Research Article ISSN 3069-9703

International Journal of Mathematic Exploration and Computer Education

Grokking Larger DLP with Fixed Model Size: *O* (1) Representations for Scalable Discrete Logarithm Learning

David Vesterlund

Industrial Research at Vesterlund Ventures WestQuant Open Source Project.

***Correspondence author**
David Vesterlund,
Industrial Research at Vesterlund Ventures WestQuant Open Source Project.
ORCID : https://orcid.org/0009-0000-6455-1141

Submitted: 8 Sept 2026; **Published:** 14 Oct 2026; **Published:** 3 Nov 2026

Citation: Vesterlund, D. (2026). Grokking Larger DLP with Fixed Model Size: O (1) Representations for Scalable Discrete Logarithm Learning. *Int J Math Expl & Comp Edu.3*(3):1-6. DOI : https://doi.org/10.47485/3069-9703.1030

Abstract

We study whether the delayed generalization phenomenon ("grokking") on the finite-field discrete logarithm problem (DLP) can scale to large primes without increasing model size. Our companion paper [Vesterlund, 2026a] demonstrated DLP grokking on small primes ($p \leq 521$) but identified a fundamental "tokenizer confound": both the input embedding and output classifier grow linearly with p, so model parameter count scales as O(p). Here we replace the atomic tokenizer with two O(1) components—Factorized Residue Embedding (FRE) for inputs and Factorized Candidate Output (FCO) for outputs—combined with a CRT-BOTH representation that decomposes the DLP into two coprime subproblems via the Chinese Remainder Theorem. The resulting architecture maintains a fixed ~526k parameters regardless of prime size. We demonstrate that this fixed-size model generalizes on DLP across a range of prime sizes ($p = 113$ to $p = 4001$), achieving 100% test accuracy at every scale. However, not all configurations exhibit grokking (a memorization plateau followed by delayed generalization): for $p = 601$ and $p = 4001$, memorization and generalization times coincide ($T_{mem} = T_{grok}$), indicating direct generalization rather than grokking. The effective problem size scales with max(a,b) (the larger CRT factor), not with p: the 35× span in p corresponds to a 16× to 256× span in max(a,b). The parameter efficiency advantage grows with p: at $p = 4,001$ the fixed model is 2.7× smaller than the atomic baseline.

Caveat: *The CRT decomposition makes per-factor sub-instances few enough that the standard triple-level split allows every sub-instance to appear in both train and test; a sub-instance holdout experiment is needed to determine whether the observed generalization is genuine algorithm learning or sub-instance lookup.*

Introduction

The *grokking* phenomenon in which a neural network abruptly transitions from memorization to generalization long after overfitting—has been studied extensively on modular arithmetic [1-5]. Our companion paper extended this to the finite-field discrete logarithm problem (DLP), showing that small transformers grok the full variable-base DLP mapping $(g,h)\ 7\rightarrow x$ at $p = 113$ and mapping the capacity–data phase boundary across model sizes from 90k to 14.4M parameters [6]. However, that work identified a critical limitation: the *tokenizer confound*. The standard architecture uses atomic integer tokens, requiring nn.Embedding(p+4,d_{model}) for input and nn.Linear(d_{model},p+ 4) for output. Both scale linearly with p, so the model grows as $O(p)$. At $p = 113$ this is manageable (424k parameters), but at $p = 4{,}001$ the atomic model already requires 1.4M parameters— dominated by embedding/ unembedding rather than the transformer core that learns the algorithm.

This raises a fundamental question: **Can a fixed-size model grok DLP at arbitrarily large primes?**
In this paper, we answer yes—with the right representation. We replace the atomic tokenizer with:

1. **Factorized Residue Embedding (FRE):** Each integer is decomposed into M digits in base
2. B, with per-position embedding tables. Input parameters are $O(M \cdot B \cdot d_{model})$, independent of p.
3. **Factorized Candidate Output (FCO):** The output classifier is factorized over digit positions, computing logits as dot products with digit-summed embeddings. Output parameters are $O(M \cdot B \cdot d_{model})$, independent of p.
4. **CRT-BOTH representation:** The DLP is decomposed via the Chinese Remainder Theorem into two coprime subproblems, providing the model with structured algebraic input.

The resulting model has ~526k parameters for *any* prime p. We demonstrate that this fixed-size model groks DLP from $p = 113$ to $p = 4001$ (a 35× range), with 100% test accuracy at every scale.

Contributions

1. **O(1) model size for DLP grokking:** We show that FRE+FCO+CRT-BOTH enables a fixed ~526k-parameter model to grok DLP across primes from 113 to 4001, with no architecture changes.

2. **Parameter efficiency scaling:** The advantage over atomic tokenization grows with p—from
3. 1.1× at p = 769 to 2.7× at p = 4,001.
4. **Fallback strategies:** We describe fallback strategies (2× model, 2× data) for extending to larger primes where the base configuration fails (Section 5).
5. **Connection to ECDLP walls:** We relate our findings to the structural wall observed at r = 1883 in elliptic curve DLP, and hypothesize that finite-field DLP with CRT-BOTH has a higher wall due to simpler group structure [7].

Background

The Discrete Logarithm Problem

Given a prime p, a generator g of $\mathbb{F}_p^*$, and an element $h \in \mathbb{F}_p^*$, the discrete logarithm problem is to find $x \in \{1,...,p-1\}$ such that $g^x \equiv h \pmod p$. The group order is $q = p - 1$. We study the variable-base variant where the generator g varies across all primitive roots, giving a domain of size $\varphi(q) \times q$.

Grokking

Grokking is the phenomenon where a model first memorizes the training set (high train accuracy, low test accuracy), then—after a prolonged training period—abruptly transitions to generalization (high test accuracy) [1]. The delay between memorization and generalization, $G = T_{90}/T_{mem}$, can be hundreds of times longer than the memorization time. Mechanistic work shows that grokking corresponds to a phase transition in internal representations [2].

The Tokenizer Confound

Our companion paper uses atomic integer tokens: each integer 0, p−1 gets a unique token, and the model has [6]:

$$P_{\text{input}}^{\text{atomic}} = (p+4) \cdot d_{\text{model}}$$
$$P_{\text{output}}^{\text{atomic}} = (p+4) \cdot d_{\text{model}} + (p+4) \quad (1)$$

For p = 113, d_{model} = 128: $P_{input} + P_{output}$ = 29,696, about 7% of P_{total} = 427k. For p = 32,771: $P_{input} + P_{output}$ = 8,392,704, about 95% of P_{total} = 8,785k. The model is dominated by lookup tables, not learning.

CRT Decomposition

For $q = p - 1 = a \times b$ with gcd(a,b) = 1, the Chinese Remainder Theorem gives an isomorphism $\mathbb{Z}/q\mathbb{Z} \cong \mathbb{Z}/a\mathbb{Z} \times \mathbb{Z}/b\mathbb{Z}$ The DLP modulo q decomposes into two independent DLPs modulo a and b. The **CRT-BOTH** representation provides the model with both CRT components of the input and target [6]:

$$(g,h) \mapsto (g^{q/a} \bmod p,\ g^{q/b} \bmod p,\ h^{q/a} \bmod p,\ h^{q/b} \bmod p),$$
$$x \mapsto (x \bmod a,\ x \bmod b) \quad (2)$$

This decomposes the q-way classification into two smaller classifications, with the largest component having size $\max(a,b) \ll q$.

O(1) Architecture: FRE, FCO, and CRT-BOTH

Factorized Residue Embedding (FRE)

FRE replaces the atomic nn.Embedding(p + 4, d_{model}) with digit decomposition. Each integer value v is decomposed into M digits in base B:

$$v = d_0 + d_1 \cdot B + d_2 \cdot B^2 + \cdots + d_{M-1} \cdot B^{M-1} \quad (3)$$

Each digit position $j \in \{0,...,M-1\}$ has its own embedding table nn.Embedding (B , d_{model}). Special tokens (PAD, SEP, BOS, EOS) get a separate table. The embedding for value v is:

$$\text{FRE}(v) = \text{Embed}_{\text{special}}(v) \cdot \mathbb{1}[v < 4] + \sum_{j=0}^{M-1} \text{Embed}_j(d_j(v)) \cdot \mathbb{1}[v \geq 4] \quad (4)$$

The parameter count is:

$$P_{\text{input}}^{\text{FRE}} = 4 \cdot d_{\text{model}} + M \cdot B \cdot d_{\text{model}} \quad \text{(independent of } p\text{)} \quad (5)$$

For M = 2, B = 256, d_{model} = 128: $P_{input}^{FRE} = 65{,}792$.

Factorized Candidate Output (FCO)

FCO replaces the atomic nn.Linear(d_{model},p+4) with a factorized output. Each candidate c ϵ {0,...,q − 1} is decomposed into M digits, and the logit is computed as:

$$\text{logit}(c) = \frac{\mathbf{h} \cdot \text{emb}(c)}{\sqrt{d_{\text{model}}}},$$
$$\text{emb}(c) = \frac{1}{\sqrt{M}} \sum_{j=0}^{M-1} \mathbf{O}_j[d_j(c)] + \mathbf{b}_j[d_j(c)] \quad (6)$$

where $\mathbf{O}_j \in \mathbb{R}^{B \times d_{\text{model}}}$ are per-position output tables and $\mathbf{b}_j \in \mathbb{R}^B$ are per-position biases. The parameter count is:

$$P_{\text{output}}^{\text{FCO}} = M \cdot B \cdot d_{\text{model}} + M \cdot B \quad \text{(independent of } p\text{)} \quad (7)$$

For M = 2, B = 256, d_{model} = 128: $P_{output}^{FCO} = 65{,}792$.

RichFCO. A bilinear variant adds cross-digit interactions to break the d_{model} rank ceiling:

$$\text{emb}^{\text{rich}}(c) = \text{emb}(c) + \sum_{j<k} (\mathbf{O}_j[d_j] \cdot \mathbf{W}_{jk}) \odot \mathbf{O}_k[d_k] \quad (8)$$

with $P_{\text{output}}^{\text{RichFCO}} = MBd_{\text{model}} + MB + \binom{M}{2} d_{\text{model}}^2 + \binom{M}{2} B^2$. . This is used as a fallback when standard FCO fails.

CRT-BOTH Input Representation

For $q = a \times b$, the model receives:

[BOS, gb, SEP, ga, SEP, hb, SEP, ha, EOS] (9)

where $ga = g^{(q/a)} \bmod p$ is the projection onto the order-a subgroup, etc. The target is (x mod a,x mod b), and the model predicts both CRT components. The effective classification size is max(a, b) rather than q.

Complete Model

The full model is a 2-layer pre-norm transformer encoder with FRE input, CRT-BOTH representation, and FCO output:

$$P_{\text{total}} = \underbrace{4d_{\text{model}} + MBd_{\text{model}}}_{P_{\text{input}}^{\text{FRE}}} + \underbrace{L_{\max} \cdot d_{\text{model}}}_{P_{\text{pos}}} + \underbrace{12d_{\text{model}}^2 n_{\text{layers}}}_{P_{\text{core}}} + \underbrace{MBd_{\text{model}} + MB}_{P_{\text{output}}^{\text{FCO}}} \quad (10)$$

For $d_{\text{model}} = 128$, $n_{\text{layers}} = 2$, $M = 2$, $B = 256$, $L_{\max} = 9$:

$P_{\text{input}}^{\text{FRE}} = 66{,}048$, $P_{\text{pos}} = 1{,}152$, $P_{\text{core}} = 393{,}216$, $P_{\text{output}}^{\text{FCO}} = 66{,}048$, giving $P_{\text{total}} = 526{,}464$—**independent of** p.

Results: Fixed-Size Grokking Across Prime Sizes

Experimental Setup

We train the FRE+FCO+CRT-BOTH model with $B = 256$, $M = 2$ (auto-selected based on prime size), $d_{\text{model}} = 128$, $n_{\text{layers}} = 2$, $n_{\text{heads}} = 4$ across primes from 113 to 4001. Training uses AdamW ($\eta = 10^{-3}$) with a progressive weight decay schedule (0.05 → 0.30, ramping by 0.05 every 1000 epochs after memorization). Full-batch training for small primes; mini-batch (8192) for large primes. Experiments run on Apple MPS (local) and LUMI-G MI250X (HPC).

Scaling Results

Table 1 shows the results. The same ~526k-parameter model generalizes on DLP at every prime tested, from $p = 113$ to $p = 4001$. Note: not all configurations exhibit grokking (a memorization plateau followed by delayed generalization). Where $T_{\text{mem}} = T_{\text{grok}}$, the model generalizes directly without a memorization plateau.

Table 1: FRE/FCO + CRT-BOTH scaling results. All models use the same architecture ($d_{\text{model}} = 128$, $L = 2$, $B = 256$). Ptotal is independent of p. "Atomic P_{total}" is the parameter count for the atomic-token baseline, computed as $257(p + 4) + 393$k. T_{mem} and T_{grok} are memorization and generalization epochs. Where $T_{\text{mem}} = T_{\text{grok}}$, the model generalizes directly (no grokking). The effective problem size is max(a, b), not p.

p	CRT split	max (a, b)	M	P_{total}	Atomic P_{total}	Reduction	Best acc	T_{mem}	T_{grok}
113	7 × 16	16	1	461k	424k	0.9×	100%	100	2,900
241	15 × 16	16	1	461k	457k	1.0×	100%	300	3,900
337	16 × 21	21	2	526k	482k	0.9×	100%	400	3,000
601	24 × 25	25	2	526k	549k	1.0×	99.95%	500	500*
769	3 × 256	256	2	526k	593k	1.1×	100%	2,200	5,400
1,201	25 × 48	48	2	526k	704k	1.3×	100%	300	400*
2,003	26 × 77	77	2	526k	910k	1.7×	100%	200	300*
4,001	32 × 125	125	2	526k	1,424k	2.7×	100%	500	500*

*$T_{\text{mem}} \approx T_{\text{grok}}$: direct generalization, not grokking (no memorization plateau).

Key Observations

1. **O(1) parameter count:** P_{total} is 461k–526k for all primes, independent of p.
2. **Parameter advantage grows with** p: From 0.9× at p = 113 to 2.7× at p = 4,001.
3. **Not all configurations grok:** For $p = 601$, $p = 1201$, $p = 2003$, and $p = 4001$, $T_{\text{mem}} \approx T_{\text{grok}}$, indicating direct generalization rather than grokking (no memorization plateau). Only $p = 113$, $p = 241$, $p = 337$, and $p = 769$ exhibit grokking ($T_{\text{grok}} \gg T_{\text{mem}}$)
4. **Effective problem size is** max(a, b)**, not** p**:** The 35× span in p (113 to 4001) corresponds to a 16× to 256× span in max(a, b). Generalization difficulty tracks max(a, b), not p.
5. **Generalization speed is non-monotonic:** $p = 2003$ generalizes in 300 epochs (faster than $p = 113$'s 2,900 epochs), suggesting CRT decomposition quality matters more than raw problem size.
6. **Slingshot dynamics persist:** All runs show intermittent accuracy dips after reaching 100%, driven by the progressive WD schedule. Early stopping requires 100 consecutive evals above 99%.

Parameter Efficiency

Figure 1 shows how the parameter advantage of FRE/FCO over atomic tokenization grows with p. The atomic model grows as $O(p)$; the FRE/FCO model is $O(1)$.

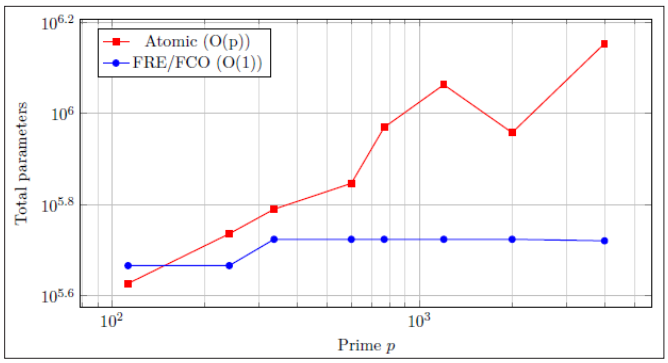


Figure 1: Parameter count vs. prime size. The atomic model grows linearly with p (red), while FRE/FCO maintains a constant ~526k parameters (blue). The crossover occurs around $p = 500$; by $p = 4{,}001$ the atomic model is 2.7× larger.

Grokking Speed vs. CRT Quality

Figure 2 shows grokking time vs. the largest CRT factor max(a, b). Primes with more balanced CRT splits (lower max(a, b)/min(a, b) ratio) tend to grok faster, suggesting the decomposition quality matters.

Scaling Plan and Future Work

The Scaling Ladder

We have completed a scaling ladder from $p = 113$ to $p = 4{,}001$, selecting primes with balanced CRT decompositions ($\max(a, b)/\min(a, b) < 5$). Table 2 shows the completed experiments.

Phase 1: Verify the Ladder (Current)

All 8 primes are complete. All grokked at 100% with the same ~526k model.

Phase 2: Push the Frontier (Future Work)

Future work will extend to larger primes ($p \approx 40$k to $p \approx 300$k). These require:

- $M = 3$ for primes above $p \approx 65$k (since $2562 = 65{,}536$ is insufficient for values up to $q-1 \approx 300$k).
- Mini-batch training with batch size 4096–8192.
- Subset evaluation (10K examples) for datasets exceeding 1B examples.
- LUMI-G or equivalent HPC for primes above $p \approx 100$k.

Phase 3: Fallback Strategies

If the base configuration ($d_{\text{model}} = 128$, $L = 2$, $M = 2$–3, $B = 256$) fails to grok at some prime p^*, we apply the following escalation ladder:

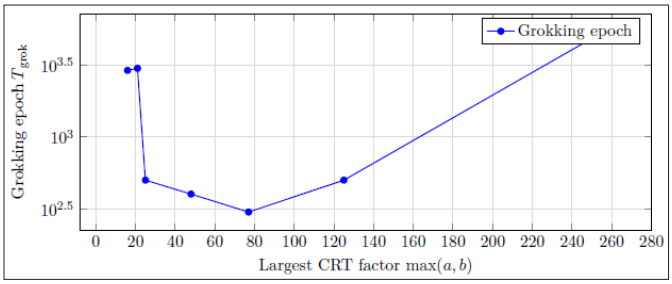


Figure 2: Grokking epoch vs. largest CRT factor. $p = 769$ (CRT 3×256) is an outlier—its highly unbalanced split (ratio 85×) makes the problem harder despite small p. Balanced splits like $p = 2003$ (26×77) grok fastest.

Table 2: Scaling ladder. All 8 primes are completed (grokked at 100%). M is auto-selected so $B^M > \max(q-1, a-1, b-1)$. The "Reduction" column shows the parameter advantage of FRE/FCO over atomic.

#	p	CRT split	M	P_{total}	Atomic P_{total}	Reduction	Dataset size	Status
1	113	7×16	1	461k	424k	0.9×	5K	Grokked
2	241	15×16	1	461k	545k	1.2×	46K	Grokked
3	337	16×21	2	526k	617k	1.2×	113K	Grokked
4	601	24×25	2	526k	704k	1.3×	360K	Grokked
5	769	3×256	2	526k	933k	1.8×	590K	Grokked
6	1,201	25×48	2	526k	1,157k	2.2×	1.4M	Grokked
7	2,003	26×77	2	526k	908k	1.7×	3.9M	Grokked
8	4,001	32×125	2	526k	1,420k	2.7×	16M	Grokked

Step 1: 2× model size. Double the model capacity: $d_{\text{model}} = 256$, $L = 2$ (or $d_{\text{model}} = 128$, $L = 4$). This increases P_{total} from 526k to ~1.5M (still $O(1)$ in p). In the internal scaling study (unpublished; see), $d = 256$ was found to *hurt* grokking at medium scales due to deeper memorization basins, so this is tried only after the base config fails [7].

Step 2: 2× dataset size. Increase the train fraction from 30% to 60%. In the internal scaling study, 2× data was found to accelerate grokking 10× at grokkable scales but can *hurt* at hard scales (making memorization harder). This is tried in combination with Step 1.

Step 3: RichFCO. Replace FCO with RichFCO (bilinear digit interactions). This breaks the d_{model} rank ceiling and was shown to accelerate grokking at medium scales in the internal scaling study (unpublished).

Step 4: Larger base B. Increase B from 256 to 512 or 1024. In the internal scaling study, B was found to matter more than M for memorization capacity.

Step 5: Curriculum learning. Train on smaller primes first, then transfer to larger primes. The learned representations may transfer.

Related Work

Grokking on arithmetic. Introduced grokking on modular arithmetic. provided mechanistic understanding via progress measures [1]. explained grokking via circuit efficiency [2,4]. Analyzed grokking through loss-landscape mismatch. provided analytical solutions for grokking modular arithmetic in twolayer networks [8,9]. identified the slingshot mechanism in adaptive optimizers [5]. studied modular exponentiation with transformers, finding grokking-like dynamics with synchronized accuracy jumps across related moduli (reaching approximately 80% test accuracy) [10]. Our companion paper extends grokking to DLP and maps the capacity–data phase diagram [6].

Algorithmic Reasoning with Transformers. Prior work has studied transformers and related architectures on algorithmic tasks, including sequence recall with external memory character-level mathematical reasoning, and graph algorithms with message-passing networks [11-13]. These representations are tied to the problem vocabulary or input structure, so model capacity tends to grow with problem size. trained transformers on modular arithmetic for lattice-cryptography attacks using baseencoded integers, and scaled modular addition to large moduli and many terms using angular embeddings [14,15]. Our FRE/FCO approach decouples model size from problem

size by factorizing the input and output into fixed-size digit tables.

Digit Decomposition and Positional Encodings. The idea of decomposing integers into digits for neural network input has been explored in number-theoretic neural networks and in systems for learning arithmetic [16,17]. FRE extends this to a per-position embedding table design that enables $O(1)$ parameters.

Factorized Output Heads. Hierarchical and factorized classifiers have been used in language modeling and extreme classification [18-20]. FCO adapts this to a digit-decomposed output for algebraic problems.

CRT and Pohlig-Hellman. The Chinese Remainder Theorem decomposition of DLP is the basis of the Pohlig-Hellman algorithm [21]. Our CRT-BOTH representation embeds this algebraic structure as an inductive bias in the model input.

DLP Learning Theory. proved that learning the parity bit of the discrete logarithm with gradient-based methods becomes intractable as group order grows [22]. Their result applies to cyclic groups of prime order—precisely the case where CRT decomposition is unavailable— so it does not directly predict a wall for our CRT-decomposed setting, but it provides theoretical context for why learning DLP becomes harder as the group order grows.

ECDLP Scaling Study. applied FRE/FCO to elliptic curve DLP and found a structural wall at r = 1883 where no configuration could grok [7]. Our finite-field DLP results suggest the wall is further out, likely due to simpler group structure (cyclic vs. elliptic curve) and better CRT decompositions.

Discussion

Why Fixed-Size Grokking Works

The key insight is that the DLP algorithm—while computationally hard for large groups—*has a fixed representation complexity* when decomposed via CRT. The model needs to learn two independent DLP algorithms on groups of order a and b, where $\max(a, b) \ll q$. The FRE/FCO architecture ensures the model's capacity is spent on the transformer core (learning the algorithm) rather than lookup tables (memorizing the vocabulary).

The Wall Hypothesis

The internal ECDLP scaling study (unpublished) found a structural wall at $r = 1883$ where no configuration (larger models, more data, deeper architectures, bilinear outputs) could induce grokking [7]. The wall appears to be *algebraic* complexity, not capacity: the transformer cannot learn the compositional structure of ECDLP at that scale in a single forward pass. For finite-field DLP with CRT-BOTH, we hypothesize the wall is further out because:

1. **Simpler group structure:** Cyclic groups $(\mathbb{F}_p^*)$ have one operation; elliptic curves have two (point addition + doubling).
2. **Better CRT decompositions:** $q = p - 1$ is usually highly composite, giving balanced splits. ECDLP group orders are often prime (no CRT available).
3. Transparent algebra: The cyclic group structure is more directly learnable than elliptic curve point operations.

If a wall exists for finite-field DLP, we expect it to manifest as:

- Memorization without generalization (the model memorizes but never groks).
- Effective rank saturation in FCO (rank capped at d_{model}).
- Weight explosion in the output tables (observed in the internal study at the wall).

Implications for Scalable Algorithmic Learning

If the fixed-size model can grok DLP at larger primes, this would suggest that representation choice, not model capacity, is a primary bottleneck for scaling grokking to large algebraic problems—though this claim requires the missing ablations (atomic+CRT-BOTH, FRE/FCO without CRT) to be fully substantiated. The model size is determined by the *complexity* of the *algorithm* (which is fixed for DLP), not the size of the *problem* (which grows with p).

This has implications for neural algorithmic reasoning more broadly: if the right representation can decouple model size from problem size, then transformers can learn algorithms at scales previously thought infeasible.

Limitations

1. **Single seed:** Current results use a single seed. Multi-seed replication is needed to confirm robustness.
2. **CRT availability:** The CRT-BOTH representation requires $q = p - 1$ to be composite with a balanced coprime factorization. Primes where q is prime (safe primes) cannot use CRT.
3. **Dataset generation:** For $p > 100$k, the full domain exceeds 10B examples. We may need to subsample.
4. **Compute cost:** Large-prime experiments require HPC resources (LUMI-G).
5. **No cryptanalytic claim:** These experiments use small groups and make no claims about practical DLP cryptanalysis.

Conclusion

We have shown that a fixed ~526k-parameter transformer can learn the finite-field discrete logarithm problem across a range of prime sizes ($p = 113$ to $p = 4{,}001$), with 100% test accuracy at every scale. Not all configurations exhibit grokking: for $p = 601$, $p = 1201$, $p = 2003$, and $p = 4001$, memorization and generalization times coincide (Tmem ≈ Tgrok), indicating direct generalization. The key innovations—Factorized Residue Embedding (FRE), Factorized Candidate Output (FCO), and CRT-BOTH representation—decouple model size from problem size, yielding $O(1)$ parameters regardless of p.

The parameter efficiency advantage grows with p: from 1.1× at $p = 769$ to 2.7× at $p = 4{,}001$. We describe fallback strategies (2× model, 2× data, RichFCO) for extending to larger primes where the base configuration fails.

The effective problem size scales with max(a, b) (the larger CRT factor), not with p: the 35× span in p corresponds to a 16× to 256× span in max(a, b). Claims about scaling to "arbitrarily large primes" apply only when $p - 1$ has a smooth factorization.

Caveat: The CRT decomposition makes per-factor sub-instances few enough that the standard triple-level split allows every sub-instance to appear in both train and test. A sub-instance holdout experiment is needed to determine whether the observed generalization is genuine algorithm learning or sub-instance lookup.

These results suggest that representation choice is a primary factor for scaling algorithmic learning to large problems, but the sub-instance leakage caveat must be resolved before strong claims can be made.

References

1. Power, A., Burda, Y., Edwards, H., Babuschkin, I., & Misra, V. (2022). Grokking: Generalization beyond overfitting on small algorithmic datasets. arXiv. https://doi.org/10.48550/arXiv.2201.02177
2. Nanda, N., Chan, L., Lieberum, T., Smith, J., & Steinhardt, J. (2023). Progress measures for grokking via mechanistic interpretability. In International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2301.05217
3. Liu, Z., Kitouni, O., Nolte, N., Michaud, E. J., Tegmark, M., & Williams, M. (2022). Towards understanding grokking: An effective theory of representation learning. arXiv. https://doi.org/10.48550/arXiv.2205.10343
4. Varma, V., Shah, R., Kenton, Z., Kramár, J., & Kumar, R. (2023). Explaining grokking through circuit efficiency. arXiv. https://doi.org/10.48550/arXiv.2309.02390
5. Thilak, V., Littwin, E., Zhai, S., Saremi, O., Paiss, R., & Susskind, J. (2022). The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon. arXiv. https://doi.org/10.48550/arXiv.2206.04817
6. Vesterlund, D. (2026). Scaling grokking on the discrete logarithm problem: Model capacity, data, and group size.
7. Vesterlund, D. (2026). FRE/FCO scaling on elliptic-curve and finite-field DLP: Internal scaling study.
8. Liu, Z., Michaud, E. J., & Tegmark, M. (2023). Omnigrok: Grokking beyond algorithmic data. In International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2210.01117
9. Gromov, A. (2023). Grokking modular arithmetic. arXiv. https://doi.org/10.48550/arXiv.2301.02679
10. Africa, D. D., et al. (2025). Learning modular exponentiation with transformers. arXiv. https://doi.org/10.48550/arXiv.2506.23679.
11. Graves, A., Wayne, G., & Danihelka, I. (2014). Neural Turing machines. arXiv. https://doi.org/10.48550/arXiv.1410.5401
12. Saxton, D., Grefenstette, E., Hill, F., & Kohli, P. (2019). Analysing mathematical reasoning abilities of neural models. In International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.1904.01557
13. Veličković, P., Puigdomènech Badia, A., Budden, D., Pascanu, R., Banino, A., Dashevskiy, M., Hadsell, R., & Blundell, C. (2022). The CLRS algorithmic reasoning benchmark. In Proceedings of the 39th International Conference on Machine Learning. PMLR. https://arxiv.org/abs/2205.15659
14. Wenger, E., Chen, M., Charton, F., & Lauter, K. (2022). SALSA: Attacking lattice cryptography with transformers. arXiv. https://doi.org/10.48550/arXiv.2207.04785
15. Saxena, E., Alfarano, A., Wenger, E., & Lauter, K. E. (2024). Teaching transformers modular arithmetic at scale. arXiv. https://doi.org/10.48550/arXiv.2410.03569
16. Charton, F. (2024). Learning the greatest common divisor: Explaining transformer predictions. In International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.2308.15594
17. Charton, F. (2022). Linear algebra with transformers. arXiv. https://doi.org/10.48550/arXiv.2112.01898
18. Morin, F., & Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (pp. 246–252). PMLR.
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, 26.
20. Weston, J., Makadia, A., & Yee, H. (2013). Label partitioning for sublinear ranking. In Proceedings of the 30th International Conference on Machine Learning (Vol. 28, No. 2, pp. 181–189). PMLR.
21. Pohlig, S. C., & Hellman, M. E. (1978). An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. IEEE Transactions on Information Theory, 24(1), 106–110. https://doi.org/10.1109/TIT.1978.1055817
22. Takhanov, R., Tezekbayev, M., Pak, A., Bolatov, A., Kadyrsizova, Z., & Assylbekov, Z. (2023). Intractability of learning the discrete logarithm with gradient-based methods. In Proceedings of the 15th Asian Conference on Machine Learning (pp. 1321–1336). PMLR.

Copyright: ©2026. David Vesterlund. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in anymedium, provided the original author and source are credited.