\mathsf{H}}_{\mathsf{on}})$ where $\mathcal{A}^{\mathcal{G},\mathsf{H}}_{\mathsf{pre}}$ is run during the preprocessing phase and outputs a hint str and $\mathcal{A}^{\mathcal{G},\mathsf{H}}_{\mathsf{on}}$ takes the hint str as input and is run during the online phase. Then the CMC of $\mathcal{A}^{\mathcal{G},\mathsf{H}}(\mathsf{pp}, x)$ for public parameters pp and input x is defined as the CMC of $\mathcal{A}^{\mathcal{G},\mathsf{H}}_{\mathsf{on}}(\mathsf{str}, x)$.

To prove the CMC lower bound, they reduced evaluating TdScrypt to the following single-challenge game where given a hint str (generated from an input $W = g^w \in \mathsf{QR}_{N'} = \langle g \rangle$) with $\|\mathsf{str}\| = M < n$ (i.e., $\rho := \lfloor M/\omega_{\mathcal{L}} \rfloor$ group elements are encoded in str where $\tau : \mathbb{Z}_N \to \mathcal{L} := \{0,1\}^{\omega_{\mathcal{L}}}$ denotes a random injective labeling map to encode group elements to binary strings in the GGM) and the challenge $j \in [0, n)$, the goal of the game is to query W^{2^j} to H with the minimum possible number of rounds ($\leq t$ rounds). In the GGM, this can be interpreted as querying $\tau(w2^j)$ within at most t rounds. Here, since we consider the GGM, the only group operations the attacker can perform are via generic group oracle $\mathcal{G} = (\mathsf{Add}, \mathsf{Sub})$, where for $\mathfrak{a}, \mathfrak{b} \in \mathcal{L}$, $\mathsf{Add}(\mathfrak{a}, \mathfrak{b}) := \tau(\tau^{-1}(\mathfrak{a}) + \tau^{-1}(\mathfrak{b}))$ and $\mathsf{Sub}(\mathfrak{a}, \mathfrak{b}) := \tau(\tau^{-1}(\mathfrak{a}) - \tau^{-1}(\mathfrak{b}))$. As the attacker makes generic group oracle queries, we keep track of the indeterminates x_i and add a new one if needed (see [AGP24] for the formal description of the game). If we have m indeterminates at the end, we get the equation

$$a_{j,1}x_1 + \ldots + a_{j,m}x_m = w2^j \mod N.$$

The reduction considers a subset $J \subseteq [0, n)$ of challenges (with $|J| = \ell$) that are to be answered within $\leq t$ rounds, and therefore, the single-challenge game reduces to solving the system of linear equations $A\mathbf{x} = \mathbf{b}$ where $A = (a_{i,j})_{i \in [\ell], j \in [m]} \in \mathbb{Z}^{\ell \times m}$ with $|a_{i,j}| \leq 2^t$ for all $i \in [\ell]$ and vector $\mathbf{b} = (2^{j_1}w, \ldots, 2^{j_\ell}w)^\top$ with pairwise distinct $j_1, \ldots, j_\ell \in \mathbb{N}$. Here, we remark that the system of equations is defined over $\mathbb{Z}$ instead of $\mathbb{Z}_N$ since the attacker does not have the trapdoor $\mathsf{td} = N$. Then Auerbach et al. [AGP24, Lemma 2] gave a crucial Lemma that lower bounds the rank of A from the condition $|a_{i,j}| \leq 2^t$ for all $i \in [\ell]$. In particular, they proved that $\mathrm{rank}(A) \geq \ell/(3\max\{t, \log \ell\})$ which was used to show that the running time for the single-challenge game (denoted by TimeSC) is upper bounded by $n/(6\rho \log(n/2))$ with probability $> 1/2$ for a suitable pa-

rameter ρ. Combining with the incompressibility argument [DTT10, CK18], they achieved the CMC lower bound $\Omega(\frac{n^2}{\log n}\log N)$ for evaluating TdScrypt.

On the Non-Tightness of the Lower Bound. As mentioned before, the CMC lower bound $\Omega(\frac{n^2}{\log n}\log N)$ [AGP24] is not tight and off from the upper bound $O(n^2 \log N)$ by a $\log n$ factor. This non-tightness occurred due to the sub-optimal lower bound of $\mathrm{rank}(A) \geq \ell/(3\max\{t, \log \ell\})$ where the bound contains $\log \ell$ term, which leads to having $\log n$ term in the denominator of the CMC lower bound. Hence, the question is whether we can eliminate the $\log \ell$ term from [AGP24, Lemma 2], i.e., if we can prove $\mathrm{rank}(A) \geq \ell/(ct)$ for some constant $c > 0$.

A key idea to prove [AGP24, Lemma 2] was to prove the following combinatorial claim: for a subset $R \subset \mathbb{Z}^m$ with $|R| = \ell$ and every $\mathbf{r} \in R$ with $\|\mathbf{r}\|_\infty \leq 2^t$, if $m < \ell/(3\max\{t, \log \ell\})$ then there exist two distinct and disjoint subsets $R_1, R_2 \subseteq R$ such that their sums are equal, i.e., $\sum_{\mathbf{r}_1 \in R_1} \mathbf{r}_1 = \sum_{\mathbf{r}_2 \in R_2} \mathbf{r}_2$ (see [AGP24, Claim 4]). Here, they only use the infinity norm ($\|\mathbf{r}\|_\infty \leq 2^t$) to bound on rows but we observe that a stronger L_1-norm bound (i.e., $\|\mathbf{r}\|_1 \leq 2^t$) can indeed be shown (as noted in [AGP24, Appendix B]). It was an open question whether we can utilize this stronger L_1-norm bound to eliminate the $\log \ell$ term from [AGP24, Claim 4] which consequently eliminates the $\log \ell$ term from [AGP24, Lemma 2] as well.

1.1 Our Contributions

In this work, we resolve the problem raised above and prove a tighter rank bound for the system of linear equations $A\mathbf{x} = \mathbf{b}$. In particular, we prove an improved lower bound $\mathrm{rank}(A) \geq \ell/(2et)$ where e is the Euler's number (see Lemma 2). Consequently, we prove a tight CMC lower bound $\Omega(n^2 \log N)$ for evaluating TdScrypt.

A key ingredient of our result is proving a stronger version of [AGP24, Claim 4] which uses a stronger L_1-norm bound instead of the infinity norm bound.

Claim 2. *For $t, m, \ell \in \mathbb{N}$ with $\ell \geq 4$, let $R \subset \mathbb{Z}^m$ with $|R| = \ell$ and every $\mathbf{r} \in R$ satisfying $\|\mathbf{r}\|_1 \leq 2^t$. If $m < \ell/(2et)$, two subsets $R_1, R_2 \subseteq R$ exist such that at least one subset is non-empty ($R_1 \cup R_2 \neq \emptyset$), they are disjoint ($R_1 \cap R_2 = \emptyset$), and their sums are equal ($\sum_{\mathbf{r}_1 \in R_1} \mathbf{r}_1 = \sum_{\mathbf{r}_2 \in R_2} \mathbf{r}_2$).*

We stress that even though the possibility of using a stronger L_1-norm bound was already mentioned in the original paper [AGP24], proving Claim 2 is not a simple extension of prior work and requires nontrivial proof techniques. See Section 2 for the details.

Notations. Throughout the paper, e denotes the Euler's number. Unless otherwise noted, all log's have base 2, i.e., $\log x := \log_2 x$. For a positive integer N, $[N] := \{1, \ldots, N\}$ and $[0, N) := \{0, \ldots, N-1\}$. We follow standard set notations: $\mathbb{N}$ denotes the set of whole numbers, $\mathbb{Z}$ denotes the set of integers, $\mathbb{Z}_N$ denotes

the set of integers modulo N, and $\mathbb{Z}_N^* := \{x \in \mathbb{Z}_N : \gcd(x, N) = 1\}$ denotes the set of integers in $\mathbb{Z}_N$ that is coprime to N. The notation $\leftarrow\!\!\$$ denotes a uniformly random sampling, e.g., we say $x \leftarrow\!\!\$ \mathbb{Z}_N$ when x is sampled uniformly at random from $\mathbb{Z}_N$.

2 Improved Lower Bound of rank(A)

Given a system of linear equations $A\mathbf{x} = \mathbf{b}$, Auerbach et al. [AGP24] showed the lower bound of $\mathrm{rank}(A)$ if there is an upper bound for the entries of A, i.e., $A = (a_{i,j})_{i\in[\ell],j\in[m]}$ with $|a_{i,j}| \le 2^t$ for all $i \in [\ell]$ and $j \in [m]$, which is stated in Lemma 1 for completeness.

Lemma 1 ([AGP24, Lemma 2]). *Let $\ell, t \in \mathbb{N}$ with $\ell \ge 4$ and $m, w \in \mathbb{N}^+$. For any matrix $A = (a_{i,j})_{i\in[\ell],j\in[m]} \in \mathbb{Z}^{\ell\times m}$ with $|a_{i,j}| \le 2^t$ for all $i \in [\ell]$ and vector $\mathbf{b} = (2^{j_1}w, \ldots, 2^{j_\ell}w)^\top$ with pairwise distinct $j_1, \ldots, j_\ell \in \mathbb{N}$, if the system $A\mathbf{x} = \mathbf{b}$ has a solution $\mathbf{x} \in \mathbb{Z}^m$, then*

$$\mathrm{rank}(A) \ge \frac{\ell}{3\max\{t, \log \ell\}}.$$

A key idea to prove Lemma 1 was to prove the following combinatorial claim.

Claim 1 ([AGP24, Claim 4]). *For $t, m, \ell \in \mathbb{N}$ with $\ell \ge 4$, let $R \subset \mathbb{Z}^m$ with $|R| = \ell$ and every $\mathbf{r} \in R$ satisfying $\|\mathbf{r}\|_\infty \le 2^t$. If $m < \ell/(3\max\{t, \log \ell\})$, two subsets $R_1, R_2 \subseteq R$ exist such that at least one subset is non-empty ($R_1 \cup R_2 \neq \emptyset$), they are disjoint ($R_1 \cap R_2 = \emptyset$), and their sums are equal ($\sum_{\mathbf{r}_1\in R_1} \mathbf{r}_1 = \sum_{\mathbf{r}_2\in R_2} \mathbf{r}_2$).*

In this work, we improve the condition on m in Claim 1 which is stated in Claim 2, which leads to Lemma 2. While Lemma 2 has a stronger precondition than Lemma 1, we note that stronger precondition does hold i.e., Auerbach et al. [AGP24] already observed in [AGP24, Lemma 6] that the infinity norm can be improved to the 1-norm.

Lemma 2. *Let $\ell, t \in \mathbb{N}$ and $m, w \in \mathbb{N}^+$. For any matrix $A = (a_{i,j})_{i\in[\ell],j\in[m]} \in \mathbb{Z}^{\ell\times m}$ with $\sum_{j=1}^m |a_{i,j}| \le 2^t$ for all $i \in [\ell]$ and vector $\mathbf{b} = (2^{j_1}w, \ldots, 2^{j_\ell}w)^\top$ with pairwise distinct $j_1, \ldots, j_\ell \in \mathbb{N}$, if the system $A\mathbf{x} = \mathbf{b}$ has a solution $\mathbf{x} \in \mathbb{Z}^m$, then*

$$\mathrm{rank}(A) \ge \frac{\ell}{2et}.$$

The proof of Lemma 2 follows the exact same steps as the proof of Lemma 1 (see [AGP24, Lemma 2]) except that we use Claim 2 instead of Claim 1 so that "$3\max\{t, \log \ell\}$" part is updated to "$2et$". Hence, it is sufficient to prove Claim 2 to prove Lemma 2.

Claim 2. *For $t, m, \ell \in \mathbb{N}$ with $\ell \ge 4$, let $R \subset \mathbb{Z}^m$ with $|R| = \ell$ and every $\mathbf{r} \in R$ satisfying $\|\mathbf{r}\|_1 \le 2^t$. If $m < \ell/(2et)$, two subsets $R_1, R_2 \subseteq R$ exist such that at least one subset is non-empty ($R_1 \cup R_2 \neq \emptyset$), they are disjoint ($R_1 \cap R_2 = \emptyset$), and their sums are equal ($\sum_{\mathbf{r}_1\in R_1} \mathbf{r}_1 = \sum_{\mathbf{r}_2\in R_2} \mathbf{r}_2$).*

Proof. We observe that for any subset's sum $\sum_{\mathbf{r}_1 \in R_1} \mathbf{r}_1 =: \mathbf{s}_1$, we have $\|\mathbf{s}_1\|_1 \leq \sum_{\mathbf{r}_1 \in R_1} \|\mathbf{r}_1\|_1 \leq 2^t \ell$ since $\|\mathbf{r}_1\|_1 \leq 2^t$ for all $\mathbf{r}_1 \in R_1 \subseteq R$ and there are at most ℓ elements in R_1. Next, we know that there are 2^ℓ distinct (not necessarily disjoint) subsets.

Suppose that all subset's sums are distinct. Then by pigeonhole principle, it must hold that

$$2^m \binom{m + 2^t \ell}{m} \geq 2^\ell, \tag{1}$$

since there are *at most* $2^m \binom{m+2^t\ell}{m}$ possible $\mathbf{s}_1$'s that satisfies $\|\mathbf{s}_1\|_1 \leq 2^t \ell$ by Claim 3. Then by Claim 5, we observe that Equation (1) implies $m \geq \ell/(2et)$, which contradicts the condition $m < \ell/(2et)$. This contradiction occurred due to the assumption that all subset's sums are distinct. Hence, there exist two distinct, but not necessarily disjoint subsets R_1' and R_2' with the *equal subset's sum*. Applying the same trick from [AGP24, Claim 4], we can easily get disjoint sets R_1 and R_2 (by defining $R_i \coloneqq R_i' \setminus (R_1' \cap R_2')$ for $i \in [2]$). This completes the proof. □

Claim 3. *For $t, m, \ell \in \mathbb{N}$. Then there are at most $2^m \binom{m+2^t\ell}{m}$ possible $\mathbf{s} \in \mathbb{Z}^m$ that satisfies $\|\mathbf{s}\|_1 \leq 2^t \ell$.*

Proof. If we restrict our domain to $\mathbb{N}^m$, we observe that any dimension-m vector $\mathbf{s} = (s_1, \ldots, s_m) \in \mathbb{N}^m$ that satisfies $\|\mathbf{s}\|_1 \leq 2^t \ell$ can be encoded by a binary string of length $2^t \ell + m$ with Hamming weight exactly m, and therefore the number of such vectors is $\binom{m+2^t\ell}{m}$. The encoding works as follows: initialize the strong $x = \emptyset$ and define $s_{m+1} = 2^t \ell - \|\mathbf{s}\|_1 \geq 0$, Foreach $i = 1$ to m update $x \leftarrow x \circ 0^{s_i} \circ 1$. After the loop set $x = x \circ 0^{s_{m+1}}$ to ensure that the final string has length $m + 2^t \ell$. Given a binary string with Hamming weight m, we can decode the string x by setting $s_1 =$ (the number of leading 0's) and, more generally, $s_i =$ (the number of 0's between the $(i-1)^{th}$ instance of 1 and the i^{th} instance of 1) for $i \in [2, m]$.

Since we allow for positive/negative entries, the number of $\mathbf{s} \in \mathbb{Z}^m$ is at most $2^m \binom{m+2^t\ell}{m}$ since we multiply by 2^m to account for all possible sign vectors. This completes the proof.[3] □

Claim 4. *Let $t, \ell \in \mathbb{N}$ be constants. Then the function $f : \mathbb{R} \to \mathbb{R}$ defined by*

$$f(x) \coloneqq x(\log e + t + 2 + \log \ell - \log x)$$

is increasing over $x < \frac{\ell}{2et}$.

Proof. For $x < \frac{\ell}{2et}$, we observe

$$f'(x) = (\log e + t + 2 + \log \ell - \log x) + x(\log e + t + 2 + \log \ell - \log x)'$$

[3] One can also prove the claim with an alternate interpretation of the problem and prove it using Pascal's triangle or by induction. See Appendix A for alternate proofs.

$$
\begin{aligned}
&= (\log e + t + 2 + \log \ell - \log x) + x \cdot \left(-\frac{1}{x \ln 2}\right) \\
&= \log e + t + 2 + \log \ell - \frac{1}{\ln 2} - \log x \\
&> \log e + t + 2 + \log \ell - \frac{1}{\ln 2} - \log\left(\frac{\ell}{2et}\right) \\
&= 2\log e + t + \log t + 3 - \frac{1}{\ln 2} \\
&> 2\log e + t + \log t > 0,
\end{aligned}
$$

which completes the proof. □

Claim 5. *If* $m < \dfrac{\ell}{2et}$ *then for* $t \geq 2$, $2^m \dbinom{m + 2^t\ell}{m} < 2^\ell$.

Proof. We first recall that for any $n \in \mathbb{N}$ and $k \in [n]$ the following holds:

$$
\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{en}{k}\right)^k. \tag{2}
$$

By Claim 4, we have that the function $f(m) = m(\log e + t + 2 + \log \ell - \log m)$ is increasing over $m < \frac{\ell}{2et}$. Hence, if $m < \frac{\ell}{2et}$, we have

$$
\begin{aligned}
m(\log e + t + 2 + \log \ell - \log m) &= f(m) \\
&< f\left(\frac{\ell}{2et}\right) \\
&= \frac{\ell}{2et}\left(\log e + t + 2 + \log \ell - \log\left(\frac{\ell}{2et}\right)\right) \\
&= \frac{\ell}{2et}\left(2\log e + t + \log t + 3\right) \\
&< \ell,
\end{aligned} \tag{3}
$$

for $t \geq 2$. Hence,

$$
\begin{aligned}
2^m \binom{m + 2^t\ell}{m} &\leq 2^m \left(\frac{e(m + 2^t\ell)}{m}\right)^m && \triangleleft \text{ by Equation (2)} \\
&< 2^m \left(\frac{e2^{t+1}\ell}{m}\right)^m && \triangleleft \text{ since } m < \frac{\ell}{2et} < 2^t\ell \\
&= 2^{m(\log e + t + 2 + \log \ell - \log m)} \\
&< 2^\ell, && \triangleleft \text{ by Equation (3)}
\end{aligned}
$$

which completes the proof. □

3 Improved CMC Lower Bound of TdScrypt

Now, we can improve [AGP24, Lemma 1] to Lemma 3. Here, $\mathsf{params}(\lambda)$ denotes the set of all possible parameters of the form (N, τ, H, w) where N is the group order, τ denotes the labeling map, H is a random oracle, and w is a discrete log of $W \in \mathsf{QR}_{N'}$.

Lemma 3 (Single-Challenge Tradeoff). *For every pair of deterministic parallel oracle machines $\mathcal{A} = (\mathcal{A}_{\mathsf{pre}}, \mathcal{A}_{\mathsf{on}})$, all $c \in \mathbb{N}$ with security parameter λ large enough, and all $n, M, Q \in \mathsf{poly}(\lambda)$ with $n \geq 8$ and subsets $\mathsf{bad} \subseteq \mathsf{params}(\lambda)$, if, for every $(N, \tau, \mathsf{H}, w) \in \mathsf{bad}$, $\mathcal{A}_{\mathsf{on}}$ makes at most Q queries, $\mathsf{str} \coloneqq \mathcal{A}_{\mathsf{pre}}^{\tau,\mathsf{H}}(\tau(1), \tau(w))$ is of size $|\mathsf{str}| \leq M$ and*

$$\Pr_{j \leftarrow \$[0,n)}\left[\mathsf{TimeSC}_{\mathcal{A}_{\mathsf{on}}}^{\tau,\mathsf{H}}(\mathsf{str}, j) \leq \frac{n}{4e\rho}\right] \geq \frac{1}{2},$$

where $\rho = (M + \log n + \log Q + c\log\lambda + 1)/(\log N - 3(\log Q - \log n) - 3)$, then $|\mathsf{bad}| < \lambda^{-c}|\mathsf{params}(\lambda)|$.

We remark that the improvement here is removing the $1/\log n$ factor from [AGP24, Lemma 1]. The proof of Lemma 3 is largely the same as the proof of [AGP24, Lemma 1], except that the $n/(6\rho\log(n/2))$ part is updated to $n/(4e\rho)$, which simplifies the case analysis. In particular, [AGP24, Claim 2] can be updated to the following:

Claim 6. *If Lemma 3 does not hold and $\rho \leq n/(4e)$, for all $(N, \tau, \mathsf{H}, w) \in \mathsf{bad}$, $\mathrm{rank}(A) \geq \rho$.*

Proof. By the assumption that Lemma 3 does not hold, $\mathcal{A}_{\mathsf{on}}$ answers at least $\ell \coloneqq n/2 \geq 4$ challenges within $t \coloneqq n/(4e\rho)$ rounds of queries. By [AGP24, Claim 1], we may assume that bad only contains parameters where the system $A\mathbf{x} = \mathbf{b}$ has a solution. Hence, we can apply Lemma 2 to get

$$\begin{aligned}\mathrm{rank}(A) &\geq \frac{n}{4et} \\ &= \frac{n}{4e} \cdot \frac{4e\rho}{n} \\ &= \rho,\end{aligned}$$

which completes the proof. □

Proof of Lemma 3. We consider the following two cases.

Case $\rho > n/(4e)$: Then $n/(4e\rho) < 1$ and the tradeoff holds for any $j \in [0, n)$ since $\mathsf{TimeSC} \geq 1$ by [AGP24, Definition 6].

Case $\rho \leq n/(4e)$: Towards contradiction, suppose that Lemma 3 does not hold. Then by Claim 6, we have $\mathrm{rank}(A) \geq \rho$. However, [AGP24, Claim 3] tells us that if Lemma 3 does not hold then there exists a tuple $(N, \tau, \mathsf{H}, w) \in \mathsf{bad}$ such that $\mathrm{rank}(A) < \rho$ (the same proof carries over when we update the number of rounds $t = \frac{n}{6\rho\log(n/2)} - 1$ in the proof of [AGP24, Claim 5] to $t = \frac{n}{4e\rho} - 1$), which contradicts each other. Hence, Lemma 3 holds. □

Applying the improvements that we achieved, [AGP24, Theorem 1] can be improved as follows. Here, $\mathsf{cc}_{\mathsf{mem}}$ denotes the cumulative memory cost.

Theorem 1. *Let $n \in \mathsf{poly}(\lambda)$ with $n \geq 8$ and let $\mathcal{A}$ be a deterministic parallel oracle machine that evaluates* TdScrypt *correctly with probability $\chi(\lambda)$ over the choice of the parameters and input W. Then, assuming that the factoring is hard, in the GGM and ROM with probability at least $\chi(\lambda) - \epsilon(\lambda)$,*

$$\mathsf{cc}_{\mathsf{mem}}\left(\mathcal{A}^{\tau,\mathsf{H}}(\tau(1),\tau(w))\right) \in \Omega\left(n^2 \log N\right),$$

where $\epsilon(\lambda) \in \mathsf{negl}(\lambda)$, and the probability is taken over $(N, \tau, \mathsf{H}, w) \leftarrow\!\!\$\ \mathsf{params}(\lambda)$.

References

ABH17. Joël Alwen, Jeremiah Blocki, and Ben Harsha. Practical graphs for optimal side-channel resistant memory-hard functions. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1001–1017. ACM Press, October / November 2017.

AGP24. Benedikt Auerbach, Christoph U. Günther, and Krzysztof Pietrzak. Trapdoor memory-hard functions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part III*, volume 14653 of *LNCS*, pages 315–344. Springer, Cham, May 2024.

AS15. Joël Alwen and Vladimir Serbinenko. High parallel complexity graphs and memory-hard functions. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 595–603. ACM Press, June 2015.

BCS16. Dan Boneh, Henry Corrigan-Gibbs, and Stuart E. Schechter. Balloon hashing: A memory-hard function providing provable protection against sequential attacks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 220–248. Springer, Berlin, Heidelberg, December 2016.

BDK16. Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. Argon2: New generation of memory-hard functions for password hashing and other applications. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 292–302, 2016.

BH22. Jeremiah Blocki and Blake Holman. Sustained space and cumulative complexity trade-offs for data-dependent memory-hard functions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part III*, volume 13509 of *LNCS*, pages 222–251. Springer, Cham, August 2022.

BHK+19. Jeremiah Blocki, Benjamin Harsha, Siteng Kang, Seunghoon Lee, Lu Xing, and Samson Zhou. Data-independent memory hard functions: New attacks and stronger constructions. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 573–607. Springer, Cham, August 2019.

BLP23. Alex Biryukov and Marius Lombard-Platet. PURED: A unified framework for resource-hard functions. In Anupam Chattopadhyay, Shivam Bhasin, Stjepan Picek, and Chester Rebeiro, editors, *INDOCRYPT 2023, Part II*, volume 14460 of *LNCS*, pages 126–149. Springer, Cham, December 2023.

BP17. Alex Biryukov and Léo Perrin. Symmetrically and asymmetrically hard cryptography. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 417–445. Springer, Cham, December 2017.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

CK18. Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 415–447. Springer, Cham, April / May 2018.

DTT10. Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and PRGs. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 649–665. Springer, Berlin, Heidelberg, August 2010.

FLW14. Christian Forler, Stefan Lucks, and Jakob Wenzel. Memory-demanding password scrambling. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 289–305. Springer, Berlin, Heidelberg, December 2014.

Per09. Colin Percival. Stronger key derivation via sequential memory-hard functions. 01 2009.

Sho97. Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Berlin, Heidelberg, May 1997.

A Alternate Proofs of Claim 3

Reminder of Claim 3. *For $t, m, \ell \in \mathbb{N}$. Then there are at most $2^m\binom{m+2^t\ell}{m}$ possible $\mathbf{s} \in \mathbb{Z}^m$ that satisfies $\|\mathbf{s}\|_1 \leq 2^t\ell$.*

Proof using Pascal's triangle. We want to find the number of tuples $(s_1, \ldots, s_m)$ that satisfies $\sum_{i=1}^m |s_i| \leq 2^t\ell$. This can be reformulated as a problem of placing $m-1$ dividers among k units (i.e., 1's) for integer $k \in [0, 2^t\ell]$, where each partition represents the sum assigned to $|s_i|$ for each $i \in [m]$. We observe that there are $\binom{m-1+k}{m-1}$ ways to allocate $m-1$ dividers, and since s_i can be either positive or negative to have the same value $|s_i|$ for each $i \in [m]$, the total number of valid tuples $(s_1, \ldots, s_m)$ is $2^m\binom{m-1+k}{m-1}$ for each $k \in [0, 2^t\ell]$. Hence, the total number of tuples $(s_1, \ldots, s_m) \in \mathbb{Z}^m$ that satisfies $\sum_{i=1}^m |s_i| \leq 2^t\ell$ is at most $\sum_{k=0}^{2^t\ell} 2^m\binom{m-1+k}{m-1} = 2^m\binom{m+2^t\ell}{m}$ using Pascal's triangle. In particular, we have

$$\begin{aligned}\sum_{k=0}^{2^t\ell} 2^m\binom{m-1+k}{m-1} &= \sum_{k=0}^{2^t\ell} 2^m\binom{m-1+k}{k} \\ &= 2^m\left[\binom{m-1}{0} + \binom{m}{1} + \binom{m+1}{2} + \ldots + \binom{m-1+2^t\ell}{2^t\ell}\right]\end{aligned}$$

$$= 2^m \left[\underbrace{\binom{m}{0} + \binom{m}{1}}_{\binom{m+1}{1}} + \binom{m+1}{2} + \ldots + \binom{m-1+2^t\ell}{2^t\ell} \right]$$

$$= 2^m \left[\underbrace{\binom{m+1}{1} + \binom{m+1}{2}}_{=\binom{m+2}{2}} + \ldots + \binom{m-1+2^t\ell}{2^t\ell} \right]$$

$$= \ldots = 2^m \left[\binom{m-1+2^t\ell}{2^t\ell - 1} + \binom{m-1+2^t\ell}{2^t\ell} \right]$$

$$= 2^m \binom{m+2^t\ell}{2^t\ell} = 2^m \binom{m+2^t\ell}{m}. \qquad \square$$

Proof using Induction. From the previous proof, we observe that the total number of tuples $(s_1, \ldots, s_m) \in \mathbb{Z}^m$ that satisfies $\sum_{i=1}^m |s_i| \le 2^t\ell$ is at most $\sum_{k=0}^{2^t\ell} 2^m \binom{m-1+k}{m-1}$, and it remains to show that $\sum_{k=0}^{2^t\ell} 2^m \binom{m-1+k}{m-1} = 2^m \binom{m+2^t\ell}{m}$, which is immediately followed by Claim 7 (using induction). $\square$

Claim 7. *For any nonnegative integer q, $\sum_{k=0}^{q} \binom{m-1+k}{m-1} = \binom{m+q}{m}$.*

Proof. We can prove this by induction.

- (Base Case) If $q = 0$ then $\binom{m-1}{m-1} = 1 = \binom{m}{m}$.
- (Inductive Hypothesis) Suppose that $\sum_{k=0}^{q} \binom{m-1+k}{m-1} = \binom{m+q}{m}$ for some integer $q \ge 0$.
- (Inductive Step) Then we have

$$\sum_{k=0}^{q+1} \binom{m-1+k}{m-1} = \sum_{k=0}^{q} \binom{m-1+k}{m-1} + \binom{m+q}{m-1}$$
$$= \binom{m+q}{m} + \binom{m+q}{m-1}$$
$$= \binom{m+q+1}{m},$$

which completes the proof. $\square$