Cryptanalysis of Lattice-Based Sequentiality Assumptions and Proofs of Sequential Work

Chris Peikert* Yi Tang†

June 7, 2024

Abstract

This work *completely breaks* the sequentiality assumption (and broad generalizations thereof) underlying the candidate lattice-based proof of sequential work (PoSW) recently proposed by Lai and Malavolta at CRYPTO 2023. In addition, it breaks an essentially identical variant of the PoSW, which differs from the original in only an arbitrary choice that is immaterial to the design and security proof (under the falsified assumption). This suggests that whatever security the original PoSW may have is fragile, and further motivates the search for a construction based on a sound lattice-based assumption.

Specifically, for sequentiality parameter T and SIS parameters $n, q, m = n \log q$, the attack on the sequentiality assumption finds a solution of quasipolynomial norm $m^{\lceil \log T \rceil}$ (or norm $O(\sqrt{m})^{\lceil \log T \rceil}$ with high probability) in only *logarithmic* $\tilde{O}_{n,q}(\log T)$ depth; this strongly falsifies the assumption that finding such a solution requires depth *linear* in T. (The $\tilde{O}$ notation hides polylogarithmic factors in the variables appearing in its subscript.) Alternatively, the attack finds a solution of polynomial norm $m^{1/\varepsilon}$ in depth $\tilde{O}_{n,q}(T^{\varepsilon})$, for any constant $\varepsilon > 0$. Similarly, the attack on the (slightly modified) PoSW constructs a valid proof in *polylogarithmic* $\tilde{O}_{n,q}(\log^2 T)$ depth, thus strongly falsifying the expectation that doing so requires linear sequential work.

1 Introduction

The notion of *timed* (or *timed-release*) cryptography was formally introduced and realized in 1996 by Rivest, Shamir, and Wagner [RSW96], following initial concepts due to May [May93] and related ideas of Cai, Lipton, Sedgewick, and Yao [CLSY93]. The general thrust of this area is to devise "puzzles" that require (roughly) a prespecified amount of time to solve—even for solvers that have a large amount of computing power. More precisely, solving the puzzle should be *inherently sequential* (i.e., high computation depth) in nature, so that using *many processors in parallel* does not lead to any major speedup in finding a solution, versus using just one processor. (Of course, using a *faster* sequential processor will unavoidably result in a speedup, but the range of available processor speeds is substantially narrower than the ability to purchase huge numbers of parallel processors.)

Several variations on this theme have emerged in the literature. The focus of this work is on one of the most basic timed primitives, called a *proof of sequential work* (PoSW) [MMV13, CP18, AKK+19]. Here the goal is simply to quickly convince a skeptical verifier that a sequential computation of some significant desired length has been performed. In other words, the computation should be inherently sequential and

*University of Michigan, `cpeikert@umich.edu`.
†University of Michigan, `yit@umich.edu`.

tunable, but the result of such a computation should be publicly and very quickly verifiable. (No secret message is encrypted or decrypted, however.) Applications of PoSW include anti-spam and denial-of-service measures [DN92], and reducing the wasteful energy consumption of proof-of-work blockchains like Bitcoin (because using large-scale parallel computation is of little marginal benefit).

1.1 (In)Security in a Quantum World

Unfortunately, most prior timed-cryptography constructions will become *completely insecure* in the presence of general-purpose *quantum computers*. This is because the security of these constructions relies on the conjectured hardness of problems that are in fact *easy* for quantum computers. As some of the most notable examples, the constructions of [RSW96, Pie19, Wes19] rely on the presumed hardness of factoring integers, but the breakthrough work of Shor [Sho94] gave an efficient quantum algorithm for this problem. Therefore, quantum computers would be able to completely circumvent the (conjectured) classical sequentiality of these puzzles, and solve them in relatively low quantum depth.

In this light, it is important to find constructions of timed cryptography that are quantum-secure, or "post quantum"—i.e., that can be run on today's computers, but are believed to remain secure against attacks by future quantum computers. In other parts of cryptography, the most promising post-quantum systems are based on *lattice* problems, particularly *short integer solution* (SIS) [Ajt96] and *learning with errors* (LWE) [Reg05], and their variants. Over the past two decades, countless efficient and powerful cryptographic concepts have been realized from these lattice foundations.

Yet despite so much progress in general, post-quantum *timed* cryptography is still in its infancy, with very few and limited constructions. In particular, for proofs of sequential work we know of only two types of post-quantum constructions: ones in the idealized random-oracle model (or under a closely related sequential-hashing assumption) [MMV13, CP18], and a very interesting "algebraic" proposal by Lai and Malavolta [LM23] from CRYPTO 2023 that is related to lattices, and does not require random oracles.

As a foundation for their PoSW candidate, Lai and Malavolta introduced a new SIS-related problem and sequentiality assumption, for which they gave some credible evidence. Essentially, the problem is to evaluate a long chain of iterated SIS hash functions, and the assumption is that doing so requires computation depth roughly proportional to the length of the chain. Their elegant PoSW protocol works analogously to the factoring-based construction of [Pie19] by exploiting the homomorphic properties of the SIS hash function, and they proved its security under the new sequentiality assumption.

1.2 Contributions

Our first main contribution is to *strongly falsify* the lattice-based sequentiality assumption proposed in [LM23], and broad generalizations thereof. In a bit more detail, the conjecture is that finding a "somewhat short" solution to a certain regular linear system (corresponding to iterated hash evaluations) requires *nearly linear depth* in the sequentiality parameter T; see Section 2.3 for details. For typical parameters, we instead solve this problem in depth only *polylogarithmic* in T. Also, other parameterizations of our attack find asymptotically *much shorter* solutions in *small polynomial depth* T^{ε}, for any constant $\varepsilon > 0$. So, tightening the quantitative definition of "short" offers limited hope for salvaging the assumption, unless the norm bound is made quite small; see Section 1.3 below for a discussion.

Interestingly, while our attack breaks the *assumption* underlying the security proof for the PoSW protocol of [LM23], it does not break the *PoSW itself* as originally defined—and so far we have not found an attack that does so. However, we do manage to break *two slight variants* of the PoSW from [LM23], by employing our core techniques in more sophisticated ways (see Figure 4 for an illustration). These variants are supported by essentially identical security proofs (under the same kind of falsified assumption) as the original PoSW. Indeed, one of the variants differs from the original in *only an arbitrary choice* in the core "folding" operation, so we consider it to be effectively identical to the original.

Our specific contributions are organized as follows.

- In Section 3 we give a suite of very general and modular tools for efficiently computing, combining, and using lattice "trapdoors" in low computation depth. As we showcase in the rest of the paper, these tools can be combined in various ways to yield attacks on lattice-based timed cryptography proposals, and may also be of independent interest for other applications.
- In Section 4 we use our tools to give a low-depth recursive attack that strongly falsifies the sequentiality assumption from [LM23].
- Finally, in Section 5 we extend the attack to break two slight variants of the PoSW protocol from [LM23].

We refer to each individual section for further background and context for the results and techniques given therein.

1.3 Discussion and Future Work

As noted above, while we have not managed to break the PoSW from [LM23] exactly as it is written, we did break a variant that differs only in one minor and arbitrary choice, which has no effect on its underlying assumption or security proof. This state of affairs suggests that whatever security the original PoSW may have is quite fragile, and not due to any intentional design choice or technique in the security proof. Additional ideas might lead to a successful attack against the original PoSW; alternatively, it might actually be secure, perhaps with a different security proof under some other plausible assumption. We leave these topics for future work.

The effectiveness of our attacks hinges on the following key feature of the assumption and PoSW protocols following [LM23]: solutions are allowed to be "somewhat short," even though the "honest" computation generates a "very short" solution. More specifically, the norm (in ℓ_∞, say) of a solution is allowed to be *quasipolynomial* in the sequentiality parameter T, whereas the honestly computed solution has *constant* ℓ_∞ norm. The reason for this gap is that in the protocol, the honest solution is repeatedly "folded" into one having smaller dimension but larger norm, so the verifier needs to use more permissive norm checks. (Additionally, the knowledge extractor in the security proof incurs another quasipolynomial blowup in the norm of its extracted solution, relative to the norm bounds used by the verifier.)

Our attacks crucially exploit this gap by computing a "somewhat short" solution of quasipolynomial norm in only polylogarithmic depth $\operatorname{polylog}(T)$, or even a "short" one of some polynomial norm in small polynomial depth T^ε, for any constant $\varepsilon > 0$. However, we do not see how to compute a "very short" solution having constant ℓ_∞ norm (like the honestly computed one) in depth sublinear in T. So, a much weaker version of the sequentiality assumption from [LM23] corresponding to these parameters, still seems plausible. Constructing a proof system that quickly proves knowledge of such a short solution is an interesting and worthwhile open problem.

2 Preliminaries

2.1 Vector and Matrix Norms

For a real vector $\mathbf{x}$ and $p \geq 1$, define its ℓ_p norm as $\|\mathbf{x}\|_p := \left(\sum_i |x_i|^p\right)^{1/p}$, and its ℓ_∞ norm as $\|\mathbf{x}\|_\infty := \max_i |x_i|$. Observe that the Euclidean norm is simply the ℓ_2 norm. For any n-dimensional vector $\mathbf{x}$ and any

$1 \le p \le r \le \infty$, a standard bound is

$$\|\mathbf{x}\|_r \le \|\mathbf{x}\|_p \le \|\mathbf{x}\|_r \cdot n^{1/p-1/r} \ ,$$

where we adopt the convention that $a^{1/\infty} = 1$ for any $a > 0$. We extend any ℓ_p norm $\|\cdot\|_p$ on vectors to matrices $\mathbf{X}$ by taking the maximum over its columns $\mathbf{x}_j$, i.e., $\|\mathbf{X}\|_p := \max_j \|\mathbf{x}_j\|_p$.

For matrices it will be convenient to use the *operator* norm $\|\mathbf{X}\|_{r \leftarrow p} := \max_{\mathbf{y} \neq \mathbf{0}} \|\mathbf{X}\mathbf{y}\|_r / \|\mathbf{y}\|_p$ for $p, r \in [1, \infty]$. In words, the operator norm bounds the factor by which left-multiplication by $\mathbf{X}$ can expand norms, going from ℓ_p to ℓ_r:

$$\|\mathbf{X} \cdot \mathbf{Y}\|_r \le \|\mathbf{X}\|_{r \leftarrow p} \cdot \|\mathbf{Y}\|_p \ . \tag{2.1}$$

It is immediate that $\|\mathbf{X}\mathbf{Y}\|_{r \leftarrow p} \le \|\mathbf{X}\|_{r \leftarrow q} \cdot \|\mathbf{Y}\|_{q \leftarrow p}$. For any $p, r \in [1, \infty]$, observe that the identity matrix $\mathbf{I}_n$ satisfies $\|\mathbf{I}_n\|_{r \leftarrow p} = \max(n^{1/r-1/p}, 1)$, and in particular $\|\mathbf{I}_n\|_{p \leftarrow p} = 1$; and for any $\mathbf{X} \in \mathbb{R}^{n \times m}$,

$$\|\mathbf{X}\|_{r \leftarrow p} \le n^{1/r} \cdot \|\mathbf{X}\|_{\infty \leftarrow p} \le n^{1/r} \cdot \|\mathbf{X}\|_\infty \cdot m^{1-1/p} \ . \tag{2.2}$$

Random matrices. For tighter bounds, it is convenient in some cases to rely on standard results from random matrix theory; see, e.g., [Ver12]. For example, if $\mathbf{X} \in \mathbb{R}^{m \times m}$ is a random matrix with independent columns drawn from *subgaussian* distributions (which may vary from column to column), then it will satisfy $\|\mathbf{X}\|_{2 \leftarrow 2} = O(\sqrt{m})$ except with probability $2^{-\Omega(m)}$. (We refer to [Ver12] for the definition of subgaussian, which we do not need in this work.) For simplicity, we simply say "high probability" to represent $1 - 2^{-\Omega(m)}$. Observe that by the union bound, any $T = 2^{o(m)}$ events that *individually* occur with high probability will also *all* occur with high probability; we often implicitly use this fact in our high-probability statements.

Block-wise norms. For some of our purposes it will be important to have finer-grained bounds on the (operator) norms of the *row blocks* of a vector or matrix. Let $\|\cdot\|$ be a norm on vectors or matrices, such as the ones defined above. When $\mathbf{X}$ is seen as being made up of row blocks $\mathbf{X}_i$ (whose definition will always be clear from context), we take $\|\mathbf{X}\|$ to be the column *vector* of norms $\|\mathbf{X}_i\|$, i.e., $\|\mathbf{X}\| := (\|\mathbf{X}_i\|)_i$. For vectors of norms having the same dimension, we use the partial ordering given by $\mathbf{a} \le \mathbf{b}$ if $a_i \le b_i$ for all i. So, for any matrices $\mathbf{X}, \mathbf{Y}$ where now $\mathbf{X}$ is seen as being made up of row blocks (but $\mathbf{Y}$ is not), Equation (2.1) still holds, but now both sides are vectors (and $\|\mathbf{Y}\|_p$ is a scalar).

2.2 Computational Model

For simplicity, when describing and analyzing our attacks we mainly use the following abstract arithmetic-circuit model, with a focus on circuit depth. Each wire has "type" either $\mathbb{Z}$ or $\mathbb{Z}_q$ (for some fixed integer q), which is the set of its possible values.[1] The set of available gates is: addition with arbitrary fan-in, where all input and output wires have the same type; multiplication with fan-in two, where the output wire has type $\mathbb{Z}_q$ if either input does, and type $\mathbb{Z}$ otherwise; and "(bit) decomposition," which maps from $\mathbb{Z}_q$ to $\mathbb{Z}^\ell$ for $\ell = \lceil \log_2 q \rceil$ (see Section 2.3 below for further details). In particular, multiplication of matrices of any size (each over $\mathbb{Z}$ or $\mathbb{Z}_q$) can be implemented by a polynomial-size, depth-two arithmetic circuit, by computing each row-column inner product in parallel.

As is typical, to deal with varying input and output sizes we consider *families* of circuits, which are parameterized by the input size(s) and possibly some other values, which will always be clear from context.

[1] In our instantiations, the set of possible values for any $\mathbb{Z}$-wire will be a subset of $\mathbb{Z} \cap [-q/2, q/2)$, so we can alternatively consider having just $\mathbb{Z}_q$-wires.

Every circuit family in this work is implicitly *logspace uniform*, i.e., there is a logspace Turing machine that outputs the circuit description given its parameters (in unary). In particular, the circuits can be computed in polylogarithmic depth, and their size is polynomial. (Logspace uniformity is widely seen as the most appropriate one for complexity classes defined by polylogarithmic depth.)

Any circuit family in our arithmetic model can be compiled (in uniform logspace) to, e.g., the standard model of Boolean circuits with bounded fan-in, with multiplicative depth overhead that is logarithmic in the maximum fan-in, and polylogarithmic in q. The former overhead comes from the fan-in of addition gates, and the latter comes from implementing integer and modular arithmetic using Boolean operations on binary representations. (In this representation, the decomposition gate becomes a null operation.)

2.3 Sequentiality Assumption

Here we recall the lattice-based candidate sequentiality assumption recently proposed by Lai and Malavolta [LM23] (with some slight differences in the notation). This assumption was used as the foundation for a candidate *proof of sequential work* (PoSW); see Section 5 for further details.

Let q be a positive integer modulus and $\mathbf{g} \in \mathbb{Z}_q^\ell$ be a suitable "gadget" vector; for concreteness, we use the standard powers-of-two gadget defined by $\mathbf{g}^t = (1, 2, 4, \ldots, 2^{\ell-1})$ for $\ell = \lceil \log_2 q \rceil$, but all of our results easily adapt to other choices of gadgets (see [MP12] for further details). Let $\mathbf{g}^{-1} \colon \mathbb{Z}_q \to \mathbb{Z}^\ell$ be the corresponding "(bit) decomposition" function: $\mathbf{g}^{-1}(u)$ is binary and hence "short," and $\langle \mathbf{g}, \mathbf{g}^{-1}(u) \rangle = \mathbf{g}^t \cdot \mathbf{g}^{-1}(u) = u$ for any $u \in \mathbb{Z}_q$. Finally, extend the gadget and its decomposition operation to work on matrices (including vectors) as follows: for any positive integer n, let $\mathbf{G}_n := \mathbf{I}_n \otimes \mathbf{g}^t \in \mathbb{Z}_q^{n \times n\ell}$ denote the block-wise application of $\mathbf{g}^t$ to each ℓ-dimensional column block, and let $\mathbf{G}_n^{-1}(\cdot)$ denote the entry-wise application of $\mathbf{g}^{-1}$ on any matrix $\mathbf{U}$ having n rows, so that $\mathbf{G}_n \cdot \mathbf{G}_n^{-1}(\mathbf{U}) = \mathbf{U}$.

For dimensions n and $m = n\ell$, matrix $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times m}$, vector $\mathbf{u}_0 \in \mathbb{Z}_q^n$, and sequentiality parameter T, Lai and Malavolta [LM23] consider the following linear system, where $\mathbf{G} = \mathbf{G}_n \in \mathbb{Z}_q^{n \times m}$:[2]

$$\underbrace{\begin{pmatrix} \mathbf{G} & & & & \\ \bar{\mathbf{A}} & \mathbf{G} & & & \\ & \bar{\mathbf{A}} & \ddots & & \\ & & \ddots & \mathbf{G} & \\ & & & \bar{\mathbf{A}} & \mathbf{G} \end{pmatrix}}_{\bar{\mathbf{A}}_T \in \mathbb{Z}_q^{Tn \times Tm}} \cdot \underbrace{\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_T \end{pmatrix}}_{\mathbf{x} \in \mathbb{Z}^{Tm}} = \begin{pmatrix} \mathbf{u}_0 \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix} \in \mathbb{Z}_q^{Tn} . \tag{2.3}$$

As the foundation of their PoSW candidate, they pose the following sequentiality assumption. (They actually give a general family of assumptions with various parameters, but in this work we mainly focus on the instantiation underlying the PoSW.)

Conjecture 2.1. For sufficiently large q, given uniformly random $(\bar{\mathbf{A}}, \mathbf{u}_0)$, computing a "somewhat short" solution $\mathbf{x} \in \mathbb{Z}^{Tm}$ to Equation (2.3)—specifically, one having ℓ_∞ norm $\|\mathbf{x}\|_\infty \leq (Cn)^{2 \log T}$ for a certain

[2]For convenience, we have made a slight but immaterial tweak to the system appearing in [LM23], by dropping the $\bar{\mathbf{A}}$ matrix in the bottom-right block (that appears below our bottom-right $\mathbf{G}$ matrix) and dropping the bottom-most block $\mathbf{u}_T$ on the right-hand side (below our zero blocks). It is easy to see that the two systems are equivalent.

In addition, [LM23] considers a more compact and "algebraically structured" version of the system over a certain polynomial ring, where each d-by-d block of $\bar{\mathbf{A}}$ is the (structured) multiplication matrix of a random ring element. Our attack works for arbitrary $\bar{\mathbf{A}}$, so for generality we adopt the above presentation. For simplicity, we adopt a typical instantiation where $d = \Omega(n)$, though we note that all our attacks also work for any $d = n^{\Omega(1)}$, at the cost of $O(1)$- and $(\log^{O(1)} T)$-factor larger depths in Sections 4 and 5, respectively.

constant $C > 0$—requires depth $(1 - o(1)) \cdot T$.[3]

Notice that a "very short" *binary* solution, which has ℓ_∞ norm $\|\mathbf{x}\|_\infty \leq 1$, can be computed in depth proportional to T, as $\mathbf{x}_1 = \mathbf{G}_n^{-1}(\mathbf{u}_0) \in \mathbb{Z}^m$ and then $\mathbf{x}_i = -\mathbf{G}_n^{-1}(\bar{\mathbf{A}}\mathbf{x}_{i-1}) \in \mathbb{Z}^m$ for $i = 2, \ldots, T$. This works because

$$\bar{\mathbf{A}}\mathbf{x}_{i-1} + \mathbf{G}\mathbf{x}_i = \bar{\mathbf{A}}\mathbf{x}_{i-1} - \bar{\mathbf{A}}\mathbf{x}_{i-1} = \mathbf{0} \ .$$

The reason for the gap between the "very short" bound obtained by the above computation, versus the "somewhat short" bound in the assumption, is the $O(n)^{2 \log T}$ "slack factor" in the proof of sequential work from [LM23]. More specifically, the honest prover can use a "very short" solution to convince the verifier, but the knowledge extractor can extract only a "somewhat short" solution from any (possibly malicious) prover that manages to convince the verifier. Looking ahead, the attacks we give in Section 4 crucially exploit this gap, using a low-depth computation to find a solution that is significantly longer than the "very short" one, but still below the "somewhat short" threshold.

3 Attack Framework

Here we develop a suite of general tools that can be combined in various ways to yield attacks on sequentiality assumptions and proofs of sequential work.

3.1 Gadget Trapdoors

We first recall from [MP12] the notion of a (gadget) *trapdoor* for a matrix $\mathbf{A} \in \mathbb{Z}_q^{N \times W}$, for any dimensions N, W. This is any "short" matrix $\mathbf{R} \in \mathbb{Z}^{W \times M}$, where $M = N\ell$, for which

$$\mathbf{A}\mathbf{R} = \mathbf{G}_N = \mathbf{I}_N \otimes \mathbf{g}^t \in \mathbb{Z}_q^{N \times M} \ . \tag{3.1}$$

More precisely, $\mathbf{R}$ should have suitably bounded operator norm $\|\mathbf{R}\|_{r \leftarrow p}$ for whatever p, r are most appropriate for the application. For example, when bounds on the Euclidean ℓ_2 norm are desired, the spectral norm $\|\mathbf{R}\|_{2 \leftarrow 2}$ is usually most useful. In this work we will also frequently use the ℓ_∞ norm, and also finer-grained *block-wise* norm bounds, as defined in Section 2. Observe that $\mathbf{G}_N$ itself has $\mathbf{I}_M$ as a trapdoor.

Using such a trapdoor, it is easy to compute, in low depth, a comparably short solution $\mathbf{x} \in \mathbb{Z}^W$ to $\mathbf{A}\mathbf{x} = \mathbf{u}$, for any syndrome $\mathbf{u} \in \mathbb{Z}_q^N$: simply let $\mathbf{x} = \mathbf{R} \cdot \mathbf{G}_N^{-1}(\mathbf{u})$. This works because both $\mathbf{R}$ and $\mathbf{G}_N^{-1}(\mathbf{u})$ are short, and

$$\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{R} \cdot \mathbf{G}_N^{-1}(\mathbf{u}) = \mathbf{G}_N \cdot \mathbf{G}_N^{-1}(\mathbf{u}) = \mathbf{u} \ .$$

Recall that $\mathbf{G}_N^{-1}$ applies $\mathbf{g}^{-1}$ to each entry of $\mathbf{u}$ independently, so $\mathbf{x}$ can be computed in depth $O(1)$. And because $\mathbf{G}_N^{-1}(\mathbf{u})$ is binary, for any $r \in [1, \infty]$ we have that $\mathbf{x}$ satisfies the (potentially block-wise) norm bound

$$\|\mathbf{x}\|_r \leq \|\mathbf{R}\|_{r \leftarrow \infty} \ . \tag{3.2}$$

For example, by Equation (2.2) this is at most $W^{1/r} \cdot \|\mathbf{R}\|_\infty \cdot M$, but tighter bounds may be available in specific circumstances.

[3] More specifically, the base of the exponential is $\gamma = 2d$, where again $d = \Omega(n)$ is the dimension of the structured blocks of $\bar{\mathbf{A}}$. Naturally, one can consider other norms as well, like the Euclidean ℓ_2 norm.

Subspace trapdoors. More generally, for some relations (including the specific one from [LM23]) it suffices, and will yield better bounds, to have a limited trapdoor with respect to just the top s rows for some $s \ll N$, which we call an *s-subspace* trapdoor.[4] For convenience of usage later on, we allow s to be arbitrary, including $s > N$. An s-subspace trapdoor of $\mathbf{A}$ is a short matrix $\mathbf{R} \in \mathbb{Z}^{W \times s\ell}$ for which

$$\mathbf{AR} = \mathbf{G}_{N,s\ell} := \mathbf{I}_{N,s} \otimes \mathbf{g}^t \in \mathbb{Z}_q^{N \times s\ell} , \tag{3.3}$$

where $\mathbf{I}_{a,b}$ is the a-by-b matrix whose diagonal entries are 1 and all other entries are 0; equivalently, it is obtained by padding $\mathbf{I}_{\min(a,b)}$ with zeros on the right or the bottom, as appropriate. So, $\|\mathbf{I}_{a,b}\|_{r \leftarrow p} = \|\mathbf{I}_{\min(a,b)}\|_{r \leftarrow p}$. Notice that the gadget matrix $\mathbf{G}_N$ itself has the matrix $\mathbf{I}_{M,s\ell}$ as an s-subspace trapdoor.

Similarly, we define the "left zero-padded" matrices

$$\mathbf{I}_{N,[r]s} := \begin{pmatrix} \mathbf{0}_{N,\min(r,s)} & \mathbf{I}_{N,s-\min(r,s)} \end{pmatrix} \quad \text{and} \quad \mathbf{G}_{N,[r\ell]s\ell} := \mathbf{I}_{N,[r]s} \otimes \mathbf{g}^t .$$

Then observe that $\mathbf{I}_{N,[0]s} = \mathbf{I}_{N,s}$ and $\mathbf{I}_{N_0+N_1,s} = \begin{pmatrix} \mathbf{I}_{N_0,} & {}^{s} \\ \mathbf{I}_{N_1,[N_0]s} \end{pmatrix}$, and similarly for $\mathbf{G}_{N_0+N_1,[r\ell]s\ell}$.

Definition 3.1 (s-admissible vector). For a positive integer s, a vector $\mathbf{u} \in \mathbb{Z}_q^N$ is *s-admissible* if all its entries below the first $\min(s, N)$ are zero.

Using an s-subspace trapdoor $\mathbf{R}$ for $\mathbf{A}$, for any s-admissible syndrome $\mathbf{u} \in \mathbb{Z}_q^N$, it is easy to compute, in depth $O(1)$, a solution to $\mathbf{Ax} = \mathbf{u}$ that satisfies the norm bound in Equation (3.2). Let $\mathbf{u}_s := \mathbf{I}_{s,N} \cdot \mathbf{u} \in \mathbb{Z}_q^s$ be the truncation or padding by zeros (as appropriate) of $\mathbf{u}$; observe that $\mathbf{I}_{N,s} \cdot \mathbf{u}_s = \mathbf{u}$ because $\mathbf{u}$ is s-admissible. Take $\mathbf{x} = \mathbf{R} \cdot \mathbf{G}_s^{-1}(\mathbf{u}_s) \in \mathbb{Z}^W$. Then we have that, as needed,

$$\mathbf{Ax} = \mathbf{G}_{N,s\ell} \cdot \mathbf{G}_s^{-1}(\mathbf{u}_s) = \mathbf{I}_{N,s} \cdot \mathbf{u}_s = \mathbf{u} .$$

3.2 Trapdoor Combiners

The core idea underlying our attacks is to *recursively* compute, in fairly low depth, a trapdoor for the block-triangular matrix of the linear system in question, using trapdoors for sub-matrices of the system. This trapdoor can then be used to find a short solution for any desired syndrome, as described above. In this section we give a variety of low-depth, non-recursive *combiner algorithms* for constructing trapdoors and finding short solutions using sub-trapdoors. Sections 4 and 5 then give *parallel* recursive "driver" algorithms that use these combiners for specific attacks.

The base case is where the system's matrix is simply the gadget matrix $\mathbf{G}_n$ for some (typically small) n—which trivially has the identity matrix $\mathbf{I}_{n\ell}$ as a trapdoor—or some other matrix having a known trapdoor. For the recursive case, suppose that the system's matrix $\mathbf{A}$ has $N = N_0 + N_1$ rows and block lower-triangular form

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_0 & \\ \mathbf{W} & \mathbf{A}_1 \end{pmatrix} , \tag{3.4}$$

where $\mathbf{A}_0$ and $\mathbf{A}_1$ respectively have N_0 and N_1 rows; typically, one would have $N_0 \approx N_1$. Note that the matrix $\bar{\mathbf{A}}_T$ from the system in Equation (2.3) has this form, where $\mathbf{A}_0 = \mathbf{A}_1 = \bar{\mathbf{A}}_{T/2}$ for even T (and similarly for odd T) and $\mathbf{W}$ is all zeros except in its upper-rightmost n-by-m block.

As an initial result, we show in the following lemma how to get a trapdoor for $\mathbf{A}$ by combining individual trapdoors for $\mathbf{A}_0, \mathbf{A}_1$. (We actually will not use the lemma as stated here in this work.)

[4] This definition and the associated techniques naturally generalize to any particular set of rows, or even to any subspace of the column space.

Lemma 3.2. *Let $\mathbf{A}$ have the form given in Equation (3.4). There is a depth-$O(1)$ arithmetic circuit* $\mathsf{CombTrap}$ *that, given $\mathbf{A}$ and a trapdoor $\mathbf{R}_i$ (having $M_i = N_i\ell$ columns) for each $\mathbf{A}_i$, outputs a trapdoor $\mathbf{R}$ (having $M = N\ell$ columns) for $\mathbf{A}$ satisfying for all $p, r \in [1, \infty]$ the block-wise bound*

$$\|\mathbf{R}\|_{r\leftarrow p} \leq \begin{pmatrix} \|\mathbf{R}_0\|_{r\leftarrow p} \\ \|\mathbf{R}_1\|_{r\leftarrow p} \cdot (M_0^{1-1/p} \cdot M_1^{1/p} + 1) \end{pmatrix} \tag{3.5}$$

Moreover, for $p = 2$, the bound holds with $O(\sqrt{M})$ in place of $(M_0^{1-1/p} \cdot M_1^{1/p} + 1)$, with high probability.

Proof. Observe that

$$\mathbf{A} \cdot \begin{pmatrix} \mathbf{R}_0 & \\ & \mathbf{R}_1 \end{pmatrix} = \begin{pmatrix} \mathbf{G}_{N_0} & \\ \mathbf{W}\mathbf{R}_0 & \mathbf{G}_{N_1} \end{pmatrix} .$$

Therefore, a trapdoor $\mathbf{R}$ for $\mathbf{A}$ is

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_0 & \\ & \mathbf{R}_1 \end{pmatrix} \begin{pmatrix} \mathbf{I}_{M_0} & \\ \mathbf{R}' & \mathbf{I}_{M_1} \end{pmatrix} = \begin{pmatrix} \mathbf{R}_0 & \\ \mathbf{R}_1\mathbf{R}' & \mathbf{R}_1 \end{pmatrix} \text{ where } \mathbf{R}' = \mathbf{G}_{N_1}^{-1}(-\mathbf{W}\mathbf{R}_0) \in \mathbb{Z}^{M_1 \times M_0} . \tag{3.6}$$

This is because

$$\mathbf{A}\mathbf{R} = \begin{pmatrix} \mathbf{G}_{N_0} & \\ \mathbf{W}\mathbf{R}_0 & \mathbf{G}_{N_1} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{M_0} & \\ \mathbf{R}' & \mathbf{I}_{M_1} \end{pmatrix} = \begin{pmatrix} \mathbf{G}_{N_0} & \\ \mathbf{W}\mathbf{R}_0 - \mathbf{W}\mathbf{R}_0 & \mathbf{G}_{N_1} \end{pmatrix} = \mathbf{G}_N .$$

Note that $\mathbf{R}$ can be computed (given $\mathbf{R}_0, \mathbf{R}_1, \mathbf{W}$) in depth $O(1)$ in our model (see Section 2.2), by first computing $\mathbf{W}\mathbf{R}_0$, then $\mathbf{R}'$, then $\mathbf{R}_1\mathbf{R}'$.

For the norm bound, first observe that $\mathbf{R}' \in \mathbb{Z}^{M_1 \times M_0}$ has binary entries, so $\|\mathbf{R}'\|_\infty \leq 1$. From Equation (3.6), and sub-additivity and transitivity of the operator norm, we get the block-wise bound

$$\|\mathbf{R}\|_{r\leftarrow p} \leq \begin{pmatrix} \|\mathbf{R}_0\|_{r\leftarrow p} \\ \|\mathbf{R}_1\|_{r\leftarrow p} \cdot (\|\mathbf{R}'\|_{p\leftarrow p} + 1) \end{pmatrix} .$$

The bound $\|\mathbf{R}'\|_{p\leftarrow p} \leq M_0^{1-1/p} \cdot M_1^{1/p}$ follows from Equation (2.2). Moreover, if we use a randomized, subgaussian variant of $\mathbf{g}^{-1}$, we get that $\|\mathbf{R}'\|_{2\leftarrow 2} = O(\sqrt{M})$ with high probability. ☐

3.2.1 Subspace Trapdoors

To attack the specific relation from [LM23] (see Equation (2.3)), because the syndrome is n-admissible it suffices to compute an n-*subspace* trapdoor, regardless of how large $N = Tn$ is. This can be done by a simple optimization of the above approach, which yields much better matrix-norm bounds, and somewhat better computation-depth bounds (for models with bounded fan-in).

Definition 3.3 (s-admissible matrix). For a positive integer s, we say that a matrix $\mathbf{A}$ having the form given in Equation (3.4) is s-*admissible* if all the columns of $\mathbf{W}$ are s-admissible vectors (Definition 3.1), i.e., all their entries below the first $\min(s, N_1)$ are zero.

More generally (looking ahead to Section 3.2.3), for positive integers $s_1, \ldots, s_{k-1}$, we say that a matrix $\mathbf{A}$ having the form given in Equation (3.10) below is $(s_1, \ldots, s_{k-1})$-admissible if all the columns of all the $\mathbf{W}_{i,j}$ are s_i-admissible vectors. When $s_1 = \cdots = s_{k-1} = s$, for brevity we typically just say that such a matrix is s-admissible.

For example, observe that the matrix $\bar{\mathbf{A}}_T$ from [LM23] (see Equation (2.3)) is n-admissible.

For s-admissible matrices, we generalize Lemma 3.2 as follows.[5] To simplify the statement, the reader may take s and all the s_i to be equal, and at most N_0 and N_1, which is the only parameterization we use in this work.

Lemma 3.4. *Let s, s_0, s_1 be positive integers such that $s_0 \geq \min(N_0, s)$ and $s_1 \geq \min(N_1, s - N_0)$, and let $\mathbf{A}$ have the form given in Equation (3.4) and be s_1-admissible. There is a depth-$O(1)$ arithmetic circuit* CombTrap *that, given input $\mathbf{A}$ and an s_i-subspace trapdoor $\mathbf{R}_i$ (having $s_i\ell$ columns) for each $\mathbf{A}_i$, outputs an s-subspace trapdoor $\mathbf{R}$ (having $s\ell$ columns) for $\mathbf{A}$ satisfying for all $p, r \in [1, \infty]$ the block-wise bound*

$$\|\mathbf{R}\|_{r\leftarrow p} \leq \begin{pmatrix} \|\mathbf{R}_0\|_{r\leftarrow p} \\ \|\mathbf{R}_1\|_{r\leftarrow p} \cdot s^{1-1/p} \cdot s_1^{1/p} \cdot \ell \end{pmatrix} , \tag{3.7}$$

and for $p = 2$, the bound holds with $O(\sqrt{\max(s, s_1)\ell})$ in place of $s^{1-1/p} \cdot s_1^{1/p} \cdot \ell$, with high probability.

More generally, the above claim holds with the following changes:

- *$s, s_0, s_1, \ldots, s_{k-1}$ are positive integers such that $s_i \geq \min(N_i, s - \sum_{j<i} N_j)$ for all i, and $\mathbf{A}$ has the form given in Equation (3.10) below and is $(s_1, \ldots, s_{k-1})$-admissible;*
- *the arithmetic circuit depth is $O(k)$;*
- *the block-wise norm bound on $\|\mathbf{R}\|_{r\leftarrow p}$ has $\|\mathbf{R}_0\|_{r\leftarrow p}$ for block 0 and $\|\mathbf{R}_i\|_{r\leftarrow p} \cdot s^{1-1/p} \cdot s_i^{1/p} \cdot \ell$ for every block $i > 0$, and for $p = 2$, factor $O(\sqrt{\max(s, s_i)\ell})$ in place of $s^{1-1/p} \cdot s_i^{1/p} \cdot \ell$.*

For clarity, we sometimes include the s and s_i parameters, or just s when they all are equal, as a subscript to CombTrap. We point out that unlike the bound (3.5) in Lemma 3.2, the matrix norm bound (3.7) in Lemma 3.4 does not have an additive $+1$ term, due to a difference in how the subspace trapdoor is constructed.

Proof. For simplicity of exposition, we prove the special case $k = 2$ in full detail, and describe the straightforward generalization to arbitrary k in Section 3.2.3 below.

The CombTrap algorithm works as follows:

1. Express $\mathbf{G}_{N,s\ell} = \begin{pmatrix} \mathbf{G}_{N_0, \quad s\ell} \\ \mathbf{G}_{N_1,[N_0\ell]s\ell} \end{pmatrix}$.
2. Let $\tilde{\mathbf{R}}_0$ be the truncation or zero-padding (as appropriate) of $\mathbf{R}_0$ to $s\ell$ columns, i.e., $\tilde{\mathbf{R}}_0 = \mathbf{R}_0 \cdot \mathbf{I}_{s_0\ell, s\ell}$.
3. Let $\mathbf{W}_{s_1}$ and $\mathbf{H}_{s_1}$ respectively be the truncations or zero-paddings (as appropriate) of $\mathbf{W}$ and $\mathbf{G}_{N_1,[N_0\ell]s\ell}$ to s_1 rows, i.e., $\mathbf{W}_{s_1} = \mathbf{I}_{s_1,N_1} \cdot \mathbf{W}$ and $\mathbf{H}_{s_1} = \mathbf{I}_{s_1,N_1} \cdot \mathbf{G}_{N_1,[N_0\ell]s\ell} = \mathbf{G}_{s_1,[N_0\ell]s\ell}$.
4. Output

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_0 & \\ & \mathbf{R}_1 \end{pmatrix} \begin{pmatrix} \mathbf{I}_{s_0\ell,s\ell} \\ \mathbf{R}' \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{R}}_0 \\ \mathbf{R}_1\mathbf{R}' \end{pmatrix} \text{ where } \mathbf{R}' = \mathbf{G}_{s_1}^{-1}(\mathbf{H}_{s_1} - \mathbf{W}_{s_1}\tilde{\mathbf{R}}_0) \in \mathbb{Z}^{s_1\ell \times s\ell} . \tag{3.8}$$

We first prove correctness, i.e., that the above output $\mathbf{R}$ is an s-subspace trapdoor of $\mathbf{A}$. For the top block, we have that

$$\mathbf{A}_0\tilde{\mathbf{R}}_0 = \mathbf{A}_0\mathbf{R}_0 \cdot \mathbf{I}_{s_0\ell,s\ell} = (\mathbf{I}_{N_0,s_0} \otimes \mathbf{g}^t) \cdot (\mathbf{I}_{s_0,s} \otimes \mathbf{I}_{\ell,\ell}) = \mathbf{I}_{N_0,s} \otimes \mathbf{g}^t = \mathbf{G}_{N_0,s\ell} ,$$

[5] As with subspace trapdoors themselves, this modification naturally generalizes to any linear subspace of the column space.

where the penultimate equality follows from the mixed-product property and the fact that $s_0 \geq \min(N_0, s)$. For the bottom block, first observe that the columns of $\mathbf{G}_{N,s\ell}$ are $\min(N, s)$-admissible, and hence the columns of $\mathbf{G}_{N_1,[N_0\ell]s\ell}$ are s_1-admissible, since $s_1 \geq \min(N_1, s - N_0) = \min(N, s) - N_0$. Because the columns of $\mathbf{W}$ are also s_1-admissible, we have that $\mathbf{I}_{N_1,s_1} \cdot \mathbf{H}_{s_1} = \mathbf{G}_{N_1,[N_0\ell]s\ell}$ and $\mathbf{I}_{N_1,s_1} \cdot \mathbf{W}_{s_1} = \mathbf{W}$. Then the correctness of the bottom block can be verified by the following calculation, similar to the one in the proof of Lemma 3.2:

$$\mathbf{A}_1\mathbf{R}_1 \cdot \mathbf{R}' = \mathbf{G}_{N_1,s_1\ell} \cdot \mathbf{G}_{s_1}^{-1}(\mathbf{H}_{s_1} - \mathbf{W}_{s_1}\tilde{\mathbf{R}}_0) = \mathbf{I}_{N_1,s_1} \cdot (\mathbf{H}_{s_1} - \mathbf{W}_{s_1}\tilde{\mathbf{R}}_0) = \mathbf{G}_{N_1,[N_0\ell]s\ell} - \mathbf{W}\tilde{\mathbf{R}}_0 \ .$$

Finally, the depth bound of $O(1)$ holds by inspection, and bounds on $\|\mathbf{R}'\|_{p\leftarrow p}$ analogous to the ones in the proof of Lemma 3.2 hold, with $s_1\ell$ in place of M_1 and $s\ell$ in place of M_0. □

Remark 3.5. By setting $s = N$ and $s_i = N_i$, we can recover Lemma 3.2 as a special case of Lemma 3.4, though with a worse guaranteed norm bound. (For $p = 2$, the high-probability bounds match.) However, it is possible to refine the bound in Lemma 3.4 to match the one in Lemma 3.2, using the following factor in place of $s^{1-1/p} \cdot s_1^{1/p} \cdot \ell$:

$$\left(\min(N_0, s)^{1-1/p} \cdot s_1^{1/p} \cdot \ell + \mathrm{sign}(\max(s - N_0, 0))\right) \ .$$

This factor can be obtained by inspecting more carefully the structure of $\mathbf{H}_{s_1} - \mathbf{W}_{s_1}\tilde{\mathbf{R}}_0$: for $s > N_0$, $\mathbf{H}_{s_1} = \mathbf{G}_{s_1,[N_0\ell]s\ell}$ has a submatrix $\mathbf{G}_{s_1,(s-N_0)\ell}$ on the right and $\tilde{\mathbf{R}}_0$ has zero-padding for these columns, and then for this submatrix, $\mathbf{G}_{s_1}^{-1}(\mathbf{G}_{s_1,(s-N_0)\ell}) = \mathbf{I}_{s_1\ell,(s-N_0)\ell}$.

3.2.2 Combining Solver

Here we describe a slightly optimized solution finder, which is important for our attack on the PoSW protocol with its original norm bounds (see Section 5). Instead of building a full trapdoor from sub-trapdoors and then using it to find a comparably short solution for a desired syndrome, we can instead *directly* use the sub-trapdoors to get a somewhat shorter solution. Essentially, this optimization corresponds to solving the system "slightly honestly," where we sequentially compute a solution one block at a time, using the sub-trapdoors to compute each block of the solution in low depth. Comparing the combination of Equations (3.2) and (3.7) above with Equation (3.9) below, this method yields better block-wise norm bounds for (say) the ℓ_∞ norm by at least an $s\ell$ factor for an s-subspace trapdoor. (Here as in Lemma 3.4, the reader may take all the s_i to be equal and at most N_0 and N_1, which is the only parameterization we will use in this work.)

Lemma 3.6. *Let s_0, s_1 be positive integers, let $\mathbf{A}$ have the form given in Equation (3.4) and be s_1-admissible, and let $\mathbf{u} = \left(\begin{smallmatrix}\mathbf{u}_0\\ \mathbf{u}_1\end{smallmatrix}\right) \in \mathbb{Z}_q^N$ be a syndrome where each $\mathbf{u}_i \in \mathbb{Z}_q^{N_i}$ and is s_i-admissible. There is a depth-$O(1)$ arithmetic circuit* CombSolve *that, given $\mathbf{A}$, $\mathbf{u}$, and an s_i-subspace trapdoor $\mathbf{R}_i$ (having $s_i\ell$ columns) for each $\mathbf{A}_i$, outputs a solution to $\mathbf{A}\mathbf{x} = \mathbf{u}$ satisfying for any $p, r \in [1, \infty]$ the block-wise norm bound*

$$\|\mathbf{x}\|_r \leq \begin{pmatrix} \|\mathbf{R}_0\|_{r\leftarrow p} \cdot (s_0\ell)^{1/p} \\ \|\mathbf{R}_1\|_{r\leftarrow p} \cdot (s_1\ell)^{1/p} \end{pmatrix} \ . \tag{3.9}$$

More generally, the above claim holds with the following changes:

- *$s_0, s_1, \ldots, s_{k-1}$ are positive integers, $\mathbf{A}$ has the form given in Equation (3.10) below and is $(s_1, \ldots, s_{k-1})$-admissible, and $\mathbf{u} = (\mathbf{u}_i)_{i=0}^{k-1} \in \mathbb{Z}_q^N$ and each $\mathbf{u}_i$ is s_i-admissible;*
- *the arithmetic circuit depth is $O(k)$;*

- *the block-wise norm bound on* $\|\mathbf{x}\|_r$ *has* $\|\mathbf{R}_i\|_{r\leftarrow p} \cdot (s_i \ell)^{1/p}$ *for every block* $i \geq 0$.

Proof. As in the proof of Lemma 3.4, we prove the special case $k = 2$ in full detail, and describe the immediate generalization to arbitrary k in Section 3.2.3.

For our purposes in Section 5.3 it is convenient to define CombSolve in terms of a "helper" function CombSolveHelper that has a slightly different interface. Given:

- $\mathbf{A}$ and an initial solution block $\mathbf{x}_0$ that is meant to satisfy $\mathbf{A}_0\mathbf{x}_0 = \mathbf{u}_0$,
- the remaining syndrome $\mathbf{u}_1 \in \mathbb{Z}_q^{N_1}$, and
- an s_1-subspace trapdoor $\mathbf{R}_1$ for $\mathbf{A}_1$ (so $\mathbf{R}_0$ is not needed),

it outputs a *full* solution $\mathbf{x} = \left(\begin{smallmatrix}\mathbf{x}_0\\ \mathbf{x}_1\end{smallmatrix}\right)$ to $\mathbf{A}\mathbf{x} = \mathbf{u}$, by computing a solution to $\mathbf{A}_1\mathbf{x}_1 = \mathbf{u}_1' := \mathbf{u}_1 - \mathbf{W}\mathbf{x}_0$, namely, $\mathbf{x}_1 = \mathbf{R}_1 \cdot \mathbf{G}_{s_1}^{-1}(\mathbf{I}_{s_1,N_1} \cdot \mathbf{u}_1')$. Note that $\mathbf{x}_1$ is correct because $\mathbf{R}_1$ is an s_1-subspace trapdoor for $\mathbf{A}_1$, and $\mathbf{u}_1$ and the columns of $\mathbf{W}$ are s_1-admissible. By inspection, the depth of CombSolveHelper is $O(1)$.

We now define CombSolve. It computes the initial solution block as $\mathbf{x}_0 = \mathbf{R}_0 \cdot \mathbf{G}_{s_0}^{-1}(\mathbf{I}_{s_0,N_0} \cdot \mathbf{u}_0)$, and then outputs the full solution CombSolveHelper$(\mathbf{A}, \mathbf{x}_0, \mathbf{u}_1, \mathbf{R}_1)$. The correctness of $\mathbf{x}_0$ (i.e., that $\mathbf{A}_0\mathbf{x}_0 = \mathbf{u}_0$) follows from the hypotheses that $\mathbf{R}_0$ is an s_0-subspace trapdoor for $\mathbf{A}_0$, and that $\mathbf{u}_0$ is s_0-admissible. Observe that the output is computed in depth $O(1)$, and satisfies $\|\mathbf{x}_i\|_r \leq \|\mathbf{R}_i\|_{r\leftarrow p} \cdot (s_i\ell)^{1/p}$ because the output of $\mathbf{G}_{s_i}^{-1}$ is binary. □

3.2.3 Larger Arity

We now describe how all of the above easily generalizes to the case where $\mathbf{A}$ can be split into larger numbers of blocks, as claimed in Lemmas 3.4 and 3.6. This will be needed for our attacks on PoSW variants in Section 5. Specifically, suppose that for some $k \geq 2$, the matrix $\mathbf{A}$ has the form

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_0 & & & \\ \mathbf{W}_{1,0} & \mathbf{A}_1 & & \\ \vdots & \vdots & \ddots & \\ \mathbf{W}_{k-1,0} & \mathbf{W}_{k-1,1} & \cdots & \mathbf{A}_{k-1} \end{pmatrix}, \tag{3.10}$$

where $\mathbf{A}_i$ has N_i rows.

The claimed generalization of CombTrap from Lemma 3.4 is as follows. It can be verified that under the hypotheses of Lemma 3.4, if $\mathbf{R}_i$ is an s_i-subspace trapdoor (having $s_i\ell$ columns) for $\mathbf{A}_i$, then an s-subspace trapdoor for $\mathbf{A}$ is

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_0 & & & \\ & \mathbf{R}_1 & & \\ & & \ddots & \\ & & & \mathbf{R}_{k-1} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{s_0\ell,s\ell} \\ \mathbf{R}_1' \\ \vdots \\ \mathbf{R}_{k-1}' \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{R}}_0 \\ \tilde{\mathbf{R}}_1 \\ \vdots \\ \tilde{\mathbf{R}}_{k-1} \end{pmatrix}, \tag{3.11}$$

where letting $N_{i'}' = \sum_{i<i'} N_i$, we express

$$\mathbf{G}_{N,s\ell} = \begin{pmatrix} \mathbf{G}_{N_0,[N_0'\ell]s\ell} \\ \mathbf{G}_{N_1,[N_1'\ell]s\ell} \\ \vdots \\ \mathbf{G}_{N_{k-1},[N_{k-1}'\ell]s\ell} \end{pmatrix}$$

and set

$$\begin{aligned}\mathbf{H}_{s_i;i} &= \mathbf{I}_{s_i,N_i} \cdot \mathbf{G}_{N_i,[N_i'\ell]s\ell} = \mathbf{G}_{s_i,[N_i'\ell]s\ell} \ , \\ \mathbf{W}_{s_i;i,j} &= \mathbf{I}_{s_i,N_i} \cdot \mathbf{W}_{i,j} \ , \\ \mathbf{R}_i' &= \mathbf{G}_{s_i}^{-1}\Big(\mathbf{H}_{s_i;i} - \sum_{j<i} \mathbf{W}_{s_i;i,j} \cdot \tilde{\mathbf{R}}_j\Big) \ . \end{aligned} \tag{3.12}$$

The matrices $\mathbf{R}_j'$ and $\tilde{\mathbf{R}}_j$ can be computed in k sequential stages for $j = 0, \dots, k-1$, accumulating the $\mathbf{W}_{s_i;i,j} \cdot \tilde{\mathbf{R}}_j$ terms in parallel over all $i > j$ in the jth stage. So, $\mathbf{R}$ can be computed in arithmetic depth $O(k)$. The claimed block-wise matrix norm bound on $\mathbf{R}$ follows immediately from operator-norm bounds on the binary matrices $\mathbf{R}_i'$, as in the proof of Lemma 3.4.

The optimized solver CombSolve from Lemma 3.6 generalizes in a very similar way. Namely, the "helper" function CombSolveHelper sequentially computes solution blocks

$$\mathbf{x}_i = \mathbf{R}_i \cdot \mathbf{G}_{s_i}^{-1}\Big(\mathbf{I}_{s_i,N_i} \cdot \mathbf{u}_i - \sum_{j<i} \mathbf{W}_{s_i;i,j} \cdot \mathbf{x}_j\Big)$$

for $i = 1, \dots, k-1$, where the summations are accumulated in parallel in each stage.

4 Attack on the Assumption

Here we use the tools from Section 3 to give a recursive attack on the sequentiality assumption underlying the PoSW from [LM23] (see Section 2.3). Our ultimate result in this respect is as follows.

Theorem 4.1 (Attack on the sequentiality assumption). *Let $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n \times m}$, T and $k \in [2, T]$ be positive integers, and $\bar{\mathbf{A}}_T$ be the matrix given in Equation (2.3). Given $\bar{\mathbf{A}}$, T, and k, an n-subspace trapdoor $\bar{\mathbf{R}}_T$ for $\bar{\mathbf{A}}_T$ satisfying the following (scalar) norm bound for any $p, r \in [1, \infty]$ can be computed by a depth-$\tilde{O}_{n,q}(k \log_k T)$ Boolean circuit:*

$$\|\bar{\mathbf{R}}_T\|_{r \leftarrow p} \le (k^{1/r} \cdot m)^{\lceil \log_k T \rceil} \cdot \|\mathbf{I}_m\|_{r \leftarrow p} \ .$$

Moreover, for $p = 2$ and $T = 2^{o(m)}$, the bound holds with $\sqrt{m}$ in place of m in the factor on the left, with high probability.

As a result, a solution $\mathbf{x}$ to Equation (2.3) satisfying the following (scalar) norm bound for any $p \in [1, \infty]$ can be computed by a depth-$\tilde{O}_{n,q}(k \log_k T)$ Boolean circuit:

$$\|\mathbf{x}\|_p \le (k^{1/p} \cdot m)^{\lceil \log_k T \rceil} \cdot m^{1/p} \ ,$$

and for $p = 2$ and $T = 2^{o(m)}$ the bound holds with $\sqrt{m}$ in place of m in the factor on the left, with high probability.

We prove this theorem below as a corollary of a much more general setup and attack. As a couple of examples, we can get the following parameterizations of Theorem 4.1:

- For $k = 2$, in depth just $\tilde{O}_{n,q}(\log T)$ we get a solution having ℓ_∞ norm bounded by $m^{\lceil \log_2 T \rceil}$, or ℓ_2 norm (and hence ℓ_∞ norm as well) bounded by $(2m)^{(\lceil \log_2 T \rceil + 1)/2}$ with high probability.

$\mathsf{RecTrap}_s(\mathbf{A})$

if $\mathbf{A} = \mathbf{G}_n$: **return** $\mathbf{I}_{n\ell,s\ell}$
foreach i *in parallel* : $\mathbf{R}_i = \mathsf{RecTrap}_s(\mathbf{A}_i)$
return $\mathsf{CombTrap}_s(\mathbf{A}, [\mathbf{R}_i]_i)$

Figure 1: Algorithm that computes a subspace trapdoor for any recursively s-admissible matrix $\mathbf{A}$.

- By setting $k \approx T^{\varepsilon}$ for an arbitrarily small constant $\varepsilon > 0$, in depth $\tilde{O}_{n,q}(T^{\varepsilon})$ we get a solution of just polynomially bounded ℓ_∞ norm $m^{1/\varepsilon}$.

Regarding the the assumption made in [LM23] (Conjecture 2.1), under the (very mild) condition $q = 2^{o(n)}$ and hence $m = o(n^2)$, the ℓ_∞ norm of our attack's solution is less than the bound of $\Theta(n)^{2\log_2 T}$ from the assumption. So, in a typical parameterization where $T \gg n$ (or even just $\log T = \log^{\Omega(1)} n$) and $\log q = \tilde{O}_{n,T}(1) = \tilde{O}_T(1)$, the attack's polylogarithmic $\tilde{O}_T(1)$ depth falsifies (even a major weakening of) the assumption, which posits that finding such a solution requires linear depth $(1 - o(1)) \cdot T$ (or in weaker form, polynomial depth T^{ε} for some constant $\varepsilon > 0$).

General attack. Our attack applies to a broad generalization of the system in Equation (2.3), namely, any one in which the matrix is block lower-triangular and has "gadget" matrices $\mathbf{G}$ (or any other matrices having known trapdoors) as the diagonal blocks. For simplicity of presentation, throughout this section and Section 5, our general attack algorithms assume that the system's matrix $\mathbf{A}$ is *implicitly* given in block form following Equation (3.4) (or more generally, Equation (3.10)), and similarly for its component $\mathbf{A}_i$ matrices, *recursively*. Then, an attack against a specific assumption or protocol is obtained as an instantiation of the general attack, by specifying a recursive block structure for the proposal in question.

Definition 4.2 (Recursively s-admissible). For a positive integer s, a matrix $\mathbf{A}$ is *recursively s-admissible* if $\mathbf{A} = \mathbf{G}_n$ for some n (the base case), or if it has the form given in Equation (3.4) (or more generally, Equation (3.10)), it is s-admissible, and all its $\mathbf{A}_i$ submatrices are recursively s-admissible.

For example, notice that the matrix $\bar{\mathbf{A}}_T$ from Equation (2.3) is recursively n-admissible for any choice of arity $k \in [2, T]$, with a recursive block structure of depth $\lceil \log_k T \rceil$.

In the rest of this section, we define and analyze a simple recursive "driver" $\mathsf{RecTrap}$ of the trapdoor combiner $\mathsf{CombTrap}$ from Lemma 3.4; see Figure 1 for its formal definition. Then we show that a straightforward instantiation of this driver yields Theorem 4.1 as corollary.

Lemma 4.3. *Let $\mathbf{A}$ be a recursively s-admissible matrix that recursively has the form given in Equation (3.10) for some arity k and recursion depth at most d (where $d = 0$ is the base case). There is a depth-$O(kd)$ arithmetic circuit $\mathsf{RecTrap}_s$ that, given $\mathbf{A}$, computes an s-subspace trapdoor $\mathbf{R}$ for $\mathbf{A}$ satisfying for any $p, r \in [1, \infty]$ the (scalar) norm bound*

$$\|\mathbf{R}\|_{r \leftarrow p} \leq (k^{1/r} \cdot s\ell)^d \cdot \|\mathbf{I}_{s\ell}\|_{r \leftarrow p} .$$

Moreover, for $p = 2$ and $k^d = 2^{o(s\ell)}$, the bound holds with $\sqrt{s\ell}$ in place of $s\ell$ in the factor on the left, with high probability. In particular, for $r = \infty$ we have that $\|\mathbf{R}\|_{\infty \leftarrow p} \leq (s\ell)^d$, and $\|\mathbf{R}\|_{\infty \leftarrow 2} \leq (s\ell)^{d/2}$ with high probability.

Proof. At each level of the recursion, by Lemma 3.4, $\mathsf{CombTrap}$ has arithmetic depth $O(k)$. Hence the overall arithmetic depth of $\mathsf{RecTrap}_s(\mathbf{A})$ is $O(kd)$.

For the operator norm of $\mathbf{R}$, again by Lemma 3.4, we have the recurrence

$$\|\mathbf{R}\|_{r\leftarrow p} \leq \left\| \begin{pmatrix} \|\mathbf{R}_0\|_{r\leftarrow p} \\ \|\mathbf{R}_1\|_{r\leftarrow p} \cdot s\ell \\ \vdots \\ \|\mathbf{R}_{k-1}\|_{r\leftarrow p} \cdot s\ell \end{pmatrix} \right\|_r \leq k^{1/r} \cdot \max_i \|\mathbf{R}_i\|_{r\leftarrow p} \cdot s\ell \ .$$

For $p = 2$, the recurrence holds with $\sqrt{s\ell}$ in place of $s\ell$, with high probability. The base case of this recurrence is $\|\mathbf{I}_{n\ell,s\ell}\|_{r\leftarrow p} \leq \|\mathbf{I}_{s\ell}\|_{r\leftarrow p}$, and the recursion has depth at most d. This yields the claimed bound on $\|\mathbf{R}\|_{r\leftarrow p}$. □

Proof of Theorem 4.1. Recall that the matrix $\bar{\mathbf{A}}_T$ from Equation (2.3) is recursively n-admissible for any choice of arity $k \in [2, T]$, with a corresponding depth of $d = \lceil \log_k T \rceil$. The bound on $\|\bar{\mathbf{R}}_T\|_{r\leftarrow p}$ then follows immediately from Lemma 4.3 and the fact that $m = n\ell$. The lemma also says that the arithmetic circuit computing $\bar{\mathbf{R}}_T$ has depth $O(kd) = O(k \log_k T)$ in our model (see Section 2.2).

For compiling the arithmetic circuit to a Boolean circuit, observe that the main computations in $\mathsf{RecTrap}$ are the $\mathsf{CombTrap}$ calls, and in the latter (see Equations (3.11) and (3.12)), the fan-in of the addition gates is at most $n\ell = O(n \log q)$: the computation of $\mathbf{R}_i \cdot \mathbf{R}'_i$ uses inner products of dimension $s_i\ell = n\ell$, and for the computation of $\mathbf{W}_{n;i,j} \cdot \tilde{\mathbf{R}}_j$, by the structure of $\bar{\mathbf{A}}_T$, all but the rightmost $n\ell$ columns of $\mathbf{W}_{n;i,j}$ are zero. Hence the Boolean circuit has depth $\tilde{O}_{n,q}(k \log_k T)$, as claimed.

For the particular solution $\mathbf{x}$, we use the subspace-trapdoor solver (see Section 3.1), letting $\mathbf{x} = \bar{\mathbf{R}}_T \cdot \mathbf{G}_n^{-1}(\mathbf{u}_0)$. Then

$$\|\mathbf{x}\|_p \leq \|\bar{\mathbf{R}}_T\|_{p\leftarrow p} \cdot \|\mathbf{G}_n^{-1}(\mathbf{u}_0)\|_p \leq (k^{1/p} \cdot m)^{\lceil \log_k T \rceil} \cdot m^{1/p} \ .$$

□

Remark 4.4 (Relation to preprocessing attack). We note that [LM23, Section 3.2] describes a *preprocessing* attack on a variant of the assumption from Section 2.3 in which $\bar{\mathbf{A}}$ is *fixed in advance*. (Recall that in the actual proposed assumption, $\bar{\mathbf{A}}$ is sampled as part of the instance.) This attack uses depth at least T during the preprocessing, but arithmetic depth only $O(1)$ once a desired right-hand side of Equation (2.3) is decided.

In the language of our attack, the preprocessing can be seen as naïvely computing a (very short) n-subspace trapdoor for $\bar{\mathbf{A}}_T$ in large depth, by just honestly computing a solution for each column of $\mathbf{G}_{Tn,n\ell}$. Our attack exponentially improves on the depth (but with a worse norm) by recursively computing an n-subspace trapdoor from such a trapdoor for a multiplicatively smaller value of T (instead of using honest evaluations). This falsifies the assumption underlying the PoSW protocol from [LM23], without using any preprocessing.

5 Attacks on Proofs of Sequential Work

Lai and Malavolta [LM23] also gave a candidate *proof of sequential work* (PoSW) protocol and proved its security based on the sequentiality assumption stated in Section 2.3. While the attack from Section 4 strongly *falsifies the assumption*, and thus renders the security proof vacuous, it does not immediately follow that the *PoSW itself* is broken. Indeed, the attack as stated does not break the protocol, because it produces a solution vector whose components have much larger norms than the "honestly computed" ones, so running the (honest) prover with this vector would not yield short enough component vectors to convince the verifier.

$\mathsf{Prover}(\bar{\mathbf{A}}, T, \mathbf{u}_0, \{\mathbf{u}_i\}_{i\in[T]}, \{\mathbf{x}_i\}_{i\in[T]})$	$\mathsf{Verifier}(\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times n\ell}, T, \mathbf{u}_0, \mathbf{u}_T \in \mathbb{Z}_q^n, \beta)$
if $T = 1$:	**if** $T = 1$:
send $\mathbf{x}_1$	receive $\mathbf{x}_1$
return	**return** $[\lVert\mathbf{x}_1\rVert_\infty \le \beta$
	$\wedge\, \mathbf{G}\mathbf{x}_1 = \mathbf{u}_0 \wedge -\bar{\mathbf{A}}\mathbf{x}_1 = \mathbf{u}_1]$
$T' = (T-1)/2$	$T' = (T-1)/2$
send $\mathbf{x}_{T'+1}$	receive $\mathbf{x}_{T'+1}$
	if $\lVert\mathbf{x}_{T'+1}\rVert_\infty > \beta$: **return** 0
receive c	send random c with $\lvert c\rvert \le \gamma$
$\mathbf{u}'_i = \boxed{c\,\cdot}\,\mathbf{u}_i + \mathbf{u}_{i+T'+1}$	$\mathbf{u}'_0 = \boxed{c\,\cdot}\,\mathbf{u}_0 + (-\bar{\mathbf{A}}\mathbf{x}_{T'+1})$
$\mathbf{x}'_i = \boxed{c\,\cdot}\,\mathbf{x}_i + \mathbf{x}_{i+T'+1}$	$\mathbf{u}'_{T'} = \boxed{c\,\cdot}\,\mathbf{G}\mathbf{x}_{T'+1} + \mathbf{u}_T$
$\mathsf{Prover}(\bar{\mathbf{A}}, T', \mathbf{u}'_0, \{\mathbf{u}'_i\}_{i\in[T']}, \{\mathbf{x}'_i\}_{i\in[T']})$	**return** $\mathsf{Verifier}(\bar{\mathbf{A}}, T', \mathbf{u}'_0, \mathbf{u}'_{T'}, 2\boxed{\gamma'}\cdot\beta)$

Figure 2: The core algorithms of the PoSW from [LM23], with $\boxed{\text{boxed}}$ modifications. For simplicity, we give the code only for T of the form $T = 2^t - 1$. The original PoSW applies the challenge factor c to the *second* term in each sum (instead of the first), and it uses $\gamma' = \gamma := \Omega(n)$.

In this section we give attacks that use the tools from Section 3 in a more sophisticated way, to break two PoSWs that are very similar (but not quite identical) to the one given in [LM23]; see Figure 2 for the formal definition. Both variants tweak the core "folding" operation to multiply the verifier's random challenge by the *first*, rather than the *second*, half of the prover's solution vector. In the design of the protocol the choice is arbitrary, since the analysis and security proof apply equally well to either version, but the choice seems to have significant implications for attacks, as we explain below. In addition, in our first variant protocol, the verifier checks the prover's responses using *somewhat relaxed norm bounds* (which are polynomially related to the original ones). Our second variant uses the original norm bounds from [LM23]; the only change is the tweaked folding operation.

The rest of this section is organized as follows. In Section 5.1 we describe some of the challenges that arise in attacking the PoSW protocol of [LM23]. In Section 5.2 we break the first variant protocol using the simple $\mathsf{RecTrap}$ algorithm from Section 4, but with a slightly different recursive block decomposition of the system's matrix. Then in Section 5.3 we break the second variant using a much more sophisticated recursive strategy.

5.1 Challenges in Attacking PoSWs

We start by describing some of the additional challenges that arise in attacking the PoSW protocol of [LM23], which motivate our protocol tweaks and enhanced attacks. See Figure 2 for a formal definition of the protocol, with our tweaks highlighted.

Structure of the PoSW. In the PoSW, the honest prover computes a short solution $\mathbf{x}$ to Equation (2.3), then engages in an interactive public-coin protocol to convince the verifier that it knows such a solution. (The protocol can be made non-interactive in the usual way via the Fiat–Shamir transform.) To do this, it uses the verifier's small random challenge to linearly "fold" the first and second halves of $\mathbf{x}$ together, which yields a somewhat longer solution of half the dimension, then recursively proves knowledge of this folded solution.

More precisely, the prover first announces $\mathbf{u}_T = -\bar{\mathbf{A}}\mathbf{x}_T$ (or alternatively, announces $\mathbf{x}_T$ and has the verifier compute $\mathbf{u}_T$ itself) as its claimed result of the sequential computation. Then it gives a proof of knowledge of its solution $\mathbf{x}$ to Equation (2.3): it announces $\mathbf{x}_{(T+1)/2}$ (assume that T is odd for simplicity), which the verifier checks is short enough, and the verifier announces a small random challenge c.[6] Observe that the remaining halves of $\mathbf{x}$ form two solutions to reduced-dimension instances of Equation (2.3), with known right-hand sides of the appropriate form. So, the prover linearly combines these solutions using c, and recursively proves knowledge of the resulting (somewhat longer) solution in the same manner, until the dimension is small enough to simply reveal and check the solution. Note that with each successive stage of the recursion, the verifier must apply a more relaxed norm check on the prover's announced value, because folding increases the norm of the solution by some fixed factor (independent of T).

The difficulty. The key challenge in attacking the PoSW seems to be as follows: (1) the prover must first announce some $\mathbf{u}_T$ to the verifier, so (2) the prover can know *at most one* short solution for whatever value it announces, and (3) the only way we see to convince the verifier is by knowing a solution whose middle component $\mathbf{x}_{(T+1)/2}$, along with all subsequent middle components under the recursive folding, are *nearly as short* as what the honest prover would compute. We elaborate on each of these points next.

The announced value of $\mathbf{u}_T$ represents the claimed result of the sequential computation that the prover supposedly performed to get a solution to Equation (2.3). It is straightforward to show that computing *distinct* short solutions, for *any* fixed and possibly adversarially chosen right-hand side, is at least as hard as solving the corresponding SIS problem for the random matrix $\bar{\mathbf{A}}$. So, under the standard assumption that SIS is intractable (see, e.g., [Ajt96, MR04]), once an efficient prover (of any depth) reveals some $\mathbf{u}_T$, it can know *at most one* short solution for it. The same goes for the later stages of the protocol with lower-dimensional instances of Equation (2.3), where the first and last components of the right-hand side are determined by the previous stages.[7]

With these constraints, the only way we see to convince the verifier is by proceeding exactly as the honest (specified) prover would, using a single known solution. This means that the middle component of the solution vector, and all subsequent middle components under recursive folding, need to satisfy the verifier's norm checks. In particular, it seems that *whichever half of the solution is multiplied by the verifier's challenge needs to be nearly as short as what the honest prover would compute*. The PoSW from [LM23] multiplies by the *second* half, and we do not see a way to generate such a short second half in depth significantly less than T. This is because our techniques yield trapdoors and solutions whose second halves are larger than their first halves by some polynomial factor. Again we stress that the attack from Section 4 achieves low depth by exploiting the moderately large slack factor: the solution is merely "somewhat short" in its second half (and fourth quarter, etc.), but not nearly as short as what the honest computation produces.

[6] As mentioned before, [LM23] considers an "algebraically structured" version of the system Equation (2.3) over a certain commutative polynomial ring, and the challenge c is a ring element. In our treatment, we actually need to model the challenge c as a matrix, and measure its shortness by some suitable operator norm. The norm bound on c used in [LM23] translates to $\gamma = 2d = \Omega(n)$ in our setting; see the latter part of Footnote 2. Moreover, because the ring is commutative, c commutes with other matrix factors, and in particular with $\bar{\mathbf{A}}$. For simplicity, we still write c as a scalar, and we give more details about treating c as a matrix in the analysis below.

[7] This state of affairs is quite different from the context of the attack from Section 4, where the adversary is not bound to any particular right-hand side of Equation (2.3), and knows short solutions to many different right-hand sides via its trapdoor.

5.2 Breaking a Relaxed PoSW

The above discussion motivates a natural alternative folding operation: multiply the verifier's random challenge c by the *first*, rather than the *second*, half of the solution. That is, instead of folding a solution $\mathbf{x}$ into the lower-dimensional one $\mathbf{x}_{\text{first}} + c \cdot \mathbf{x}_{\text{last}}$, use $c \cdot \mathbf{x}_{\text{first}} + \mathbf{x}_{\text{last}}$. The security proof from [LM23] (modified in the obvious way) holds equally well for this option, because the two halves are treated symmetrically.[8]

Interestingly, this trivial modification makes the PoSW breakable using our attack framework. In this subsection, as a warmup we break this tweaked protocol with a verifier that also uses somewhat relaxed norm checks. The precise statement is as follows; we prove it below after introducing the definitions and tools needed to do so.

Theorem 5.1 (Attack on the PoSW with relaxed parameters). *There is a depth-$\tilde{O}_{n,q}(\log T)$ malicious prover (modeled as a Boolean circuit) that, given any $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{u}_0 \in \mathbb{Z}_q^n$, and positive integer T, convinces the verifier in the modified PoSW, where in the folding operation the challenge is multiplied by the first half of the solution, and parameters $\gamma' \geq (\gamma + m)/2$ and $\beta \geq m$ are used.*

The attack simply uses the RecTrap algorithm from Section 4, but with a slightly different recursive block decomposition of the matrix $\bar{\mathbf{A}}_T$. Essentially, this works because the $\mathbf{x}_{\text{last}}$ constructed by the attack is a small factor longer than $\mathbf{x}_{\text{first}}$, so the two summands $c \cdot \mathbf{x}_{\text{first}}, \mathbf{x}_{\text{last}}$ in the tweaked folding operation are more "balanced," hence their sum passes the appropriately relaxed norm checks. However, we also need to ensure that the revealed middle component is sufficiently short, at every stage of the recursion. For this we impose a suitable recursive block structure on $\bar{\mathbf{A}}_T$, which treats the middle components specially, as base cases.

Definition 5.2 (PoSW topology). A block vector or matrix has *PoSW topology* of depth d and base rows n if for the base case $d = 0$ it has n rows, and if for $d > 0$ it has three row blocks, respectively having PoSW topologies of depths $d-1, 0, d-1$ (all with base rows n).

Observe that by induction, a vector or matrix having PoSW topology has $T = 2^{d+1} - 1$ row blocks, of n rows each. In particular, for such T the recursively n-admissible matrix (Definition 4.2) $\bar{\mathbf{A}}_T$ from the system in Equation (2.3) can be given this PoSW topology. Note that $\bar{\mathbf{A}}_T$, like any recursively s-admissible matrix having PoSW topology with base rows n, is simply $\mathbf{G}_n$ in the base case $d = 0$.

Remark 5.3. In Definition 5.2, for simplicity of presentation, we define the PoSW topology only for matrices having exactly $T = 2^{d+1} - 1$ row blocks. Our attacks, analyses, and results in Sections 5.2 and 5.3 rely on the PoSW topology, and consequently work for T of this form. However, it is easy to generalize the recursive definition to work for an even number of blocks, following what is done in the original definition of the PoSW of [LM23]. This defines a PoSW topology for any positive integer T, and our attacks, analyses, and results immediately generalize analogously.

To aid in the analysis, we define the following operations on a vector or matrix $\mathbf{M}$ having PoSW topology of depth $d > 0$ and base rows n, whose three row blocks are denoted $\mathbf{M}_0, \mathbf{M}_1, \mathbf{M}_2$. The middle block, and folding operation with multiplier c, are respectively defined as

$$\mathsf{mid}(\mathbf{M}) := \mathbf{M}_1 = \begin{pmatrix} \mathbf{0}_{n,N'} & \mathbf{I}_n & \mathbf{0}_{n,N'} \end{pmatrix} \cdot \mathbf{M} \tag{5.1}$$

$$\mathsf{fold}_c(\mathbf{M}) := c\mathbf{M}_0 + \mathbf{M}_2 = \begin{pmatrix} c\mathbf{I}_{N'} & \mathbf{0}_{N',n} & \mathbf{I}_{N'} \end{pmatrix} \cdot \mathbf{M}\ , \tag{5.2}$$

[8] As far as we can tell, in [LM23] the specific choice of folding operation between these two options was made arbitrarily.

where $N' = T'n$ for $T' = (T-1)/2 = 2^d - 1$ is the number of rows in each of $\mathbf{M}_0, \mathbf{M}_2$. For convenience, for $\mathbf{M}$ having depth $d = 0$ also define $\mathsf{mid}(\mathbf{M}) := \mathbf{M} = \mathbf{I}_n \cdot \mathbf{M}$.

Observe that for a vector or matrix having a PoSW topology, its corresponding recursive block-wise vector of (vector or matrix) norms also has a PoSW topology of the same depth, with one row in the base case. In particular, if $|c| \leq \gamma$, then by the triangle inequality,

$$\|\mathsf{fold}_c(\mathbf{M})\|_{r\leftarrow p} \leq \mathsf{fold}_\gamma(\|\mathbf{M}\|_{r\leftarrow p}) \; . \tag{5.3}$$

This can be applied iteratively: for any c_i with $|c_i| \leq \gamma$ for all $1 \leq i \leq \tau$ where $\tau \leq d$, we have that

$$\|\mathsf{fold}_{c_\tau}(\cdots \mathsf{fold}_{c_1}(\mathbf{M})\cdots)\|_{r\leftarrow p} \leq \mathsf{fold}_\gamma^{(\tau)}(\|\mathbf{M}\|_{r\leftarrow p}) \; . \tag{5.4}$$

More generally, our treatment (which treats polynomial-ring elements as vectors and matrices) actually models the multiplier c as a square matrix having the same dimension as the base-case number of rows n. Then the folding is defined as

$$\mathsf{fold}_c(\mathbf{M}) := \begin{pmatrix} \mathbf{I}_{T'} \otimes c & \mathbf{0}_{N',n} & \mathbf{I}_{N'} \end{pmatrix} \cdot \mathbf{M} \; ,$$

and for any c with $\|c\|_{r\leftarrow r} \leq \gamma$, Equation (5.3) still holds, and similarly for its iterated version Equation (5.4).

Lemma 5.4. *Let $\mathbf{A}$ be a recursively s-admissible matrix with PoSW topology of depth d and base rows n. There is a depth-$O(d)$ arithmetic circuit $\mathsf{RecTrap}_s$ that, given $\mathbf{A}$, computes an s-subspace trapdoor $\mathbf{R}$ for $\mathbf{A}$ satisfying the following bound for any $p, r \in [1,\infty]$ and any integer $\tau \in [0,d]$:*

$$\mathsf{mid}(\mathsf{fold}_\gamma^{(\tau)}(\|\mathbf{R}\|_{r\leftarrow p})) \leq (\gamma + s\ell)^\tau \cdot \|\mathbf{I}_{n\ell,s\ell}\|_{r\leftarrow p} \cdot s\ell \; .$$

Proof. At each level of the recursion, by Lemma 3.4, $\mathsf{CombTrap}$ has arithmetic depth $O(1)$. Hence the overall arithmetic depth of $\mathsf{RecTrap}_s(\mathbf{A})$ is $O(d)$.

For the operator norm of $\mathbf{R}$, again by Lemma 3.4, we have the recurrence

$$\|\mathbf{R}\|_{r\leftarrow p} \leq \begin{pmatrix} \|\mathbf{R}_0\|_{r\leftarrow p} \\ \|\mathbf{I}_{n\ell,s\ell}\|_{r\leftarrow p} \cdot s\ell \\ \|\mathbf{R}_2\|_{r\leftarrow p} \cdot s\ell \end{pmatrix} \; ,$$

and the base case $d = 0$ has $\|\mathbf{R}\|_{r\leftarrow p} = \|\mathbf{I}_{n\ell,s\ell}\|_{r\leftarrow p}$. Hence $\|\mathbf{R}\|_{r\leftarrow p} \leq \mathsf{pf}(d)$, a block vector recursively defined as

$$\mathsf{pf}(d) := \begin{pmatrix} \mathsf{pf}(d-1) \\ \|\mathbf{I}_{n\ell,s\ell}\|_{r\leftarrow p} \cdot s\ell \\ \mathsf{pf}(d-1) \cdot s\ell \end{pmatrix} \; , \tag{5.5}$$

with base case $\mathsf{pf}(0) = \|\mathbf{I}_{n\ell,s\ell}\|_{r\leftarrow p}$. Therefore,

$$\mathsf{fold}_\gamma(\mathsf{pf}(d)) = \begin{pmatrix} \gamma\mathbf{I} & \mathbf{0} & \mathbf{I} \end{pmatrix} \cdot \begin{pmatrix} \mathsf{pf}(d-1) \\ \|\mathbf{I}_{n\ell,s\ell}\|_{r\leftarrow p} \cdot s\ell \\ \mathsf{pf}(d-1) \cdot s\ell \end{pmatrix} = (\gamma + s\ell) \cdot \mathsf{pf}(d-1) \; .$$

By iterating, for any integer $\tau \in [0,d]$, we get that $\mathsf{fold}_\gamma^{(\tau)}(\mathsf{pf}(d)) = (\gamma + s\ell)^\tau \cdot \mathsf{pf}(d-\tau)$. Therefore, as desired,

$$\begin{aligned} \mathsf{mid}(\mathsf{fold}_\gamma^{(\tau)}(\|\mathbf{R}\|_{r\leftarrow p})) &\leq \mathsf{mid}(\mathsf{fold}_\gamma^{(\tau)}(\mathsf{pf}(d))) \\ &= (\gamma + s\ell)^\tau \cdot \mathsf{mid}(\mathsf{pf}(d-\tau)) \\ &\leq (\gamma + s\ell)^\tau \cdot \|\mathbf{I}_{n\ell,s\ell}\|_{r\leftarrow p} \cdot s\ell \; . \end{aligned}$$

(Note that $\mathsf{mid}(\mathsf{pf}(d-\tau)) = \|\mathbf{I}_{n\ell,s\ell}\|_{r\leftarrow p}$ in the base case $\tau = d$, but $\|\mathbf{I}_{n\ell,s\ell}\|_{r\leftarrow p} \cdot s\ell$ is still a valid upper bound.) □

Proof of Theorem 5.1. For simplicity of presentation we assume that T has the form $T = 2^{d+1} - 1$, as required by Definition 5.2; for other T, we can use a generalized PoSW topology as described in Remark 5.3.

The matrix $\bar{\mathbf{A}}_T$ from Equation (2.3) is recursively n-admissible with PoSW topology of depth $d = \lfloor \log_2 T \rfloor$ and base rows n. The malicious prover computes an n-subspace trapdoor $\bar{\mathbf{R}}_T$ of $\bar{\mathbf{A}}_T$ following Lemma 5.4, uses the subspace-trapdoor solver (see Section 3.1) to get a solution $\mathbf{x} = \bar{\mathbf{R}}_T \cdot \mathbf{G}_n^{-1}(\mathbf{u}_0)$ to Equation (2.3), and then proceeds in the same way as the honest prover. By Lemma 5.4, this prover has arithmetic depth $O(d) = O(\log T)$. Then similar to the proof of Theorem 4.1, by the structure of $\bar{\mathbf{A}}_T$, this arithmetic circuit compiles to a Boolean circuit of depth $\tilde{O}_{n,q}(\log T)$.

Note that acting as the honest prover using a valid solution (of any norm) ensures that the verifier's "linear" checks are satisfied, so it just remains to confirm that the verifier's norm checks are also satisfied. By Lemma 5.4 again, for all $\tau \in [0, d]$ the block-wise norm of the solution $\mathbf{x}$ satisfies

$$\mathsf{mid}(\mathsf{fold}_\gamma^{(\tau)}(\|\mathbf{x}\|_\infty)) \le \mathsf{mid}(\mathsf{fold}_\gamma^{(\tau)}(\|\mathbf{R}\|_{\infty\leftarrow\infty})) \cdot 1 \le (\gamma + m)^\tau \cdot m \le (2\gamma')^\tau \cdot \beta \, .$$

So, considering Equation (5.4) and the fact that the verifier's challenges c all satisfy $|c| \le \gamma$ (or more precisely, $\|c\|_{\infty\leftarrow\infty} \le \gamma$), the verifier's norm checks on the folded solution $\mathbf{x}$ itself are indeed satisfied. □

5.3 Breaking a PoSW with the Original Norm Bounds

The prior subsection breaks the PoSW with tweaked folding operation and relaxed norm bounds. Here we break a PoSW whose only modification from [LM23] is the tweaked folding operation (the norm bounds remain unchanged). This attack uses much more sophisticated recursive strategy, which we summarize below, and present formally in Figure 3; see also Figure 4 for a visual illustration. Our main result in this subsection is as follows; as in the previous subsection, we prove it below after introducing the needed tools.

Theorem 5.5 (Attack on the PoSW with original norm bounds). *For $\log q = o(n)$, there is a depth-$\tilde{O}_{n,q}(\log^2 T)$ malicious prover (modeled as Boolean circuit) that, given any $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n\times m}$, $\mathbf{u}_0 \in \mathbb{Z}_q^n$, and positive integer T, convinces the verifier in the modified PoSW where in the folding operation the challenge is multiplied by the first half of the solution, and the original parameters $\gamma' = \gamma$ and $\beta = 1$ are used.*

Recall that our previous attack breaks the PoSW with parameters $\gamma' = (\gamma + m)/2$ and $\beta = m$ (Theorem 5.1). Our goal in the improved attack is to handle $\gamma' = \gamma$ and $\beta = 1$. At a high level, we improve the parameter β by applying the "direct solution" technique from Section 3.2.2, and we improve γ' by designing a more sophisticated block decomposition of $\bar{\mathbf{A}}_T$ (in a finer-grained version of the PoSW topology).

Obtaining $\beta = 1$. To see the idea behind our new strategy, consider the concrete example of (the core recursive part of) the PoSW for $T = 15$. In the first round, the prover sends the "middle" component $\mathbf{x}_8$ to the verifier, which checks that $\|\mathbf{x}_8\|_\infty \le \beta$. Our previous attack constructs the solution as $\mathbf{x} = \bar{\mathbf{R}}_T \cdot \mathbf{G}_n^{-1}(\mathbf{u}_0)$, where $\bar{\mathbf{R}}_T$ is a n-subspace trapdoor for $\bar{\mathbf{A}}_T$ generated by $\mathsf{RecTrap}$. As a result, our previous attack can ensure only that $\|\mathbf{x}_8\|_\infty \le \|\mathbf{R}_8\|_{\infty\leftarrow\infty}$ (where $\mathbf{R}_8$ is the corresponding row block of $\bar{\mathbf{R}}_T$), and by construction, this is bounded by $\|\mathbf{I}\|_{\infty\leftarrow\infty} \cdot m = m$, which is why we took $\beta = m$.

The factor m above comes from Lemma 3.4, but notice that in its "direct solution" counterpart Lemma 3.6, the expansion factor is $m^{1/\infty} = 1$ instead of m. So, if $\mathbf{x}$ is constructed using $\mathsf{CombSolve}$ instead, we would have $\|\mathbf{x}_8\|_\infty \le 1$, which allows us to take $\beta = 1$, at least for the first round. To get an attack that works for every round, we apply this idea recursively, solving for the first half of $\mathbf{x}$ recursively, and *in parallel* computing suitable trapdoor(s) for the second-half block. We then use $\mathsf{CombSolveHelper}$ to construct an entire solution from these pieces. See the definition of $\mathsf{RecSolve}^{\mathsf{vec}}$ in Figure 3 for the precise definition; also

$\mathsf{RecSolve}_s^{\text{vec}}(\mathbf{A}, \mathbf{u})$

if $\mathbf{A} = \mathbf{G}_n$: **return** $\mathsf{CombSolve}(\mathbf{A}, \mathbf{u}, [\mathbf{I}_{m,s\ell}])$
in parallel:
  $\mathbf{x}_0 = \mathsf{RecSolve}_s^{\text{vec}}(\mathbf{A}_0, \mathbf{u}_0)$
  $L = \mathsf{RecSolve}_s^{\text{list}}(\mathbf{A}_2)$
return $\mathsf{CombSolveHelper}(\mathbf{A}, \mathbf{x}_0, \mathbf{u}', [\mathbf{I}_{m,s\ell}; L])$

$\mathsf{RecSolve}_s^{\text{trap}}(\mathbf{A})$

if $\mathbf{A} = \mathbf{G}_n$: **return** $\mathbf{I}_{m,s\ell}$
in parallel:
  $\mathbf{R}_0 = \mathsf{RecSolve}_s^{\text{trap}}(\mathbf{A}_0)$
  $L = \mathsf{RecSolve}_s^{\text{list}}(\mathbf{A}_2)$
return $\mathsf{CombTrap}_s(\mathbf{A}, [\mathbf{R}_0, \mathbf{I}_{m,s\ell}; L])$

$\mathsf{RecSolve}_s^{\text{list}}(\mathbf{A})$

if $\mathbf{A} = \mathbf{G}_n$: **return** $[\mathbf{I}_{m,s\ell}]$
in parallel:
  $L = \mathsf{RecSolve}_s^{\text{list}}(\mathbf{A}_0)$
  $\mathbf{R}_2 = \mathsf{RecSolve}_s^{\text{trap}}(\mathbf{A}_2)$
return $[L; \mathbf{I}_{m,s\ell}, \mathbf{R}_2]$

Figure 3: Algorithm $\mathsf{RecSolve}^{\text{vec}}$ that solves $\mathbf{Ax} = \mathbf{u}$ for any recursively s-admissible matrix $\mathbf{A}$ having PoSW topology with base rows n, and any s-admissible syndrome $\mathbf{u}$. It parses $\mathbf{u} = \binom{\mathbf{u}_0}{\mathbf{u}'}$ where $\mathbf{u}_0 \in \mathbb{Z}_q^{N_0}$ for $N_0 = (N - n)/2$. Throughout, $m = n\ell$.

see Figure 4 for an illustration, where the solid lines represent recursive calls to $\mathsf{RecSolve}^{\text{vec}}$ (ignore the other types of lines for now).

Obtaining $\gamma' = \gamma$. Now let us resume the example. After the norm check in the first round, the prover receives a challenge c_1 from the verifier. Then in the second round, the prover sends $\mathbf{x}'_4 = c_1 \cdot \mathbf{x}_4 + \mathbf{x}_{12}$ to the verifier, which tests whether $\|\mathbf{x}'_4\|_\infty \leq 2\gamma' \cdot \beta$. Once the above "direct solution" idea is applied recursively, we can similarly get $\|\mathbf{x}_4\|_\infty \leq 1$. However, the $\|\mathbf{x}_{12}\|_\infty$ term in the second half still picks up an m factor when the trapdoor for the second-half block is constructed using $\mathsf{CombTrap}$. This results in no improvement to the parameter $\gamma' = (\gamma + m)/2$.

Our key new idea here is to partly "sequentialize" the solving within the second half: instead of constructing a combined, longer trapdoor for the entire second-half block from trapdoors for its sub-blocks, we just directly use those trapdoors; see $\mathsf{RecSolve}^{\text{list}}$ in Figure 3. (Recall that $\mathsf{CombSolveHelper}$ works for an arbitrary block decomposition.) Specifically, the sub-blocks in the second-half block are the third-quarter block, a singleton block $\mathbf{G}_n$, and the last-quarter block. We then simply use this list of trapdoors in $\mathsf{CombSolveHelper}$ to get the final solution $\mathbf{x}$. This leads to $\|\mathbf{x}_{12}\|_\infty \leq 1$, thus allowing the use of $\gamma' = \gamma$ in the second round.

Continuing the example, we see that the "$(2^\tau + 1)/2^{\tau+1}$-points" (e.g., the "3/4-point" $\mathbf{x}_{12}$ we just considered, the "5/8-point" $\mathbf{x}_{10}$, etc.) are the only places where we need to apply the "sequentialization" idea in order to obtain $\gamma' = \gamma$. Trapdoors for other groups of sub-blocks can be safely combined into one trapdoor without violating the norm bounds; see $\mathsf{RecSolve}^{\text{trap}}$ in Figure 3. Importantly, therefore, the lengths of the lists of trapdoors remain linear in the recursion depth d, so the depths of the sequential solving steps remain low. See Figure 4 for an illustration of the entire example, where the thick and thin dashed lines represent recursive calls to $\mathsf{RecSolve}^{\text{list}}$ and $\mathsf{RecSolve}^{\text{trap}}$, respectively; note that these recursive calls alternate in a way that exactly traces the "$(2^\tau + 1)/2^{\tau+1}$-points." Altogether, combining the "direct solution" and the "sequentialization" idea, we manage to get our new attack to work for the original parameters $\gamma' = \gamma$ and $\beta = 1$.

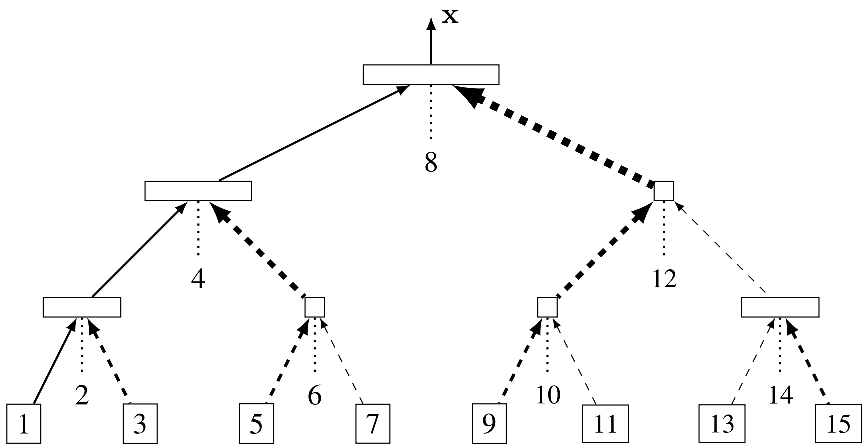

Figure 4: An example recursion tree for $\mathsf{RecSolve}^{\text{vec}}$ on a matrix with PoSW topology of depth $d = 3$. Recursive calls to RecSolve in the "modes" vec, list, trap are represented by solid lines, thick dashed lines, and thin dashed lines, respectively; singleton identity matrices are marked by thin dotted lines. The thickness of each (thick dashed) line for a list-mode call roughly depicts the number of returned trapdoors, and the width of each internal recursion node roughly depicts the amount of sequential work at that node. Each number i marks the vector/trapdoor corresponding to the block $\mathbf{x}_i$ in the solution $\mathbf{x}$.

Lemma 5.6. *Suppose that $s\ell \le \gamma^2$. Let $\mathbf{A}$ be a recursively s-admissible matrix with PoSW topology of depth d and base rows n, and let $\mathbf{u}$ be an s-admissible vector. There is a depth-$O(d^2)$ arithmetic circuit $\mathsf{RecSolve}_s^{vec}$ that, given $\mathbf{A}$, computes a solution to $\mathbf{A}\mathbf{x} = \mathbf{u}$ satisfying the following bound for any $p \in [1, \infty]$ and any integer $\tau \in [0, d]$:*

$$\mathsf{mid}(\mathsf{fold}_\gamma^{(\tau)}(\|\mathbf{x}\|_p)) \le (2\gamma)^\tau \cdot (s\ell)^{1/p} .$$

In particular, for $p = \infty$, we have that $\mathsf{mid}(\mathsf{fold}_\gamma^{(\tau)}(\|\mathbf{x}\|_\infty)) \le (2\gamma)^\tau$.

Remark 5.7. In Lemma 5.6 we assume that $s\ell \le \gamma^2$. It is possible to relax the assumption to $s\ell \le \gamma^{2+a}$ for any integer $a \ge 0$, at the cost of increasing the arithmetic depth from $O(d^2)$ to $O(d^{2+a})$. Note that for any polynomial relationship between $s\ell$ and γ, this a will be a constant and so the extra factor d^a in the arithmetic depth will be polynomial in the depth of the PoSW topology.

The generalization is to introduce a parameter α in $\mathsf{RecSolve}^{\text{list}}$ and form $\mathsf{RecSolve}^{\text{list}}[\alpha]$. $\mathsf{RecSolve}^{\text{vec}}$ and $\mathsf{RecSolve}^{\text{trap}}$ now make recursive calls to $\mathsf{RecSolve}^{\text{list}}[a]$ instead. For $\alpha = 0$, $\mathsf{RecSolve}^{\text{list}}[0]$ makes recursive calls to $\mathsf{RecSolve}^{\text{list}}[0]$ and $\mathsf{RecSolve}^{\text{trap}}$ as before, while for $\alpha > 0$, $\mathsf{RecSolve}^{\text{list}}[\alpha]$ makes recursive calls to $\mathsf{RecSolve}^{\text{list}}[\alpha]$ and $\mathsf{RecSolve}^{\text{list}}[\alpha - 1]$ instead (and combines by simply concatenating the returned lists, still with an $\mathbf{I}_{m,s\ell}$ in the middle).

Proof. Again we assume that T has the form $T = 2^{d+1} - 1$; the case of general T can be handled as described in Remark 5.3.

We first bound the arithmetic depth of RecSolve, in any of its three "modes" (vec, trap, list), when given a recursively s-admissible matrix having PoSW topology of depth d. It makes two *parallel* recursive calls (with varying modes) on submatrices that follow the PoSW topology (with the same base case), so its total recursion depth is d. Then because $\mathsf{RecSolve}^{\text{trap}}$ returns a single trapdoor, $\mathsf{RecSolve}^{\text{list}}$ returns a list having $O(d)$ trapdoors. So, by Lemmas 3.4 and 3.6, the arithmetic depth of the non-recursive work in RecSolve—namely, CombSolveHelper or CombTrap—is $O(d)$, and hence the overall arithmetic depth is $O(d^2)$.

For the (block-wise) norm of $\mathbf{x}$, we analyze more generally the norms of the outputs of $\mathsf{RecSolve}^{\{\text{vec,list,trap}\}}$ when given an arbitrary recursively s-admissible matrix with PoSW topology. For an (arbitrary) vector $\mathbf{x}$ returned by $\mathsf{RecSolve}^{\text{vec}}$, again by Lemma 3.6, we have the recurrence

$$\|\mathbf{x}\|_p \leq \begin{pmatrix} \|\mathbf{x}_0\|_p \\ \|\mathbf{I}_{m,s\ell}\|_{p\leftarrow p} \cdot (s\ell)^{1/p} \\ \|\hat{\mathbf{R}}_2\|_{p\leftarrow p} \cdot (s\ell)^{1/p} \end{pmatrix} = \begin{pmatrix} \|\mathbf{x}_0\|_p & \\ & (s\ell)^{1/p} \\ \|\hat{\mathbf{R}}_2\|_{p\leftarrow p} \cdot (s\ell)^{1/p} & \end{pmatrix} ,$$

and the base case is $\|\mathbf{x}\|_p = \|\mathbf{G}_s^{-1}(\star)\|_p \leq (s\ell)^{1/p}$. Here $\hat{\mathbf{R}}_2$ is the block matrix whose row blocks are the vertically stacked s-subspace trapdoors (which, to recall, all have $s\ell$ columns) from the list L returned by $\mathsf{RecSolve}^{\text{list}}$; we similarly stack other outputs of $\mathsf{RecSolve}^{\text{list}}$ below. For a trapdoor stack $\hat{\mathbf{R}}$ returned by $\mathsf{RecSolve}^{\text{list}}$, straightforwardly,

$$\|\hat{\mathbf{R}}\|_{p\leftarrow p} = \begin{pmatrix} \|\hat{\mathbf{R}}_0\|_{p\leftarrow p} \\ \|\mathbf{I}_{m,s\ell}\|_{p\leftarrow p} \\ \|\mathbf{R}_2\|_{p\leftarrow p} \end{pmatrix} = \begin{pmatrix} \|\hat{\mathbf{R}}_0\|_{p\leftarrow p} \\ 1 \\ \|\mathbf{R}_2\|_{p\leftarrow p} \end{pmatrix} ;$$

and for a trapdoor $\mathbf{R}$ returned by $\mathsf{RecSolve}^{\text{trap}}$, again by Lemma 3.4, we have that

$$\|\mathbf{R}\|_{p\leftarrow p} \leq \begin{pmatrix} \|\mathbf{R}_0\|_{p\leftarrow p} \\ \|\mathbf{I}_{m,s\ell}\|_{p\leftarrow p} \cdot s\ell \\ \|\hat{\mathbf{R}}_2\|_{p\leftarrow p} \cdot s\ell \end{pmatrix} = \begin{pmatrix} \|\mathbf{R}_0\|_{p\leftarrow p} & \\ & s\ell \\ \|\hat{\mathbf{R}}_2\|_{p\leftarrow p} \cdot s\ell & \end{pmatrix} \leq \begin{pmatrix} \|\mathbf{R}_0\|_{p\leftarrow p} & \\ & \gamma^2 \\ \|\hat{\mathbf{R}}_2\|_{p\leftarrow p} \cdot \gamma^2 & \end{pmatrix} ,$$

where for the last inequality we use the hypothesis $s\ell \leq \gamma^2$. Both $\|\hat{\mathbf{R}}\|_{p\leftarrow p}$ and $\|\mathbf{R}\|_{p\leftarrow p}$ have the same base case $\|\mathbf{I}_{m,s\ell}\|_{p\leftarrow p} = 1$.

So, following the recurrence pattern, when the input to $\mathsf{RecSolve}$ has PoSW topology of depth d, we have that $\begin{pmatrix}\|\mathbf{R}\|_{p\leftarrow p} & \|\hat{\mathbf{R}}\|_{p\leftarrow p} & \|\mathbf{x}\|_p\end{pmatrix}$ is bounded from above by the matrix $\begin{pmatrix}\mathsf{pf}_{\text{trap}}(d) & \mathsf{pf}_{\text{list}}(d) & \mathsf{pf}_{\text{vec}}(d)\end{pmatrix}$, which is recursively defined as follows:

$$\begin{pmatrix}\mathsf{pf}_{\text{trap}}(d) & \mathsf{pf}_{\text{list}}(d) & \mathsf{pf}_{\text{vec}}(d)\end{pmatrix} := \begin{pmatrix} \mathsf{pf}_{\text{trap}}(d-1) & \mathsf{pf}_{\text{list}}(d-1) & \mathsf{pf}_{\text{vec}}(d-1) & \\ \gamma^2 & 1 & & (s\ell)^{1/p} \\ \mathsf{pf}_{\text{list}}(d-1)\cdot\gamma^2 & \mathsf{pf}_{\text{trap}}(d-1) & \mathsf{pf}_{\text{list}}(d-1)\cdot(s\ell)^{1/p} & \end{pmatrix} .$$

For convenience, we denote $\mathsf{profile}(d) := \begin{pmatrix}\mathsf{pf}_{\text{trap}}(d) & \mathsf{pf}_{\text{list}}(d) & \mathsf{pf}_{\text{vec}}(d)\end{pmatrix}$. The base case of the recursion is $\mathsf{profile}(0) = \begin{pmatrix}1 & 1 & (s\ell)^{1/p}\end{pmatrix}$.

From these definitions it can be verified that

$$\begin{aligned} \mathsf{fold}_\gamma(\mathsf{profile}(d)) &= \begin{pmatrix}\gamma\mathbf{I} & \mathbf{0} & \mathbf{I}\end{pmatrix} \cdot \begin{pmatrix} \mathsf{pf}_{\text{trap}}(d-1) & \mathsf{pf}_{\text{list}}(d-1) & \mathsf{pf}_{\text{vec}}(d-1) & \\ \gamma^2 & 1 & & (s\ell)^{1/p} \\ \mathsf{pf}_{\text{list}}(d-1)\cdot\gamma^2 & \mathsf{pf}_{\text{trap}}(d-1) & \mathsf{pf}_{\text{list}}(d-1)\cdot(s\ell)^{1/p} & \end{pmatrix} \\ &= \mathsf{profile}(d-1) \cdot \begin{pmatrix} \gamma & 1 & 0 \\ \gamma^2 & \gamma & (s\ell)^{1/p} \\ 0 & 0 & \gamma \end{pmatrix} . \end{aligned}$$

Hence by iterating, for any τ, we get that

$$\mathsf{fold}_\gamma^{(\tau)}(\mathsf{profile}(d)) = \mathsf{profile}(d-\tau) \cdot \begin{pmatrix} \gamma & 1 & 0 \\ \gamma^2 & \gamma & (s\ell)^{1/p} \\ 0 & 0 & \gamma \end{pmatrix}^\tau .$$

It can be verified by induction that this matrix power expands to

$$\begin{pmatrix} \gamma & 1 & 0 \\ \gamma^2 & \gamma & (s\ell)^{1/p} \\ 0 & 0 & \gamma \end{pmatrix}^{\tau} = \begin{pmatrix} (2\gamma)^{\tau-1}\cdot\gamma & (2\gamma)^{\tau-1} & (2^{\tau-1}-1)\cdot\gamma^{\tau-2}\cdot(s\ell)^{1/p} \\ (2\gamma)^{\tau-1}\cdot\gamma^2 & (2\gamma)^{\tau-1}\cdot\gamma & (2\gamma)^{\tau-1}\cdot(s\ell)^{1/p} \\ 0 & 0 & \gamma^{\tau} \end{pmatrix} .$$

Then we get, as desired,

$$\begin{aligned} \mathsf{mid}(\mathsf{fold}_{\gamma}^{(\tau)}(\|\mathbf{x}\|_p)) &\leq \mathsf{mid}(\mathsf{fold}_{\gamma}^{(\tau)}(\mathsf{pf}_{\mathsf{vec}}(d))) \\ &= \mathsf{mid}(\mathsf{profile}(d-\tau)) \cdot \begin{pmatrix} (2^{\tau-1}-1)\cdot\gamma^{\tau-2}\cdot(s\ell)^{1/p} \\ (2\gamma)^{\tau-1}\cdot(s\ell)^{1/p} \\ \gamma^{\tau} \end{pmatrix} \\ &\leq \begin{pmatrix} \gamma^2 & 1 & (s\ell)^{1/p} \end{pmatrix} \cdot \begin{pmatrix} (2^{\tau-1}-1)\cdot\gamma^{\tau-2}\cdot(s\ell)^{1/p} \\ (2\gamma)^{\tau-1}\cdot(s\ell)^{1/p} \\ \gamma^{\tau} \end{pmatrix} \\ &= 2^{\tau-1}(\gamma^{\tau}+\gamma^{\tau-1})\cdot(s\ell)^{1/p} \\ &\leq (2\gamma)^{\tau}\cdot(s\ell)^{1/p} . \end{aligned}$$

Here note that $\mathsf{mid}(\mathsf{pf}(d-\tau)) = \begin{pmatrix} 1 & 1 & (s\ell)^{1/p} \end{pmatrix}$ in the base case $\tau = d$, but $\begin{pmatrix} \gamma^2 & 1 & (s\ell)^{1/p} \end{pmatrix}$ is still a valid upper bound. □

Proof of Theorem 5.5. Because $\log q = o(n)$, we have that $m = n\ell = o(n^2) = o(\gamma^2)$, and thus $m \leq \gamma^2$ holds (for all sufficiently large n). Recall that the matrix $\bar{\mathbf{A}}_T$ from Equation (2.3) is recursively n-admissible with PoSW topology of depth $d = \lfloor \log_2 T \rfloor$ and base rows n. The malicious prover computes a solution $\mathbf{x}$ to Equation (2.3) following Lemma 5.6, and then proceeds in the same way as the honest prover. By Lemma 5.6, this prover has arithmetic depth $O(d^2) = O(\log^2 T)$. Similarly to the proof of Theorem 4.1, by the structure of $\bar{\mathbf{A}}_T$, this compiles to a Boolean circuit of depth $\tilde{O}_{n,q}(\log^2 T)$.

It remains to confirm that this will satisfy the verifier's norm checks, and in particular to analyze the γ-folding of the block-wise norms of the solution $\mathbf{x}$. Similar to the proof of Theorem 5.1, by Lemma 5.6 again, for all $\tau \in [0, d]$ the block-wise norm of the solution $\mathbf{x}$ satisfies

$$\mathsf{mid}(\mathsf{fold}_{\gamma}^{(\tau)}(\|\mathbf{x}\|_{\infty})) \leq (2\gamma)^{\tau}\cdot 1 ,$$

so the verifier's norm checks are indeed satisfied with bound $(2\gamma')^{\tau}\cdot\beta$ for parameters $\gamma' = \gamma$ and $\beta = 1$. □

References

[Ajt96] M. Ajtai. Generating hard instances of lattice problems. *Quaderni di Matematica*, 13:1–32, 2004. Preliminary version in STOC 1996. Pages 2 and 16.

[AKK+19] H. Abusalah, C. Kamath, K. Klein, K. Pietrzak, and M. Walter. Reversible proofs of sequential work. In *EUROCRYPT*, pages 277–291. 2019. Page 1.

[CLSY93] J. Cai, R. J. Lipton, R. Sedgewick, and A. C. Yao. Towards uncheatable benchmarks. In *Structure in Complexity Theory Conference*, pages 2–11. 1993. Page 1.

[CP18] B. Cohen and K. Pietrzak. Simple proofs of sequential work. In *EUROCRYPT*, pages 451–467. 2018. Pages 1 and 2.

[DN92] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *CRYPTO*, volume 740, pages 139–147. 1992. Page 2.

[LM23] R. W. F. Lai and G. Malavolta. Lattice-based timed cryptography. In *CRYPTO*, pages 782–804. 2023. Pages 2, 3, 5, 6, 7, 8, 9, 12, 13, 14, 15, 16, 17, and 19.

[May93] T. C. May. Timed-release crypto, February 1993. `http://cypherpunks.venona.com/date/1993/02/msg00129.html`. Page 1.

[MMV13] M. Mahmoody, T. Moran, and S. P. Vadhan. Publicly verifiable proofs of sequential work. In *Innovations in Theoretical Computer Science (ITCS)*, pages 373–388. 2013. Pages 1 and 2.

[MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718. 2012. Pages 5 and 6.

[MR04] D. Micciancio and O. Regev. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.*, 37(1):267–302, 2007. Preliminary version in FOCS 2004. Page 16.

[Pie19] K. Pietrzak. Simple verifiable delay functions. In *Innovations in Theoretical Computer Science Conference (ITCS)*, volume 124 of *LIPIcs*, pages 60:1–60:15. 2019. Page 2.

[Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):1–40, 2009. Preliminary version in STOC 2005. Page 2.

[RSW96] R. L. Rivest, A. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1996. Pages 1 and 2.

[Sho94] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, 1997. Preliminary version in FOCS 1994. Page 2.

[Ver12] R. Vershynin. *Introduction to the non-asymptotic analysis of random matrices*, chapter 5, pages 210–268. Cambridge University Press, 2012. Available at `http://www-personal.umich.edu/~romanv/papers/non-asymptotic-rmt-plain.pdf`. Page 4.

[Wes19] B. Wesolowski. Efficient verifiable delay functions. *J. Cryptol.*, 33(4):2113–2147, 2020. Preliminary version in EUROCRYPT 2019. Page 2.