More Efficient SNARKs via Quasi-Abelian Codes: Faster, Smaller, and Field-Agnostic

Zhe Li[1], Hongqing Liu[2], Chaoping Xing[2], Yizhou Yao[2], Chen Yuan[2]

[1] Institute of Mathematics and Interdisciplinary Sciences,
Xidian University, Xi'an, China
lizh0048@e.ntu.edu.sg
[2] School of Computer Science,
Shanghai Jiao Tong University, Shanghai, China
{liu.hong.qing,xingcp,yaoyizhou0620,chen_yuan}@sjtu.edu.cn

Abstract. Linear error-correcting codes play a crucial role in building practical non-interactive arguments of knowledge (SNARKs) with transparent setup, and plausible post-quantum security. Basically, the key to practical efficiency is a linear code with a concretely fast encoding and a high minimum distance. However, to date, none of the candidate codes achieves the best of the two worlds: codes with provable high minimum distance, e.g., Reed-Solomon codes, suffer from quasi-linear time encoding, while linear-time encodable codes, e.g., Spielman's code, have low provable minimum distance.

In this work, we resolve this problem by explicitly constructing a family of Quasi-Abelian (QA) codes over *arbitrarily* large prime fields with concretely high minimum distance and practically efficient encoding algorithms. At the heart of our technical contribution is a fine-grained analysis on the concrete minimum distance of random QA codes of rank 1 and index c over group ring $\mathbb{F}_p[\mathbb{Z}_2^n]$. We show that in practical regimes it attains the well-known Gilbert-Varshamov bound up to a small constant gap $n/(c\log_2 p)$. Concretely, with probability $\geq 1-2^{-127}$, our random QA code over a 128-bit sized prime field with $n=20$, achieves relative minimum distance at least $0.4142, 0.6070, 0.7040$ for code rate $1/2, 1/3, 1/4$, respectively. In comparison, Spielman's code only achieves a minimum distance 0.1 for code rate $1/2$ in the same setting by the state-of-the-art analyses.

We give practically efficient encoding algorithms for QA code over $\mathbb{F}_p[\mathbb{Z}_2^n]$ by leveraging Walsh-Hadamard Transform. Specifically, for code length $c\cdot 2^n$ and rate $1/c$, our encoding only needs $cn\cdot 2^n$ additions/subtractions and $(c-1)\cdot 2^n$ multiplications over $\mathbb{F}_p$, which turns out to be concretely faster than Spielman's code. For encoding a message of length 2^{20} over a 256-bit prime field, our QA code with rate $1/2$ only takes 250 ms, while Spielman's code with rate $0.65, 1/2$ needs 410 ms, 890 ms, respectively.

We then follow the framework of Brakedown (CRYPTO 2023) to build SNARKs over large prime fields from QA codes. For proving ECDSA verification over the scalar field of Curve25519 ($\approx 2^{16}$ constraints), our SNARK needs only 1.44 second in proving, 0.08 second in verification, and a proof size of 3.2 MB. In comparison, Brakedown needs 1.6 second, 0.24 second, and 7.48 MB, respectively.

Table of Contents

1 Introduction . . . 2
1.1 Our Contribution . . . 4
2 Preliminaries . . . 7
2.1 Group Ring and Group Algebra . . . 7
3 Quasi-Abelian Codes . . . 9
3.1 Group Code. . . . 9
3.2 Minimum Distance of Quasi-Abelian Code. . . . 11
3.3 Concrete Minimum Distance . . . 18
4 Fast Encoding of Quasi-Abelian Codes . . . 19
4.1 Walsh-Hadmard Transform . . . 19
4.2 Fast Encoding of Quasi-Abelian Codes . . . 20
5 Polynomial Commitments and SNARKs from Quasi-Abelian Codes . . . 22
6 Experiments and Results . . . 24
6.1 Encoding Time . . . 25
6.2 Polynomial Commitment Schemes . . . 26
6.3 Application to SpartanNIZK [Set20] . . . 28
6.4 Application to ECDSA Verification . . . 30
A Detailed Numbers of the Experiments . . . 34
A.1 Polynomial Commitment Scheme . . . 35
A.2 Application to SpartanNIZK [Set20] . . . 37
A.3 Application to SpartanSNARK [Set20] . . . 37
B Missing Proofs of Section 3.1 . . . 38

1 Introduction

Succinct non-interactive arguments of knowledge (SNARKs) allow a prover to convince a verifier of the truth of a statement in one single message of sublinear size. Driven by their applications especially in blockchain, SNARKs have been extensively studied over the past decade. In the big SNARK family, code-based constructions [AHIV17, BCG+17, BBHR18, BCR+19, BCL22, RR22, XZS22, GLS+23, ZCF24, BFK+24, ACFY24, BCF+25, ACFY25] have been standing out because of their concrete efficiency, transparent setup, and plausible post-quantum security. In a bird's eye view, there are two templates that allow building SNARKs from some linear code, in which the concrete efficiency crucially relies on encoding time and the minimum distance of the code.

Via foldable codes. There is a line of works that builds SNARKs from codes of a strong algebraic structure [BBHR18, BCR+19, ZCF24, ACFY24, ACFY25], e.g., Reed-Solomon codes, featuring polylogarithmic (in the statement size) proof size and verifier time. However, constructions based on Reed-Solomon codes are restricted to a specific "FFT-friendly" field, due to the underlying FFT (Fast

Fourier Transform)-based encoding algorithm. This issue was recently solved in Basefold [ZCF24] by generalizing to *foldable codes*, which enjoy a divide-and-conquer style encoding algorithm that is field-agnostic. Moreover, foldable codes also have a concretely high minimum distance. For example, it achieves a relative minimum distance 0.728 for code rate 1/8 over a 256-bit sized field [ZCF24]. However, these schemes all suffer from large prover time, as their encoding algorithms involve a quasi-linear number of multiplications over the underlying large field, which constitutes a major bottleneck in practice.

Via fast encodable codes. Another line of works [AHIV17, BCL22, XZS22, GLS+23, BFK+24, BCF+25], following the blueprint of Brakedown [GLS+23] that basically allows building SNARKs from any linear code, constructs practically efficient SNARKs via a concretely fast encodable code. For example, Brakedown uses Spielman's code [Spi96], a well-known linear-time encodable code. However, Spielman's code suffers from a low minimum distance, making the proof size as well as verification time significantly larger than the Reed-Solomon instantiation of Brakedown[3]. Concretely, Spielman's code only achieves a distance 0.1 for code rate 1/2 over a 256-bit sized field by the state-of-the-art analyses [Hab23].

In this work, we follow the second approach, seeking a linear code with a concretely fast encoding and a high relative minimum distance. In fact, there are many potential candidates in the literature, in particular, from an orthogonal line of works that rely on syndrome decoding assumptions to build pseudorandom correlation generators (PCGs). For instance, Expand-Accumulate (EA) code [BCG+22], Expand-Convolute (EC) code [RRT23] and more recently Block-Accumulate (BA) code [KPR+25] have proven to be very efficient, and have concretely good minimum distance, in particular, over a binary field $\mathbb{F}_2$. However, many SNARK applications are aimed for sufficiently large fields, for example, a Mersenne prime field $\mathbb{F}_{2^{127}-1}$, or the scalar field $\mathbb{F}_{2^{255}-19}$ of Curve25519 [Ber06] that is widely used in ECDSA verification. Although their encoding algorithms are field-agnostic and heuristically moving to large fields should yield better minimum distance, it is very challenging to generalize the analyses for such codes that are combinatorial in nature. To our best knowledge, only EA code has been formally analyzed over arbitrarily large fields and thus implies SNARKs [BFK+24]. However, [BFK+24] only proves a distance 0.1 for code rate 1/2, and there remains a huge gap to 0.25 that EA is conjectured to have [BCG+22].

Repeat-Accumulate-Accumulate (RAA) code [DJM98] is another promising candidate that has super fast encoding over arbitrary finite fields and provable high minimum distance over the binary field. However, like those codes used in PCG scenarios, extending their combinatorial arguments [BCF+25] to large prime fields remains an open problem, and it seems that new innovations from advanced combinatorics are required. Still, it implies a SNARK over *binary extension fields*

[3] In Brakedown's framework, the proof size is $O(\sqrt{\frac{-\lambda N}{\log_2(1-\delta/3)}})$, where λ is the security parameter, N is the statement size, and δ is the relative minimum distance of the code. Thus, larger δ implies smaller proof size.

via Brakedown's template, by using a "packed" RAA code [BCF+25], which, however, only preserves the relative distance of binary RAA codes.

We remark that all the codes mentioned above (except for Reed-Solomon code), rely on probabilistic arguments to bound the minimum distance. That is, the code will be randomly sampled from its family. Thus, the best code one may dream of could be a fast code that, like random linear codes, approaches the well-known Gilbert-Varshamov (GV) bound [Var57]. Note that for a typical 256-bit field of SNARKs' interest, GV bound guarantees with high probability distance > 0.49 for code rate $1/2$, which is substantially larger than 0.1 of Spielman's code or EA code that we have known. We therefore raise the following question:

Can we find a linear code with a concretely fast encoding and provable high minimum distance (close to GV bound) over arbitrarily large fields?

1.1 Our Contribution

In this work, we answer the above question in the affirmative by proving a concretely high minimum distance for a family of Quasi-Abelian (QA) codes [Was77] over arbitrarily large prime fields with fast encoding algorithms. QA code is a well-studied family of codes in algebraic coding theory that has been known to asymptotically attain GV bound [FL15, FL22]. To begin with, we introduce basic notions for QA code. Let $\mathbb{F}$ be a finite field, $\mathbb{G}$ be a group, and $\mathbb{F}[\mathbb{G}]$ denote a group ring. A Quasi-$\mathbb{G}$ code $\mathcal{C}$ over $\mathbb{F}[\mathbb{G}]$ of rank k and index c (so k/c is the code rate), is defined in the following way:

$$\mathcal{C} = \left\{ (m_1, \ldots, m_k) \cdot \begin{pmatrix} a_{1,1} & \cdots & a_{1,c} \\ \vdots & \ddots & \vdots \\ a_{k,1} & \cdots & a_{k,c} \end{pmatrix} \mid (m_1, \ldots, m_k) \in (\mathbb{F}_p[\mathbb{G}])^k \right\},$$

where each $a_{i,j} \in \mathbb{F}[\mathbb{G}]$, for $i \in [1, k]$, $j \in [1, c]$. If $\mathbb{G}$ is an abelian group, $\mathcal{C}$ is called a QA code. In fact, $\mathcal{C}$ can be viewed as a linear code over $\mathbb{F}$ with code length $c \cdot |\mathbb{G}|$ and dimension $k \cdot |\mathbb{G}|$. Recently, QA code has been used to build programmable PCGs over small fields [BCCD23, BBC+24, LXYY25a, LXYY25b]. These works focus on $\mathbb{F} = \mathbb{F}_4$ and $\mathbb{G} = \mathbb{Z}_3^n$ such that multiplications in $\mathbb{F}[\mathbb{G}]$ can be done in $O(|\mathbb{G}| \log |\mathbb{G}|)$ time by FFT algorithms. Moreover, Li et al. [LLX+26] recently showed a novel approach to computing multiplications over $\mathbb{F}_p[\mathbb{Z}_2^n]$ by performing only $O(n \cdot 2^n)$ *additions* and $O(2^n)$ multiplications in $\mathbb{F}_p$, which is extremely fast as additions are significantly cheaper than multiplications for large prime fields. This allows them to obtain faster PCGs over large prime fields via QA code, beating those from EA/EC/BA codes.

As random QA codes asymptotically approach GV bound, it seems that [LLX+26] has also opened a door to building more efficient SNARKs via QA code. Unfortunately, not yet. The caveat is that Fan and Lin's argument [FL15] only applies to QA codes with fixed group size $|\mathbb{G}|$ and fixed rate k/c, where the index c goes to infinity[4]! However, to make use of the fast multiplication

[4] One can think of a trivial case that $|\mathbb{G}| = 1$, in which $\mathbb{F}_p[\mathbb{G}] = \mathbb{F}_p$ and random QA codes coincide with random linear codes.

algorithm, it falls in the case that the index c and the rate k/c are fixed, with the group size $|\mathbb{G}|$ going to infinity.

Minimum distance of random QA codes with $|\mathbb{G}|$ going to infinity has not been well-understood for many years. The only argument we are aware of is due to Gaborit and Zémor [GZ06], which consider the case that $\mathbb{G}$ is cyclic[5]. However, we are interested in $\mathbb{G} = \mathbb{Z}_2^n$, which is a non-cyclic group. Notably, in [BCCD23] the authors conjectured that Gaborit and Zémor's argument can be extended to arbitrary abelian groups and thus QA code approaches GV bound with $|\mathbb{G}|$ going to infinity.

Minimum Distance of QA codes. We give fine-grained analyses on the minimum distance of QA codes, with a special focus on $\mathbb{F}_p[\mathbb{Z}_2^n]$, where $\mathbb{F}_p$ is an arbitrary large prime field. Our argument can be naturally generalized to QA codes over $\mathbb{F}_q[\mathbb{Z}_d^n]$ where $d \mid q-1$, as well as a generic $\mathbb{F}[\mathbb{G}]$ for a concrete group $\mathbb{G}$. Our main result is stated as follows:

Theorem 1.1. *Let $\mathbb{G} = \mathbb{Z}_2^n$ with $N = 2^n$ and $\gcd(p, N) = 1$. Assume that $\epsilon = g_p(\delta) - \frac{1+\log_p N}{c} > 0$. Then, with probability at least*

$$\max\left\{1 - \frac{c(c-1)N}{2p^2} - \frac{p^{-\lceil \frac{c-1}{c\delta} \rceil c\epsilon}}{(1-p^{-c\epsilon})(p-1)},\quad 1 - \frac{cN}{p} - \frac{p^{-\lceil \frac{1}{\delta} \rceil c\epsilon}}{(1-p^{-c\epsilon})(p-1)}\right\},$$

a random QA code $\mathcal{C} \subseteq (\mathbb{F}_p[\mathbb{Z}_2^n])^c$ of rank 1 and index c has relative minimum distance δ, where $g_p(\delta) = 1 - \delta\log_p(p-1) + \delta\log_p\delta + (1-\delta)\log_p(1-\delta)$.

Since our random QA code has rate $r = \frac{1}{c}$, the classic GV bound claims that the linear code exists if $g_p(\delta) - \frac{1}{c} > 0$. The requirement in Theorem 1.1 $g_p(\delta) - \frac{1+\log_p N}{c} > 0$ only has a gap $\frac{\log_p N}{c}$ from classic GV bound over $\mathbb{F}_p$. Since p is much larger than code length N in our application, this gap becomes insignificant. Theorem 1.1 allows us to compute a concrete minimum distance for any constant c, and N, as long as p is sufficiently large. In particular, we are interested in QA code over $\mathbb{F}_p[\mathbb{Z}_2^n]$ of rank 1 and index 2, for $n = 20$. Concretely, for a 128-bit sized field, it achieves distance δ at least 0.4142 with probability at least $1 - 2^{-127}$, while for a 256-bit sized field, $\delta = 0.4570$ with probability at least $1 - 2^{-255}$. Our bounds substantially improve a recent result in [LLX+26], which only gives a distance 0.1 with a failure probability $2^{-81.4}$ for 128-bit sized fields. More concrete numbers on the minimum distance are referred to Table 2.

While we can obtain a concretely high minimum distance from Theorem 1.1, it remains open whether QA code is asymptotically good for fixed index c and field size p, as our lower bound is meaningless when $|\mathbb{G}| = N \to \infty$. Thus it would be of great theoretical interest to give a tight upper/lower bound on the probability that a random QA code can achieve distance δ, but such a development is out of reach of this work and we leave it as future work.

[5] In this case, QA codes coincide with Quasi-Cyclic (QC) codes.

Fast Encoding of QA codes. We mainly follow [LLX+26] that leverages Walsh-Hadamard Transform (WHT) to compute a ring isomorphism $\mathbb{F}_p[\mathbb{Z}_2^n] \to \mathbb{F}_p^{2^n}$, which is implied by $\mathbb{F}_p[\mathbb{Z}_2^n] \simeq \mathbb{F}_p[\mathtt{X}_1, \ldots, \mathtt{X}_n]/(\mathtt{X}_1^2 - 1, \ldots, \mathtt{X}_n^2 - 1) \simeq \mathbb{F}_p^{2^n}$. Hence, to compute $m \cdot a$, where $m, a \in \mathbb{F}_p[\mathbb{Z}_2^n]$, it suffices to first compute $\mathsf{WHT}(m) * \mathsf{WHT}(a)$ and then perform inverse Walsh-Hadamard Transform (WHT^{-1}), where '$*$' denotes component-wise multiplication. The first step requires $2n \cdot 2^n$ additions/subtractions (from 2 calls of WHT) and 2^n multiplications in $\mathbb{F}_p$, while performing WHT^{-1} needs $n \cdot 2^n$ additions/subtractions and 2^n multiplications in $\mathbb{F}_p$. We remark that the second step is unnecessary in PCG scenarios [LLX+26], but it is not the case for building SNARKs via QA codes.

We further show several optimizations over the above encoding algorithm, such as using a systematic form of $a_1, \ldots, a_c$ where $a_1 = 1 \in \mathbb{F}_p[\mathbb{G}]$. Consequently, we obtain the following results:

Theorem 1.2. *A random systematic QA code $\mathcal{C} \subseteq (\mathbb{F}_p[\mathbb{Z}_2^n])^c$ of rank 1 and index c can be encoded by performing $cn \cdot 2^n$ additions/subtractions (due to c calls of WHT) and $(c-1) \cdot 2^n$ multiplications in $\mathbb{F}_p$.*

We then evaluate the concrete efficiency of our encoding algorithms, and compare them to other fast codes proposed in the literature. The results are summarized in Figure 1 and Table 6. For the same rate 1/2, our encoding algorithm is only slightly slower than RAA code [DJM98], and significantly outperforms other fast codes [GLS+23, BFK+24] over large prime fields. We emphasize that the state-of-the-art analyses on minimum distance of RAA codes [BCF+25] are over $\mathbb{F}_2$ and only work for inverse rate ≥ 4. We view matching the concrete efficiency of RAA codes over large prime fields and inverse rate 2 as an advantage of QA codes, given that QA codes have a large provable minimum distance in such a scenario.

More Efficient SNARKs via QA code. Finally, we use QA code to build SNARKs via Brakedown's template. Since our QA code is faster and offers better provable minimum distance, our SNARK significantly outperforms [GLS+23, BFK+24] in terms of prover time, proof size, as well as verifier time. Detailed comparisons are referred to in Section 6. We remark that our implemented scheme has square-root proof size and verification time, which can be reduced to polylogarithmic by employing a "code-switching" technique as in Orion [XZS22, RR24, BCF+25]. We leave it as future work.

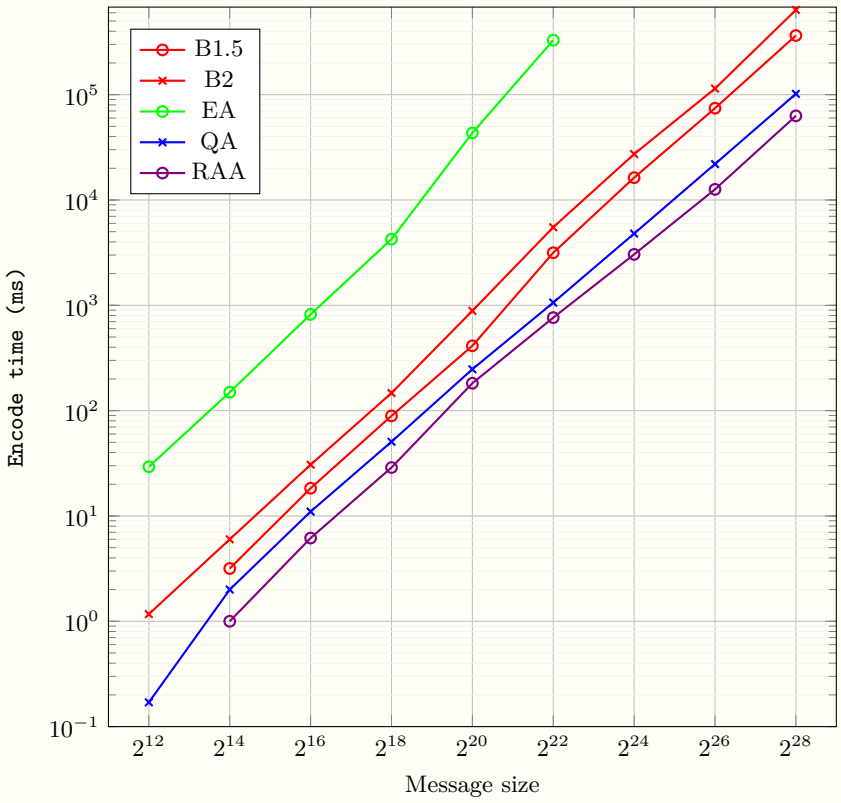
Our Rust code implementation of the encoding algorithm, PCS and SNARK is available at `https://anonymous.4open.science/r/WHTPCS-1F82`.

A minor downside of our scheme is that our Walsh-Hadamard Transform based encoding algorithm supports arbitrary finite fields, except for binary extension fields, i.e., $\mathbb{F}_{2^k}$. Although $\mathbb{F}_{2^k}$ receives particular interest in proving binary relations, our SNARK over large prime fields[6] naturally fits most applications,

[6] We also need the field size to be quite large, e.g., $> 2^{64}$, otherwise the distance gap to GV-bound might be large. This does not appear to be a limitation in SNARK scenarios as to achieve λ-bit security the field size is typically at least $\Omega(2^\lambda)$.

like ECDSA verification. Beyond that, we believe QA codes over $\mathbb{F}_{2^k}$ could have fast encoding algorithms as well.

Figure 1: Encode time performance comparisons



B1.5 and B2 stand for Spielman's code [Spi96] used in Brakedown [GLS+23] with respect to inverse rate 1.521 and 2. The rate of EA code [BFK+24], RAA code [BCF+25] and QA code is 1/2. Messages are over the sclar field $\mathbb{F}_{2^{255}-19}$ of Curve25519.

2 Preliminaries

We denote $\mathbb{F}_p$ as the finite field of p elements, where p is a prime. For a vector $\vec{v}$, we denote $\mathsf{wt}(\vec{v})$ as the Hamming weight of $\vec{v}$, i.e., the number of non-zero entries in $\vec{v}$. For vectors $\vec{u}, \vec{v}$ of the same dimension, we write $\mathsf{d}(\vec{u}, \vec{v}) = \mathsf{wt}(\vec{u} - \vec{v})$ as their Hamming distance.

2.1 Group Ring and Group Algebra

Definition 2.1 (Group Ring). *Let R be a ring, and G be a group. Define the set $R[G] = \{\sum_{g \in G} a_g g \mid a_g \in R,$ and there are finitely many $a_g \neq 0\}$. The two*

elements $\sum_{g\in G} a_g g, \sum_{g\in G} b_g g \in R[G]$ *are said to be equal if and only if* $a_g = b_g$ *for any* $g \in G$*. Define addition and multiplication on* $R[G]$ *as follows:*

$$(\sum_{g\in G} a_g g) + (\sum_{g\in G} b_g g) = \sum_{g\in G} (a_g + b_g) g,$$
$$(\sum_{g\in G} a_g g) \cdot (\sum_{g\in G} b_g g) = \sum_{g\in G} \big(\sum_{h\in G} (a_h b_{h^{-1}g}) \big) g.$$

$R[G]$ *with addition and multiplication defined as above is called a group ring.*

Lemma 2.2 (Properties of Group Rings). *Let* R *be a ring,* G *be a multiplicative group, and* $R[G]$ *be a group ring in Definition 2.1. The following holds:*

1. $R[G]$ *is commutative if and only if* R *is commutative and* G *is Abelian.*
2. *If* R *has identity* 1*, and* G *has identity* e*, then* $1e$ *is the identity of* $R[G]$*.*

In this work, we particularly instantiate $R[G]$ with R a finite field $\mathbb{F}$ and G an Abelian group $\mathbb{G}$. Such $\mathbb{F}[\mathbb{G}]$ is also known as *group algebra*. By fixing an ordering $g_1, \ldots, g_n$ of the elements in $\mathbb{G}$, $\mathbb{F}[\mathbb{G}]$ is isomorphic to $\mathbb{F}^n$ with respect to $\varphi : \mathbb{F}[\mathbb{G}] \to \mathbb{F}^n, \sum_{i=1}^n a_i g_i \mapsto (a_1, \ldots, a_n)$, where the multiplication over $\mathbb{F}^n$ is defined as follows:

$$(a_1, \ldots, a_n) \cdot (b_1, \ldots, b_n) = \Big(\sum_{g_i g_j = g_1} a_i b_j, \ldots, \sum_{g_i g_j = g_n} a_i b_j \Big).$$

In fact, the multiplication over $\mathbb{F}^n$ can be characterized by standard matrix multiplications. For $a := \sum_{i=1}^n a_i g_i \in \mathbb{F}[\mathbb{G}]$, define

$$\mathbf{A} = \begin{pmatrix} \varphi(a \cdot g_1) \\ \vdots \\ \varphi(a \cdot g_n) \end{pmatrix} \in \mathbb{F}^{n\times n}.$$

Essentially, the matrix $\mathbf{A}$ defines the multiply-by-a map $m \mapsto ma$ in $\mathbb{F}[\mathbb{G}]$ with respect to the basis $(g_1, \ldots, g_n)$.

$$\varphi(ma) = \varphi(m)\mathbf{A} = (m_1, \ldots, m_n) \begin{pmatrix} \varphi(a \cdot g_1) \\ \vdots \\ \varphi(a \cdot g_n) \end{pmatrix}.$$

Lemma 2.3. *Let* $\mathbb{G}_1, \mathbb{G}_2$ *be two finite Abelian groups. Then*

$$\mathbb{F}[\mathbb{G}_1 \times \mathbb{G}_2] \simeq \mathbb{F}[\mathbb{G}_1] \otimes_{\mathbb{F}} \mathbb{F}[\mathbb{G}_2],$$

where $\times$ *denotes the direct product, and* $\otimes_{\mathbb{F}}$ *denotes the tensor product over* $\mathbb{F}$*.*

Examples. We give concrete examples of group rings.

- Let $\mathbb{G} = \{1\}$ be the trivial group consisting of only an identity. Then $\mathbb{F}[\mathbb{G}]$ is isomorphic to $\mathbb{F}$.
- Let $\mathbb{G} = \mathbb{Z}/n\mathbb{Z}$ be the cyclic group with n elements. Assuming that n is coprime to $|\mathbb{F}|$, then one can see $\mathbb{F}[\mathbb{G}]$ is essentially the polynomial ring $\mathbb{F}[\mathtt{X}]/(\mathtt{X}^n-1)$, with respect to the ordering $\overline{0}, \overline{1}, \ldots, \overline{n-1}$ and the isomorphism $\varphi : \mathbb{F}[\mathbb{G}] \to \mathbb{F}[\mathtt{X}]/(\mathtt{X}^n - 1), \sum_{i=0}^{n-1} a_i \overline{i} \mapsto \sum_{i=0}^{n-1} a_i \mathtt{X}^i$.
- Let $\mathbb{G} = \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ with n_1, n_2 coprime to $|\mathbb{F}|$. Then by Lemma 2.3,

$$\begin{aligned}\mathbb{F}[\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}] \simeq \mathbb{F}[\mathbb{Z}_{n_1}] \otimes_{\mathbb{F}} \mathbb{F}[\mathbb{Z}_{n_2}] &\simeq \mathbb{F}[\mathtt{X}]/(\mathtt{X}^{n_1} - 1) \otimes_{\mathbb{F}} \mathbb{F}[\mathtt{Y}]/(\mathtt{Y}^{n_2} - 1) \\ &\simeq \mathbb{F}[\mathtt{X}, \mathtt{Y}]/(\mathtt{X}^{n_1} - 1, \mathtt{Y}^{n_2} - 1).\end{aligned}$$

Moreover, assuming $\mathbb{F}$ is a field of q elements and n divides $q-1$, and $\alpha_0, \ldots, \alpha_{n-1}$ are n roots of $\mathtt{X}^n - 1$ over $\mathbb{F}$. Then, $\phi : f(\mathtt{X}) \mapsto (f(\alpha_0), \ldots, f(\alpha_{n-1}))$ is a ring isomorphism of $\mathbb{F}[\mathtt{X}]/(\mathtt{X}^n - 1)$ and the module $\mathbb{F}^n$. Hence, $\phi \circ \varphi : \mathbb{F}[\mathbb{G}] \to \mathbb{F}^n$ is a ring isomorphism. Finally let $d \mid q - 1$ and $\mathbb{G} = \mathbb{Z}_d^n$, then by Lemma 2.3, $\mathbb{F}[\mathbb{G}] \simeq \mathbb{F}[\mathtt{X}_1, \ldots, \mathtt{X}_n]/(\mathtt{X}_1^d - 1, \ldots, \mathtt{X}_n^d - 1) \simeq \mathbb{F}^N$, where $N = d^n$.

In this paper, we focus on a prime field $\mathbb{F}_p$ with $p > 2$ and thus $2 \mid p - 1$, and we have $\mathbb{F}_p[\mathbb{Z}_2^n] \simeq \mathbb{F}_p[\mathtt{X}_1, \ldots, \mathtt{X}_n]/(\mathtt{X}_1^2 - 1, \ldots, \mathtt{X}_n^2 - 1) \simeq \mathbb{F}_p^{2^n}$.

3 Quasi-Abelian Codes

We start by presenting some background about Quasi-Abelian codes which is an important family of group codes. We refer the interested readers to [FL22] for more details.

Let $\mathbb{F}_p[\mathbb{G}]$ be an $\mathbb{F}_p$-vector space with basis $\mathbb{G}$, i.e., $\mathbb{F}_p[\mathbb{G}] = \{\sum_{x \in \mathbb{G}} a_x x : a_x \in \mathbb{F}_p\}$. Recall that the multiplication of the group $\mathbb{G}$ can be naturally extended to the multiplication of $\mathbb{F}_p[\mathbb{G}]$ in the following ways:

$$\left(\sum_{x \in \mathbb{G}} a_x x\right) \cdot \left(\sum_{y \in \mathbb{G}} b_y y\right) = \sum_{z \in \mathbb{G}} \left(\sum_{xy = z} a_x b_y\right) z.$$

In this sense, $\mathbb{F}_p[\mathbb{G}]$ becomes a group algebra of dimension $|\mathbb{G}|$. We refer to Section 2.1 for more details about group algebra. Since every element in $\mathbb{F}_p[\mathbb{G}]$ can be seen as a vector of length $|\mathbb{G}|$ with respect to a fixed ordering of elements in $\mathbb{G}$, we can define the Hamming metric for $\mathbb{F}_p[\mathbb{G}]$.

For any element $a = \sum_{x \in \mathbb{G}} a_x x \in \mathbb{F}_p[\mathbb{G}]$, the support of a is defined as $\{x \in \mathbb{G} : a_x \neq 0\}$ and the weight of a is defined as the size of a's support, denoted by $\mathsf{wt}(a)$. Similarly, for two elements $a = \sum_{x \in \mathbb{G}} a_x x$, $b = \sum_{y \in \mathbb{G}} b_y y$ in $\mathbb{F}_p[\mathbb{G}]$, the distance $\mathsf{d}(a, b)$ is $\mathsf{wt}(a - b)$.

3.1 Group Code.

Before proving Theorem 1.1, we first introduce the group code.

Definition 3.1. *Any left ideal $\mathcal{C}$ of $\mathbb{F}_p[\mathbb{G}]$ is called a group code of group $\mathbb{G}$ ($\mathbb{F}_p[\mathbb{G}]$-code for short) over the field $\mathbb{F}_p$. The distance of $\mathcal{C}$ is defined as*

$$\mathsf{d}(\mathcal{C}) = \min_{a \in \mathcal{C} \setminus \{0\}} \mathsf{wt}(a).$$

The dimension of group code $\mathcal{C}$ is the dimension of $\mathcal{C}$ over $\mathbb{F}_p$.

In what follows, group $\mathbb{G}$ is an abelian group of size N and $\gcd(p, N) = 1$. To further study the group algebra $\mathbb{F}_p[\mathbb{G}]$, we first need to decompose $\mathbb{F}_p[\mathbb{G}]$ into the direct sum of simple ideals.

Definition 3.2. *Let R be a ring.*

1. *An element $e \in R$ is called an* idempotent *if $e^2 = e$.*
2. *An idempotent $e \in R$ is said to be central if e is also in the center of R.*
3. *An idempotent $e \in R$ is said to be primitive if $e \neq 0$, and there are no idempotents $e' \neq 0 \neq e''$ such that $e = e' + e''$ and $e'e'' = 0 = e''e'$.*
4. *If $e_1, \ldots, e_m$ are central idempotents such that $e_i e_j = 0$ for $i \neq j$, and $\sum_{i=1}^{m} e_i = 1$, then $\{e_1 \ldots e_m\}$ forms a* complete system of central idempotents *of R.*
5. *If $e_1, \ldots, e_m$ are all the primitive central idempotents, then $\{e_1 \ldots e_m\}$ forms a* complete system of primitive central idempotents *of R.*

Since we will focus on a commutative R, an idempotent $e \in R$ is always central.

Theorem 3.3. *[FL22, Theorem 3.10] Let $\mathbb{G}$ be an abelian group of order $N > 1$ and $\gcd(p, N) = 1$. Let $\{e_0 = \frac{1}{N} \sum_{x \in \mathbb{G}} x, e_1, \ldots, e_m\}$ be the set of all primitive idempotents of $\mathbb{F}_p[\mathbb{G}]$, then the simple ideals $\mathbb{F}_i = \mathbb{F}_p[\mathbb{G}]e_i$ are field extensions over $\mathbb{F}_p$ with $\dim \mathbb{F}_i = d_i \geq 1$ and*

$$\mathbb{F}_p[\mathbb{G}] = \mathbb{F}_0 \oplus \mathbb{F}_1 \oplus \cdots \oplus \mathbb{F}_m.$$

Let $\mathbb{Z}_N$ be the residue integer ring modulo $N > 1$ and the p-coset $Q_a = \{a, pa, \cdots, \} \subseteq \mathbb{Z}_N$, where $a \in \mathbb{Z}_N$. Let $\mu_p(N)$ be the minimal size of non-trivial p-cosets of $\mathbb{Z}_N$. One can prove the following.

Lemma 3.4. *[FL22, Lemma 3.12] Using the notations in Theorem 3.3, then $\min\{d_1, \ldots, d_m\} = \mu_p(N)$.*

Remark 3.5. In our application, we are mainly concerned about $\mathbb{G} = \mathbb{Z}_2^n$ and p is a large prime. In this case, $|\mathbb{G}| = 2^n$ and by setting $a = 2^{n-1}$, it is easy to see the size of Q_a is 1. Thus, we conclude that $\mu_p(2^n) = 1$. In fact, by $\mathbb{F}_p[\mathbb{Z}_2^n] \simeq \mathbb{F}_p[\mathtt{X}_1, \ldots, \mathtt{X}_n]/(\mathtt{X}_1^2 - 1, \ldots, \mathtt{X}_n^2 - 1) \simeq \mathbb{F}_p^{2^n}$, it follows that $\mathbb{F}_p[\mathbb{Z}_2^n]$ consists of 2^n simple ideals and each has dimension 1.

One nice property of the group code is that it is a family of balanced codes.

Definition 3.6 (Balanced Codes). *Let $\mathcal{C} \subseteq \mathbb{F}_p^N$ be a linear code of dimension k and $I = \{1, \ldots, N\}$ be the index set. We say that $\mathcal{C}$ is a balanced code if there exist $I_1, \ldots, I_s$ be the subset of I and integer t such that*

1. *for $j = 1 \dots s$, I_j is an information set of size k and, i.e., for any codeword $c \in \mathcal{C}$, the projection $\rho_j : \mathbb{F}_p^I \to \mathbb{F}_p^{I_j}$ of c on I_j is an information vector for $j = 1, \dots, s$.*
2. *for any $i \in I$, $|\{j : i \in I_j\}| = t$, i.e., each index i appears exactly in t information sets.*

Note that the notion of balanced codes in this paper is different from that used in the reductions for LPN [BLVW19].

In [BM06], they proved that group codes are balanced codes.

Lemma 3.7. *[BM06, Lemma 2.2] Group codes are balanced codes.*

Utilizing this fact, Fan and Lin [FL22] further proved the following.

Lemma 3.8 (Theorem 4.3 [FL22]). *Let $\mathcal{C}$ be a balanced code over $\mathbb{F}_p$ of length N and dimension k. Let $\mathcal{B} \subseteq \mathcal{C}$ and $\omega = \sum_{b \in \mathcal{B}} \frac{\mathsf{wt}(b)}{N|\mathcal{B}|}$ (the average relative weight of $\mathcal{B}$). If $0 \le \omega \le 1 - 1/p$, then we have*

$$|\mathcal{B}| \le p^{k h_p(\omega)},$$

where $h_p(\omega) \coloneqq \omega \log_p(p-1) - \omega \log_p \omega - (1-\omega)\log_p(1-\omega)$ is the p-ary entropy function.

Lemma 3.8 shows that the cardinality of a subset $\mathcal{B} \subseteq \mathcal{C}$ is upper bounded by a function with respect to w, the average relative weight of $\mathcal{B}$. In particular, if w is small, then $|\mathcal{B}|$ is also small. This is a key property that we will utilize to upper bound the number of codewords with small weight for random QA codes. For instance, the number of Hamming weight less than ωN codewords in a balanced code $\mathcal{C}$ of length N and dimension k is at most $p^{k h_p(\omega)}$.

Definition 3.9 (Product of Codes). *For $\mathcal{C} \subseteq \mathbb{F}^n$, the product $\mathcal{C}^m \subseteq \mathbb{F}^{nm}$ is defined as m copies of $\mathcal{C}$:*

$$\mathcal{C}^m \coloneqq \{(c_1 \dots c_m) : \forall i \in [1, m], c_i \in \mathcal{C}\}.$$

If $\mathcal{C}$ is of dimension k, then $\mathcal{C}^m$ is of dimension $k \cdot m$.

Corollary 3.10 (Lemma 4.6, [FL22]). *If $\mathcal{C}$ is a balanced code, then $\mathcal{C}^h$ is also a balanced code for any $h \ge 1$.*

Corollary 3.10 implies Lemma 3.8 holds for both group codes and quasi-group codes.

3.2 Minimum Distance of Quasi-Abelian Code.

We are now ready to prove our main theorem.

Definition 3.11 (Quasi-Abelian codes). *The Quasi-Abelian code of index c and rank 1 is defined as*

$$\mathcal{C}_{a_1,\dots,a_c} = \{(ba_1, \dots, ba_c) : b \in \mathbb{F}_p[\mathbb{G}]\},$$

where $a_1, \dots, a_c \in \mathbb{F}_p[\mathbb{G}]$.

To bound the minimum distance of a random QA code $\mathcal{C}_{a_1,\ldots,a_c}$, where $a_1,\ldots,a_c \xleftarrow{\$} \mathbb{F}_p[\mathbb{G}]$, we define the random binary variable X_b for $b \in \mathbb{F}_p[\mathbb{G}]$ such that

$$X_b = 1 \text{ if } 0 < \mathsf{wt}(ba_1,\ldots,ba_c) < cN\delta, \text{ and } X_b = 0 \text{ otherwise.}$$

In this way, it suffices to consider the expectation of $\sum_{b\in\mathbb{F}_p[\mathbb{G}]\setminus\{0\}} X_b$ over the randomness of $a_1,\ldots,a_c$. For $b \in \mathbb{F}_p[\mathbb{G}]$, let d_b be the dimension of the ideal $\mathbb{F}_p[\mathbb{G}]b$ generated by b. Fan and Lin [FL22] proved the following[7].

Lemma 3.12. *Let $\mathbb{F}_p[\mathbb{G}]b$ be an ideal of dimension d_b. Then, it holds that*

$$E[X_b] \le p^{-cd_b g_p(\delta)} \tag{1}$$

where $g_p(\delta) := 1 - h_p(\delta)$.

Proof. By Corollary 3.10, the code $(\mathbb{F}_p[\mathbb{G}]b)^c$ is balanced, and it is easy to see that $(\mathbb{F}_p[\mathbb{G}]b)^c$ has dimension cd_b. Since $a_1,\ldots,a_c$ are sampled independently and uniformly from $\mathbb{F}_p[\mathbb{G}]$, the tuple $(ba_1,\ldots,ba_c)$ is uniformly distributed over $(\mathbb{F}_p[\mathbb{G}]b)^c$. Applying Lemma 3.8, the number of codewords in $(\mathbb{F}_p[\mathbb{G}]b)^c$ with Hamming weight at most $cN\delta$ is at most $p^{cd_b h_p(\delta)}$. On the other hand, $|(\mathbb{F}_p[\mathbb{G}]b)^c| = p^{cd_b}$. Hence,

$$E\left[X_b\right] \le \frac{p^{cd_b h_p(\delta)}}{p^{cd_b}} = p^{-cd_b g_p(\delta)}$$

as claimed. □

Intuitively, Lemma 3.12 shows that ideals of smaller dimension contribute a larger expectation. To formalize this, let $T_\ell := \{I \le \mathbb{F}_p[\mathbb{G}] : \dim(I) = \ell\}$ denote the the set of dimension-ℓ ideals. For an ideal I, define $I^* := \{b \in I : \mathbb{F}_p[\mathbb{G}]b = I\}$, i.e., the set of generators of I. Then, $\mathbb{F}_p[\mathbb{G}]$ can be written as disjoint union as follows:

$$\mathbb{F}_p[\mathbb{G}] = \{0\} \cup \Big(\bigcup_{0\neq I\le\mathbb{F}_p[\mathbb{G}]} I^* \Big).$$

Recall from Lemma 3.3 that

$$\mathbb{F}_p[\mathbb{G}] = \mathbb{F}_p[\mathbb{G}]e_0 \oplus \cdots \oplus \mathbb{F}_p[\mathbb{G}]e_m,$$

where $1 + m \le N$. Informally, every ideal I can be represented as a direct sum of some $\mathbb{F}_p[\mathbb{G}]e_i$, and $e_0,\ldots,e_m$ induces a basis. Consequently, every element $b \in I^*$ can be characterized by having nonzero coordinates precisely on I's components, and zero coordinates elsewhere. Recall that $e_0 = \frac{1}{N}\sum_{x\in G} x$ and by Lemma 3.4, $\mathbb{F}_p[\mathbb{G}]e_1,\ldots,\mathbb{F}_p[\mathbb{G}]e_m$ have dimension at least $\mu_p(N)$, thus we have the following decomposition:

$$\mathbb{F}_p[\mathbb{G}] = \mathbb{F}_p[\mathbb{G}]e_0 \cup \Big(\bigcup_{\ell=\mu_p(N)}^{N} \bigcup_{I\in T_\ell} I^* \Big).$$

[7] In [FL22, Lemma 5.5], they prove the case $c = 2$ and imply that the general case also holds.

In particular, it is easy to see that $\mathbb{F}_p[\mathbb{G}]e_0 \simeq \mathbb{F}_p$ and $\dim(\mathbb{F}_p[\mathbb{G}]e_0) = 1$. Therefore, we have that

$$E\Big[\sum_{b\in\mathbb{F}_p[\mathbb{G}]\setminus\{0\}} X_b\Big] = \sum_{\ell=1}^{N}\sum_{I\in T_\ell}\sum_{b\in I^*} E[X_b]. \tag{2}$$

The size of T_ℓ can be bounded in terms of $\ell/\mu_p(N)$. Note that in our application, it satisfies $\mu_p(N) = 1$. However, we prove the general case, which might be of independent interest.

Lemma 3.13. *[FL22, Lemma 6.1]* $|T_\ell| \le N^{\lceil \frac{\ell}{\mu_p(N)} \rceil}$.

Proof. The proof is quite straightforward. For each ideal $I \in T_\ell$, I is a direct sum of some $\mathbb{F}_p[\mathbb{G}]e_i$ for $i = 0, 1, \ldots, m$. Except for $\mathbb{F}_p[\mathbb{G}]e_0$, each ideal is of dimension $\ge \mu_p(N)$ and thus there are at most $\lceil \frac{\ell}{\mu_p(N)} \rceil$ different summands. Thus $|T_\ell| \le \binom{m+1}{\lceil \frac{\ell}{\mu_p(N)} \rceil} \le \binom{N}{\lceil \frac{\ell}{\mu_p(N)} \rceil} \le N^{\lceil \frac{\ell}{\mu_p(N)} \rceil}$. □

Existing Bounds. We are now ready to bound the minimum distance of $\mathcal{C}_{a_1,\ldots,a_c}$. In [FL22], they derive a weaker bound for the distance of $\mathcal{C}_{a,a'}$ by directly incorporating Lemma 3.12, Lemma 3.13 and Eq.(2). Their analysis is sufficient for the regime $\mu_p(N) \ge (\log_p N)^2$, where low-dimensional ideals are effectively ruled out. However, in our setting, we have $\mu_p(N) = 1$ and the contribution of low dimension ideals becomes significant, leading to a substantially larger failure probability if one applies the same argument directly. For comparison, we first recall their result.

Lemma 3.14. *[FL22, Lemma 6.2] For $c = 2$, if $g_p(\delta) - \frac{1}{2} - \frac{\log_p N}{\mu_p(N)} > 0$, then* $E\left[\sum_{b\in\mathbb{F}_p[\mathbb{G}]\setminus\{0\}} X_b\right] \le p^{-2\mu_p(N)\left(g_p(\delta)-\frac{1}{2}-\frac{\log_p N}{\mu_p(N)}\right)}$.

Very recently, [LLX+26] improves [FL22, Lemma 6.2] by using tighter inequalities, and via tigher counter of ideals. Their result is stated as follows:

Lemma 3.15 ([LLX+26]). *Let $H_p(\cdot)$ be the p-ary entropy function and $c \in \mathbb{N}$. Let $\mathbb{G} = \mathbb{Z}_2^n$ with $N = 2^n$ and $\gcd(p, N) = 1$. Assume that $H_p(\delta) + \frac{1+\log_p N}{c} < 1$. Then, a random rank-1 quasi-$\mathbb{G}$ code $\mathcal{C} \subseteq (\mathbb{F}_p[\mathbb{G}])^c$ has relative minimum distance $\le \delta$ with probability*

$$\lessapprox N \cdot p^{-c(1-H_p(\delta)-1/c)} = p^{1+\log_p(N)-c(1-H_p(\delta))}.$$

Concretely, for $\mu_p(N) = 1$, $\delta = 0.1$, $N = 2^{20}$ and $\lceil \log_2 p \rceil = 128$, Lemma 3.14 yields $E\left[\sum_{b\in\mathbb{F}_p[\mathbb{G}]\setminus\{0\}} X_b\right] \approx 2^{-61.46}$ while Lemma 3.15 improves to a smaller failure probability $2^{-81.46}$. Note that the relative distance 0.1 matches Spielman's code [GLS+23] and EA code [BFK+24] used in existing SNARK constructions. Although these bounds already appear sufficient for practical SNARK applications, we will derive a substantially sharper estimate that significantly changes the current landscape. Concrete comparisons of these bounds are given in Table 3.

Our Tight Analysis. We need to improve Lemma 3.14 and also generalize the result from quasi-$\mathbb{G}$ code of index 2 to arbitrary index.

Overview. Our approach consists of three main ingredients. First, using Lemma 3.16, we show that every nonzero element contained in a low-dimensional ideal $I \subseteq \mathbb{F}_p[\mathbb{G}]$ necessarily has very large Hamming weight. With Lemma 3.16 at hand, at first glance one may think that if $\dim(\mathbb{F}_p[\mathbb{G}]b) = r$, then $\mathsf{wt}(ba_1 \dots ba_c) \geq cN/r$. However, this is not true. The caveat is that Lemma 3.16 only applies to nonzero elements of the ideal. There might be some bad cases that, e.g., $b \cdot a_i \neq 0$ for $i \in [1, c-1]$ but $b \cdot a_c = 0$, for which one can only guarantee $\mathsf{wt}(ba_1 \dots ba_c) \geq (c-1)N/r$.

To deal with the above issue, we prove Lemma 3.17 and Lemma 3.19, which bound the probability that the tuple $(ba_1, \dots, ba_c)$ contains at most one zero, or no zero at all. Combining these probabilistic bounds with Lemma 3.16, we show that, with overwhelming probability, every tuple $(ba_1, \dots, ba_c)$ has sufficiently large weight whenever b belongs to a low-dimensional ideal. For b in ideals of high dimension, we instead rely on the general expectation bound from Lemma 3.12. These already implies a substantially improved distance bound for our application.

Finally, we exploit an additional symmetry that improves the concrete failure probability. Observe that for every $b \in \mathbb{F}_p[\mathbb{G}]$ and $\beta \in \mathbb{F}_p^*$, $X_b = X_{\beta \cdot b}$ since multiplying by a nonzero scalar does not change the Hamming weight. This basically allows to sum $E[X_b]$ over $b \in (\mathbb{F}_p[\mathbb{G}] \backslash \{0\})/\mathbb{F}_p^*$, improving an multiplicative factor of $p-1$ in the final probability bound.

We now first establish the lower bound on the minimum distance of ideals with small dimensions.

Lemma 3.16. *Let $I \subseteq \mathbb{F}_p[\mathbb{G}]$ be an ideal of dimension r. Then, the minimum distance of I is at least $\frac{N}{r}$.*

Proof. We prove this lemma by contradiction. Recall that I is a group code over $\mathbb{F}_p$. Assume that there exists a non-zero codeword $a \in I$ of small weight $\delta N < \frac{N}{r}$, equivalently, $\delta < 1/r$. By the definition of ideal, for any $x \in \mathbb{G}$, $a \cdot x \in I$ as well (here x is viewed as an group ring element $1x \in \mathbb{F}_p[\mathbb{G}]$). Moreover, $\mathsf{wt}(a \cdot x) = \mathsf{wt}(a) = \delta N$. Since I is a group code, each $y \in \mathbb{G}$ is in the support of some codeword $a \cdot x$ for $x \in \mathbb{G}$. Therefore, we can find at least $L := \lceil 1/\delta \rceil$ distinct codewords $ax_1, \dots, ax_L \in I$ such that the support set of ax_i is not contained by the union of the support sets of $ax_1, \dots, ax_{i-1}$.

We claim that these L codewords are $\mathbb{F}_p$-linearly independent. For each i, there exists a position k that lies in the support of ax_i but not in the support of ax_j for any $j < i$. Then if some linear combination $\sum_{j=1}^{i} \alpha_j \cdot ax_j$ of $ax_1, .., ax_i$ equals zero, the coefficient of $\alpha_j \in \mathbb{F}_p$ must be zero because the linear combination at position k is zero. By induction, all coefficients α_r are zero. This completes the claim.

Thus, the dimension of I is at least $L = \lceil 1/\delta \rceil > r$ if I contains a codeword of weight $\delta N < \frac{N}{r}$. However, since we assume I has dimension r, a contradiction happens. The proof is completed. □

Note that if ba_i is nonzero and $b \in I$ for some r-dimensional ideal, then Lemma 3.16 implies $\mathsf{wt}(ba_i) \geq \frac{N}{r}$. Consequently, if most of $(ba_1, \ldots, ba_c)$ are nonzero, then the tuple must have large total weight. Motivated by this, we next bound the probability that, for a fixed nonzero element $b \in \mathbb{F}_p[\mathbb{G}]$, that the tuple $(ba_1, \ldots, ba_c)$ has at most 1 zero.

Lemma 3.17. *Let $a_1, \ldots, a_c$ be sampled independently and uniformly from $\mathbb{F}_p[\mathbb{G}]$. The probability that, for any nonzero $b \in \mathbb{F}_p[\mathbb{G}]$ there is at most one zero among $ba_1, \ldots, ba_c$, is at least $1 - \frac{c(c-1)N}{2p^2}$.*

Proof. By Theorem 3.3, it holds that

$$\mathbb{F}_p[\mathbb{G}] = \mathbb{F}_p[\mathbb{G}]e_0 \oplus \cdots \oplus \mathbb{F}_p[\mathbb{G}]e_m,$$

where e_i is the primitive idempotent and $\dim(\mathbb{F}_p[\mathbb{G}]e_i) = d_i$. Since e_i are primitive idempotents (Definition 3.2), every $b \in \mathbb{F}_p[\mathbb{G}]$ admits a unique decompostion with respect to $(e_0, \ldots, e_m)$. Moreover, for every $a \in \mathbb{F}_p[\mathbb{G}]$, $ae_i \in \mathbb{F}_p[\mathbb{G}]e_i$ and $ab \in \mathbb{F}_p[\mathbb{G}]e_0 \oplus \cdots \oplus \mathbb{F}_p[\mathbb{G}]e_m$. Thus to prove the statement for all nonzero b, it suffices to consider the $m+1$ primitive ideompotents $e_0, \ldots, e_m$. Fix one e_i. Since multiplication by e_i is the projection onto the ideal $\mathbb{F}_p[\mathbb{G}]e_i$, if a is uniformly distriburted over $\mathbb{F}_p[\mathbb{G}]$, then ae_i is uniformly distributed in $\mathbb{F}_p[\mathbb{G}]e_i$. Hence, for each $j \in [c]$, $\Pr[a_j e_i = 0] = p^{-d_i}$ and these events are independent. The probability that there are at least two zeros in $a_1 e_i, ..., a_c e_i$ is

$$\begin{aligned}
&1 - (1 - p^{-d_i})^c - cp^{-d_i}(1 - p^{-d_i})^{c-1} \\
\leq &1 - (1 - \frac{c}{p^{d_i}} + \frac{c(c-1)}{2p^{2d_i}}) - \frac{c}{p^{d_i}}(1 - \frac{c-1}{p^{d_i}}) \\
= &\frac{c(c-1)}{2p^{2d_i}}.
\end{aligned}$$

Since $d_i \geq 1$, the above probability is upper bounded by $\frac{c(c-1)}{2p^2}$. Finally, taking a union bound over $e_0, \ldots, e_m$ with $m + 1 \leq N$, we conclude that the probability that there is at most one zero among $ba_1, \ldots, ba_c$ is at least $1 - \frac{c(c-1)N}{2p^2}$. □

We are ready to prove the main theorem.

Theorem 3.18. *Let $\mathbb{G}$ be an abelian group of order N and $\gcd(p, N) = 1$. Assume that $\epsilon = g_p(\delta) - \frac{1}{c} - \frac{\log_p N}{c}\lceil \frac{1}{\mu_p(N)} \rceil > 0$. Then, with probability at least*

$$1 - \frac{c(c-1)N}{2p^2} - \frac{p^{-\lceil \frac{c-1}{c\delta} \rceil c\epsilon}}{(1 - p^{-c\epsilon})(p-1)},$$

a random quasi-$\mathbb{G}$ code $\mathcal{C}_{a_1,\ldots,a_c} \subseteq (\mathbb{F}_p[\mathbb{G}])^c$ is an $\mathbb{F}_p$-linear code of rate $1/c$, length cN and relative minimum distance at least δ.

Proof. By Lemma 3.17, with probability at least $1 - \frac{c(c-1)N}{2p^2}$, there are at most 1 zero in $ba_1, \ldots, ba_c$ for any nonzero b. Thus for $b \in \mathbb{F}_p[\mathbb{G}]$, $(ba_1, \ldots, ba_c)$ are all

distinct, meaning that $\mathcal{C}_{a_1,\ldots,a_c}$ has dimension N and rate $\frac{1}{c}$. Condition on this event that there is at most one zero among $ba_1, \ldots, ba_c$ from now on.

Recall that we define the random binary variable X_b for $b \in \mathbb{F}_p[\mathbb{G}]$ such that

$$X_b = 1 \text{ if } 0 < \mathsf{wt}(ba_1, \ldots, ba_c) < cN\delta, \text{ and } X_b = 0 \text{ otherwise.}$$

To bound the minimum distance probability, a direct way is to bounding

$$E\Big[\sum_{b\in\mathbb{F}_p[\mathbb{G}]\setminus\{0\}} X_b\Big] = \sum_{\ell=1}^{N}\sum_{I\in T_\ell}\sum_{b\in I^*} E[X_b].$$

Let us treat the above equation with more care. Let $I \subseteq \mathbb{F}_p[\mathbb{G}]$ be an ideal of dimension less than $\frac{c-1}{c\delta}$. By Lemma 3.16, we have $\mathsf{wt}(ba_1, \ldots, ba_c) \geq (c-1)\cdot\frac{Nc\delta}{c-1} = cN\delta$. Thus, $X_b = 0$ in this case. It therefore suffices to consider the element $b \in I$ of dimension at least $t := \lceil \frac{c-1}{c\delta} \rceil$. That is,

$$E\Big[\sum_{b\in\mathbb{F}_p[\mathbb{G}]\setminus\{0\}} X_b\Big] = \sum_{\ell=t}^{N}\sum_{I\in T_\ell}\sum_{b\in I^*} E[X_b].$$

Beyond that, we can further utilize symmetry to obtain a sharper bound. The key is that in the proof of Lemma 3.12, we are essentially counting the size of a set consisting of low-weight codewords $(ba_1, \ldots, ba_c)$ for a fixed b over the randomness of $a_1, \ldots, a_c$. Observe that for every nonzero scalar $\beta \in \mathbb{F}_p^*$, it holds that

$$X_{\beta b} = X_b.$$

Indeed, b and βb differ only by a nonzero scalar multiple β, and therefore contribute to the *same* low-weight codewords. Hence it suffices to count one representative from $\{\beta b : \beta \in \mathbb{F}_p^*\}$ for $b \in \mathbb{F}_p[\mathbb{G}]\setminus\{0\}$. To this end, let $0 \notin \Gamma \subseteq \mathbb{F}_p[\mathbb{G}]$ be a set of size $\frac{p^N-1}{p-1}$ such that for any pair $b, b' \in \Gamma$ there does not exist $\beta \in \mathbb{F}_p^*$ satisfying $b = \beta b'$. Now we have

$$\Gamma = \bigcup_{0\neq I\leq\mathbb{F}_p[\mathbb{G}]} (I^* \cap \Gamma),$$

and we are aimed at bounding the following probability

$$E\Big[\sum_{b\in\Gamma} X_b\Big] = \sum_{\ell=t}^{N}\sum_{I\in T_\ell}\sum_{b\in I^*\cap\Gamma} E[X_b].$$

By Lemma 3.12, if $b \in I$ with $\dim(I) \geq t$, then $E[X_b] \leq p^{-cd_b g_p(\delta)}$. As $0 \notin I^*$, I^* has at most $p^\ell - 1$ elements. By intersecting I^* with Γ, we are essentially taking representatives from I^*. So the size of $|I^* \cap \Gamma|$ is at most $(p^\ell - 1)/(p-1)$. Therefore,

$$\sum_{b\in I^*\cap\Gamma} E\left[X_b\right] \leq \frac{p^\ell - 1}{p-1}\cdot p^{-c\ell g_p(\delta)} \leq \frac{p^{-c\ell(g_p(\delta)-\frac{1}{c})}}{p-1}.$$

Applying Lemma 3.13, we obtain

$$\sum_{\ell=t}^{N}\sum_{I\in T_\ell}\sum_{b\in I^*\cap\Gamma} E[X_b] \le \sum_{\ell=t}^{N}|T_\ell|\cdot\frac{p^{-c\ell(g_p(\delta)-\frac{1}{c})}}{p-1}$$
$$\le \sum_{\ell=t}^{N} N^{\lceil \ell/\mu_p(N)\rceil}\frac{p^{-c\ell(g_p(\delta)-\frac{1}{c})}}{p-1} \le \sum_{\ell=t}^{N}\frac{p^{-c\ell(g_p(\delta)-\frac{1}{c}-\frac{\log_p N}{c}\lceil\frac{1}{\mu_p(N)}\rceil)}}{p-1}$$
$$\le \frac{p^{-ct(g_p(\delta)-\frac{1}{c}-\frac{\log_p N}{c}\lceil\frac{1}{\mu_p(N)}\rceil)}}{(1-p^{-c(g_p(\delta)-\frac{1}{c}-\frac{\log_p N}{c}\lceil\frac{1}{\mu_p(N)}\rceil)})(p-1)}.$$

The desired result follows as $t=\lceil\frac{c-1}{c\delta}\rceil$ and $\epsilon = g_p(\delta)-\frac{1}{c}-\frac{\log_p N}{c}\lceil\frac{1}{\mu_p(N)}\rceil$. □

If p is sufficiently large compared to N, we may expect that $ba_1,\ldots,ba_c$ are all non-zero over $\mathbb{F}_p[\mathbb{G}]$. Using essentially the same argument as in Lemma 3.17, we conclude that:

Lemma 3.19. *Let $a_1,\ldots,a_c$ be random elements in $\mathbb{F}_p[\mathbb{G}]$. The probability that for any nonzero $b\in\mathbb{F}_p[\mathbb{G}]$, $ba_1,\ldots,ba_c$ are all non-zero is at least $1-\frac{cN}{p}$.*

Lemma 3.17 and Lemma 3.19 state the probability that the number of zeros among $ba_1,\ldots,ba_c$ is ≤ 1 and 0, respectively. By Lemma 3.19, the argument in Theorem 3.18 can start with an ideal of dimension at least $\lceil\frac{1}{\delta}\rceil$. We summarize this result in the following theorem.

Theorem 3.20. *Let $\mathbb{G}$ be a abelian group of order N and $\gcd(p,N)=1$. Assume that $\epsilon = g_p(\delta)-\frac{1}{c}-\frac{\log_p N}{c}\lceil\frac{1}{\mu_p(N)}\rceil>0$. Then, with probability at least*

$$1-\frac{cN}{p}-\frac{p^{-\lceil\frac{1}{\delta}\rceil c\epsilon}}{(1-p^{-c\epsilon})(p-1)},$$

a random quasi-$\mathbb{G}$ code $\mathcal{C}_{a_1,\ldots,a_c}\subseteq(\mathbb{F}_p[\mathbb{G}])^c$ is an $\mathbb{F}_p$-linear code of rate $\frac{1}{c}$, length cN and relative minimum distance at least δ.

In our applications, we are particularly interested in the case $\mathbb{G}=\mathbb{Z}_2^n$. For every odd prime p, we have $\mu_p(2^n)=1$. Substituting this into Theorem 3.18 and Theorem 3.20 yields the following corollary.

Corollary 3.21. *Let $\mathbb{G}=\mathbb{Z}_2^n$ with $N:=2^n$ and $\gcd(p,N)=1$. Assume that $\epsilon = g_p(\delta)-\frac{1+\log_p N}{c}>0$. Then, with probability at least*

$$\max\left\{1-\frac{c(c-1)N}{2p^2}-\frac{p^{-\lceil\frac{c-1}{c\delta}\rceil c\epsilon}}{(1-p^{-c\epsilon})(p-1)},\quad 1-\frac{cN}{p}-\frac{p^{-\lceil\frac{1}{\delta}\rceil c\epsilon}}{(1-p^{-c\epsilon})(p-1)}\right\},$$

a random quasi-$\mathbb{G}$ code $\mathcal{C}_{a_1,\ldots,a_c}\subseteq(\mathbb{F}_p[\mathbb{G}])^c$ is an $\mathbb{F}_p$-linear code of rate $\frac{1}{c}$, length cN and relative minimum distance at least δ.

Remark 3.22. Our current distance analyses focus on QA codes with rank-1 and arbitrary index-c. It is principally possible to extend to QA codes of higher ranks. We simply leave it as future work as in our applications the rank-1 QA codes turn out to be more efficient and useful.

3.3 Concrete Minimum Distance

Table 2 summarizes the minimum distance of (rank-1) QA codes over $\mathbb{F}_p[\mathbb{Z}_2^n]$ for various parameters including group size $N = 2^n$, field size p, and rate $1/c$. For each parameter setting, the relative distance δ of the quasi-abelian code, together with the Gilbert-Varshamov (GV) bounds over $\mathbb{F}_p$ and $\mathbb{F}_2$ are reported.

Table 2: Minimum Distance of QA codes over $\mathbb{F}_p[\mathbb{Z}_2^n]$

$N = 2^n$	$\lceil \log p \rceil$	Index c	Rate r	Distance δ	GV over $\mathbb{F}_p$	GV over $\mathbb{F}_2$	Probability
2^{15}	128	2	1/2	0.4336	0.4921	0.1100	$1 - 2^{-127}$
		3	1/3	0.6201	0.6594	0.1739	
		4	1/4	0.7139	0.7435	0.2144	
		5	1/5	0.7704	0.7942	0.2430	
2^{20}	128	2	1/2	0.4142	0.4921	0.1100	$1 - 2^{-127}$
		3	1/3	0.6070	0.6594	0.1739	
		4	1/4	0.7040	0.7435	0.2144	
		5	1/5	0.7625	0.7942	0.2430	
2^{25}	128	2	1/2	0.3947	0.4921	0.1100	$1 - 2^{-127}$
		3	1/3	0.5939	0.6594	0.1739	
		4	1/4	0.6942	0.7435	0.2144	
		5	1/5	0.7546	0.7942	0.2430	
2^{15}	256	2	1/2	0.4668	0.4960	0.1100	$1 - 2^{-255}$
		3	1/3	0.6434	0.6630	0.1739	
		4	1/4	0.7320	0.7468	0.2144	
		5	1/5	0.7853	0.7971	0.2430	
2^{20}	256	2	1/2	0.4570	0.4960	0.1100	$1 - 2^{-255}$
		3	1/3	0.6369	0.6630	0.1739	
		4	1/4	0.7271	0.7468	0.2144	
		5	1/5	0.7814	0.7971	0.2430	
2^{25}	256	2	1/2	0.4472	0.4960	0.1100	$1 - 2^{-255}$
		3	1/3	0.6304	0.6630	0.1739	
		4	1/4	0.7222	0.7468	0.2144	
		5	1/5	0.7774	0.7971	0.2430	

The table shows that with overwhelming probability, the relative distance of quasi-abelian codes is consistently close to the GV bound over $\mathbb{F}_p$ (up to a small constant gap) and significantly larger than the GV bound over $\mathbb{F}_2$. Particularly, increasing p from 128-bit to 256-bit yields a noticeable gain in δ i.e., δ is much closer to the GV bound over $\mathbb{F}_p$. As the rate r decreases, the relative distance improves consistently across all parameter settings. Moreover, the dependence on the group size N is mild and the code achieves good asymptotic properties. In summary, Table 2 demonstrates that QA codes over $\mathbb{F}_p[\mathbb{Z}_2^n]$ achieve strong and practical distance guarantees.

Table 3 compares our bounds with existing analyses in [FL22, LLX+26], where the numbers are derived from Proposition 3.14, Proposition 3.15, and Theorem 1.1, respectively. This table shows that our analyses significantly outperform

[FL22, LLX+26] over arbitrarily large fields in terms of both distance and failure probability.

Table 3: Comparison of Distance given by Different Bounds

$N = 2^n$	$\lceil \log p \rceil$	Rate r	Method	Distance δ	Failure Probability
2^{20}	256	1/2	[FL22]	0.2	$2^{-112.15}$
			[LLX+26]	0.22	$2^{-122.83}$
			Ours	0.4668	2^{-255}
2^{20}	128	1/2	[FL22]	0.1	$2^{-61.46}$
			[LLX+26]	0.1	$2^{-81.46}$
			Ours	0.4142	2^{-127}
2^{20}	64	1/2	[FL22]	0.1	$2^{-10.26}$
			[LLX+26]	0.1	$2^{-30.26}$
			Ours	0.2376	2^{-100}
			Ours	0.3294	2^{-63}

4 Fast Encoding of Quasi-Abelian Codes

4.1 Walsh-Hadmard Transform

Basically speaking, the Walsh-Hadamard transform can be seen as the FFT for the evaluation set $\{1, -1\}^n$. The computation complexity of classic FFT is dominated by the number of multiplications in $\mathbb{F}_p$. For Walsh-Hadamard transform, only additions and subtractions are used and thus the computation complexity is completely determined by the number of additions or subtractions in $\mathbb{F}_p$. This advantage will be amplified if the field size is very large. In this section, we will introduce Walsh-Hadamard transform algorithm and the fast encoding algorithm of Quasi-Abelian code based on Walsh-Hadamard transform.

Let $\vec{f} = (f_0, \ldots, f_{N-1}) \in \mathbb{F}_p^N$ be the coefficient of $f \in \mathcal{R} := \mathbb{F}_p[\mathtt{X}_1, \ldots, \mathtt{X}_n]/(\mathtt{X}_1^2 - 1, \ldots, \mathtt{X}_n^2 - 1)$. The Walsh-Hadamard transform is to compute $\widehat{f} \in \mathbb{F}_p^N$ which is the evaluation of $f(\mathtt{X})$ at N points in $\{1, -1\}^n$. Write

$$\widehat{f} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes n} \vec{f}.$$

where the $H^{\otimes n}$ represents n tensor products of H. This tensor product can be implemented extremely fast via the Walsh-Hadamard transform in Algorithm 1.

Theorem 4.1. *Algorithm 1 consumes $N \log N$ additions or subtractions in $\mathbb{F}_p$.*

Proof. Let T_n be the total number of additions or subtractions to compute $\mathsf{WHT}(f)$ for $f \in \mathcal{R}$. Recall that f is an n-variable polynomial and $N = 2^n$. From the description of Algorithm 1, we can deduce that

$$T_n = 2T_{n-1} + 2^n = \cdots = 4T_{n-2} + 2 \times 2^n = \cdots = n \times 2^n = N \log N.$$

The proof is completed. □

Algorithm 1 $\mathsf{WHT}(f)$

On input a polynomial $f \in \mathcal{R}$ with n variables. Let $\vec{f} = (f_0, \ldots, f_{N-1}) \in \mathbb{F}_p^N$ denote the coefficients of f.
if $N = 1$ **then**
  Return f_0.
else
1. Parse $\vec{f}$ into two equal-length parts $\vec{f}_L = (f_0, \ldots, f_{N/2-1})$ and $\vec{f}_R = (f_{N/2}, \ldots, f_{N-1})$.
2. Set $\vec{y}_L = \mathsf{WHT}(\vec{f}_L)$ and $\vec{y}_R = \mathsf{WHT}(\vec{f}_R)$.
3. Return $(\vec{y}_L + \vec{y}_R, \vec{y}_L - \vec{y}_R)$.

end if

Lemma 4.2. *$f \in \mathcal{R}$ with n variables. Let $\vec{f} = (f_0, \ldots, f_{N-1}) \in \mathbb{F}_p^N$ denote the coefficients of f. Then, $\mathsf{WHT}(\mathsf{WHT}(f)) = 2^n \vec{f}$. That is, $\mathsf{WHT}^{-1} = 2^{-n}\mathsf{WHT}$.*

Proof. The proof is straightforward. Note that $\mathsf{WHT}(f) = H^{\otimes n}\vec{f}$ where $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. Thus

$$\mathsf{WHT}(\mathsf{WHT}(f)) = H^{\otimes n} \times H^{\otimes n}\vec{f} = (H \times H)^{\otimes n}\vec{f} = 2^n\vec{f},$$

as $H \times H = 2I_n$ where I_n is the identity matrix of order n. □

4.2 Fast Encoding of Quasi-Abelian Codes

Our encoding of QA codes is based on the Walsh-Hadamard transform. Recall that a quasi-$\mathbb{G}$ code of rank 1 and index c has the following form

$$\mathcal{C}_{a_1,\ldots,a_c} = \{(ba_1, \ldots, ba_c) : b \in \mathbb{G}\} \subseteq (\mathbb{F}_p[\mathbb{G}])^c.$$

In this subsection, we let $\mathbb{G} = \mathbb{Z}_2^n$ and thus $\mathbb{F}_p[\mathbb{Z}_2^n]$ is nothing but the polynomial ring $\mathbb{F}_p[\mathtt{X}_1, \ldots, \mathtt{X}_n]/(\mathtt{X}_1^2-1, \ldots, \mathtt{X}_n^2-1)$. We treat b as the message and $(ba_1, \ldots, ba_c)$ as the codeword in our encoding algorithm of $\mathcal{C}_{a_1,\ldots,a_c}$. Note that if we compute the product ba_i directly by computing each coefficient of ba_i, then the computational complexity of our encoding algorithm is quadratic in code length. Clearly, it is not efficient as we do not utilize any algebraic structure of group $\mathbb{Z}_2^n$. To make our encoding algorithm extremely efficient, we resort to the Walsh-Hadamard transform.

Theorem 4.3. *Algorithm 2 is an encoding algorithm of $\mathcal{C}_{a_1,\ldots,a_c} \subseteq (\mathbb{F}_p[\mathbb{Z}_2^n])^c$ for some odd p. This encoding algorithm consumes $(c+1)n2^n$ additions/subtractions and $c2^n$ multiplications.*

Proof. The correctness of Algorithm 2 is due to Lemma 4.2. More precisely, observe that the evaluations of ba_i are the product of the evaluation of b and the

Algorithm 2 Encoding Algorithm of $\mathcal{C}_{a_1,\ldots,a_c}$

On input message $b \in \mathbb{F}_p[\mathbb{Z}_2^n]$ and the Walsh-Hadamard transform of $2^{-n}\cdot a_1, \ldots, 2^{-n}\cdot a_c$, which are denoted as $2^{-n}\widehat{a}_1, \ldots, 2^{-n}\widehat{a}_c$.

1. Compute $\widehat{b} = \mathsf{WHT}(b)$.
2. Compute $\vec{b}_i := \widehat{b} \star (2^{-n}\cdot \widehat{a}_i)$ for $i = 1, \ldots, c$, where $\star$ stands for component-wise product.
3. Compute $b_i = \mathsf{WHT}(\vec{b}_i)$ for $i = 1, \ldots, c$.

Output $(b_1, \ldots, b_c)$.

evaluation of a_i, i.e., $\mathsf{WHT}(ba_i) = \mathsf{WHT}(b) \star \mathsf{WHT}(a_i)$. Then, by Lemma 4.2, this implies

$$ba_i = 2^{-n}\cdot \mathsf{WHT}\big(\mathsf{WHT}(ba_i)\big) = 2^{-n}\cdot \mathsf{WHT}(\widehat{b}\star\widehat{a}_i) = \mathsf{WHT}\big(\widehat{b}\star(2^{-n}\cdot\widehat{a}_i)\big),$$

where the last equality follows from WHT is a linear map and multiplying by 2^{-n} is scalar multiplication. Moreover, during the encoding process, we call Walsh-Hadamard transform $c+1$ times and thus this encoding algorithm requires $(c+1)n2^n$ additions/subtractions. Clearly, the number of multiplications is $c2^n$. □

Optimizations. In the following, we introduce several optimizations for further improving the concrete efficiency.

1. We do not need to pre-compute Walsh-Hadamard transform of $2^{-n}a_1, \ldots, 2^{-n}a_c$. Since $a_1, \ldots, a_c$ are random elements in $\mathbb{F}_p[\mathbb{Z}_2^n] \simeq \mathbb{F}_p^{2^n}$, $2^{-n}\cdot\widehat{a}_1, \ldots, 2^{-n}\cdot\widehat{a}_c$ are also distributed uniformly at random in $\mathbb{F}_p^{2^n}$. Thus, it suffices to sample c random vectors in $\mathbb{F}_p^{2^n}$, and directly view them as $2^{-n}\cdot\widehat{a}_1, \ldots, 2^{-n}\cdot\widehat{a}_c$.
2. We can use a systematic form, i.e., fixing $a_1 = 1$, and sampling $c-1$ random vectors $2^{-n}\cdot\widehat{a}_2, \ldots, 2^{-n}\cdot\widehat{a}_c$. Note that in our case $\mathbb{F}_p$ will be substantially larger than 2^n, thus with high probability a random element in $\mathbb{F}_p[\mathbb{Z}_2^n]$ is invertible. For a random QA code given in this way, Theorem 3.20 suffices for bounding the relative minimum distance.

The optimized encoding algorithm is given in Algorithm 3.

Algorithm 3 Encoding Algorithm of $\mathcal{C}_{1,a_2\ldots,a_c}$

On input message $b \in \mathbb{F}_p[\mathbb{Z}_2^n]$ and $2^{-n}\cdot\widehat{a}_2, \ldots 2^{-n}\cdot\widehat{a}_c \in \mathbb{F}_p^{2^n}$, which correspond to the Walsh-Hadamard transform of $2^{-n}a_2, \ldots, 2^{-n}a_c \in \mathbb{F}_p[\mathbb{Z}_2^n]$, respectively.

1. Compute $\widehat{b} = \mathsf{WHT}(b)$.
2. Compute $\vec{b}_i := \hat{b} \star (2^{-n}\cdot\widehat{a}_i)$ for $i = 2, \ldots, c$, where $\star$ stands for component-wise product.
3. Compute $b_i^* := \mathsf{WHT}(\vec{b}_i)$ for $i = 2, \ldots, c$.

Output $(b, b_2^*, \ldots, b_c^*)$.

Theorem 4.4. *$\mathcal{C}_{1,a_2\ldots,a_c}$ given by Algorithm 3 is a linear code over $\mathbb{F}_p$, and $\mathcal{C}_{1,a_2\ldots,a_c} = \mathcal{C}_{a_1,a_1a_2,\ldots,a_1a_c}$ for any invertible $a_1 \in \mathbb{F}_p[\mathbb{G}]$. This means we can apply Theorem 3.20 to bound the distance of a random systematic QA code $\mathcal{C}_{1,a_2\ldots,a_c}$.*

Proof. We observe that for any nonzero codeword $(b, ba_2, \ldots, ba_c) \in \mathcal{C}_{a_1,a_1a_2,\ldots,a_1a_c}$,

$$(b, ba_2, \ldots, ba_c) = ba_1^{-1}(a_1, a_1a_2, \ldots, a_1a_c) \in \mathcal{C}_{a_1,a_1a_2,\ldots,a_1a_c},$$

as a_1 is invertible. Note that $a_2, \ldots, a_c$ are random elements in $\mathbb{F}_p[\mathbb{G}]$. Thus we can use the same bound for random QA codes $\mathcal{C}_{a'_1,\ldots,a'_c}$ to bound the distance of $\mathcal{C}_{1,a_2\ldots,a_c}$ conditioning on that a'_1 is invertible. The correctness of the encoding algorithm is straightforward. □

Corollary 4.5. *$\mathcal{C}_{1,a_1,\ldots,a_c}$ given by Algorithm 3 takes $(c-1)\cdot 2^n$ multiplications and $cn \cdot 2^n$ additions/subtractions in $\mathbb{F}_p$.*

5 Polynomial Commitments and SNARKs from Quasi-Abelian Codes

We build polynomial commitments as well as SNARKs from Quasi-Abelian codes following a modular approach as in previous code-based SNARKs. The framework generally works for any linear code that has a provable constant relative minimum distance, and we basically instantiate it with our QA codes. For completeness, we present a backbone construction of the PCS in Π_{PCS}. Due to space limitations, we refer the readers to [GLS+23] for more details.

Without loss of generality, we consider a prime field $\mathbb{F}_p$, in which we can use a QA code over $\mathbb{F}_p[\mathbb{Z}_2^n]$ with a constant index c. Suppose a prover commits to a multi-linear polynomial $f(X_1, \ldots, X_{2n}) \in \mathbb{F}_p[X_1, \ldots, X_{2n}]$, and later is required to open an evaluation of f at $\mathbf{x} = (x_1, \ldots, x_{2n})$. Let $N = 2^{2n}$. Brakedown's PCS utilizes an observation that a polynomial evaluation can be computed from an inner-product $\left\langle \vec{f}, \vec{x}_L \otimes \vec{x}_R \right\rangle$, where $\vec{f} \in \mathbb{F}_p^N$ denotes the coefficients of f, $\vec{x}_L, \vec{x}_R \in \mathbb{F}_p^{\sqrt{N}}$ are determined by $\mathbf{x}$, and $\otimes$ denotes the tensor-product operation. Furthermore, $\vec{f}$ can be re-arranged into a matrix in $\mathbb{F}_p^{\sqrt{N}\times\sqrt{N}}$, denoted by $\mathsf{mat}(\vec{f})$, such that $\left\langle \vec{f}, \vec{x}_L \otimes \vec{x}_R \right\rangle = \vec{x}_L^{\mathsf{T}} \cdot \mathsf{mat}(\vec{f}) \cdot \vec{x}_R$.

Protocol 4: Π_{PCS}

Let $N = 2^{2n}$, and suppose $\mathcal{P}$ wants to commit to a multi-linear polynomial $f(X_1, \ldots, X_{2n}) \in \mathbb{F}_p[X_1, \ldots, X_{2n}]$. Let QA be a random QA code over $\mathbb{F}_p[\mathbb{Z}_2^n]$ with a constant index c, and $\mathsf{QA.Enc}$ denotes the encoding algorithm of it.

Commit Phase:

1. $\mathcal{P}$ encodes each row of $\mathsf{mat}(\vec{f})$ by $\mathsf{QA.Enc}$ and obtains a matrix $G \in \mathbb{F}_p^{2^n \times c\cdot 2^n}$.

2. $\mathcal{P}$ commits to all entries of G via Merkle-Tree Hash (first hashing every column of G to obtain $c \cdot 2^n$ roots, and second hashing these $c \cdot 2^n$ roots to obtain the final Merkle-Tree root). $\mathcal{P}$ sends the Merkle-Tree root to $\mathcal{V}$.

Open Phase:

1. $\mathcal{V}$ selects a random $\vec{r} \xleftarrow{\$} \mathbb{F}_p^{2^n}$, and an evaluation point parsed as $\vec{x}_L, \vec{x}_R \in \mathbb{F}_p^{2^n}$. $\mathcal{V}$ sends $\vec{r}, \vec{x}_L, \vec{x}_R$ to $\mathcal{P}$.
2. $\mathcal{P}$ computes $\mathbf{v} := \vec{r}^{\intercal} \cdot \mathsf{mat}(\vec{f}) \in \mathbb{F}_p^{2^n}$, and $\mathbf{v}' := \vec{x}_L^{\intercal} \cdot \mathsf{mat}(\vec{f}) \in \mathbb{F}_p^{2^n}$. $\mathcal{P}$ sends $\mathbf{v}, \mathbf{v}'$ to $\mathcal{V}$.
3. $\mathcal{V}$ encodes $\mathbf{v}, \mathbf{v}'$ via $\mathsf{QA.Enc}$, and obtains a codeword $\mathbf{u}, \mathbf{u}' \in \mathbb{F}_p^{c \cdot 2^n}$. Then, $\mathcal{V}$ selects random $T \subset [1, c \cdot 2^n]$, with $|T| = t$, and asks $\mathcal{P}$ to open columns of G with respect to T.
4. $\mathcal{P}$ sends $G|_T$ and the corresponding paths of the Merkle-Tree Hash to $\mathcal{V}$.
5. $\mathcal{V}$ checks the validity of all the Merkle-Tree Hash openings. In addition, $\mathcal{V}$ checks $\mathbf{u}_j \stackrel{?}{=} \langle \vec{r}, G|_j \rangle$, and $\mathbf{u}'_j \stackrel{?}{=} \langle \vec{x}_L, G|_j \rangle$ for all $j \in T$.
6. If all the above checks pass, $\mathcal{V}$ accepts and outputs $f(\mathbf{x}) := \langle \mathbf{v}', \vec{x}_R \rangle$. Otherwise, $\mathcal{V}$ rejects.

Theorem 5.1. *The above protocol Π_{PCS} is a polynomial commitment scheme with the following properties:*

- *Soundness error [GLS+23]:* $\frac{N}{|\mathbb{F}_p|} + (1 - \frac{\delta}{3})^t + (1 - \frac{2\delta}{3})^t$;
- *Proof size:* $(t+2)\sqrt{N}$ *elements in* $\mathbb{F}_p$ *and* $O(t \log N)$ *Hash values;*
- *Prover time:* $O(N \log \sqrt{N})$ *additions and* $O(N)$ *multiplications in* $\mathbb{F}_p$*, and* $O(N)$ *Hash operations;*
- *Verifier time:* $O(\sqrt{N} \log N)$ *additions and* $O(t\sqrt{N})$ *multiplications in* $\mathbb{F}_p$ *as well as* $O(t\sqrt{N})$ *Hash operations,*

where δ is the relative minimum distance of the QA code, and t is the number of opened columns.

Theorem 5.1 establishes the relation between minimum distance and the soundness error, where a larger minimum distance allows choosing a smaller t, which also improves the complexity. For a typical $N = 2^{30}$ and a sufficiently large field, one can set $t = \frac{-\lambda}{\log_2(1-\delta/3)}$ to realize λ-bit security. In Table 5 we show suggested parameters for our PCS and compare them to those in [BFK+24, GLS+23]. Since our QA code enjoys a much better provable minimum distance than other codes used in the literature, we can choose a much smaller t, yielding PCS with much better concrete efficiency. Concrete evaluations are reported in , where the code is incorporated from [BFK+24].

We remark that our implementation based on the source code of [BFK+24] inherits an optimization that improves the proof size from $O(t\sqrt{N})$ to $O(\sqrt{tN})$. This is achieved by optimizing the dimensions of the coefficient matrix, such that the column size is $\approx t/2$ times smaller than the row size.

Table 5: Parameter Choices for PCS over $\mathbb{F}_p$ from Fast Codes

$\sqrt{N}$	$\lceil \log p \rceil$	Code	Rate r	Distance δ	t	λ	Failure Error
2^{15}	128	Brakedown	1/2	0.1	2004	97	2^{-100}
		EA	1/2	0.1	2004	97	2^{-100}
		EA (conjectured)	1/2	0.2499	781	97	N/A
		QA	1/2	0.4336	431	97	2^{-127}
			1/3	0.6201	291	97	2^{-127}
			1/4	0.7139	248	97	2^{-127}
			1/5	0.7704	227	97	2^{-127}
2^{15}	256	Brakedown	1/2	0.1	2617	128	2^{-100}
		EA	1/2	0.1	2617	128	2^{-100}
		EA (conjectured)	1/2	0.2499	1020	128	N/A
		QA	1/2	0.4668	525	128	2^{-255}
			1/3	0.6434	368	128	2^{-255}
			1/4	0.7320	318	128	2^{-255}
			1/5	0.7853	293	128	2^{-255}

6 Experiments and Results

We implement our PCS based on quasi-abelian codes by extending the Rust implementation of [BFK+24], which is built upon Spartan [Set20] and provides PCS from EA code and Spielman's code. We then incorporate these code-based PCS with Spartan's IOP, obtaining NIZKs with sublinear proof size from Spielman's code, EA code and QA code. These code-based NIZKs can be directly compiled into SNARKs by using a PCS for *sparse* polynomials [Set20], which can be further constructed from our PCS for dense polynomials by the Spark compiler of [Set20].

In Section 6.1, we report encoding time compared to the Brakedown encoder [GLS+23] (i.e., Spielman's code [Spi96, DI14]) with improved parameters [Hab23], EA codes [BFK+24], and RAA codes [BCF+25]. In Section 6.2, we compare our PCS with the PCS constructions from Brakedown [GLS+23], and EA codes [BFK+24], considering both provable distance and conjectured distance instantiations. In Section 6.3, we report the performance of code-based NIZKs, which are concretely more efficient than their SNARK versions in our tested parameter regime. Finally, in Section 6.4, we compare code-based NIZKs and SNARKs to Groth16 [Gro16], Ligero [AHIV17], Aurora [BCR+19] in ECDSA verification. For Groth16, Ligero, and Aurora, we use the implementations from [BCTV14] and the libiop library [Lab21].

Following [Set20, BFK+24], all the PCSs are instantiated over the scalar field of `Curve25519` [Ber06]. All experiments are run on a computer with an `AMD EPYC 9754` CPU and 512GB of RAM. Each instantiation is executed 6 times and the average performance is reported. For both PCS and NIZK benchmarks, we report sequential and parallel executions. The parallel executions use up to 32 threads,

specified via `RAYON_NUM_THREADS` = 32. Our Rust implementation is available at https://anonymous.4open.science/r/WHTPCS-1F82.

For Spielman's code based schemes, we evaluate two parameter sets. The first follows the original Brakedown paper [GLS+23], with inverse code rate 1.521, relative distance 0.04, $\alpha = 0.178$, $c_n = 7$, $d_n = 22$ and the number of opened columns 6593. It is denoted as `Brakedown1.5(B1.5)`. The second is derived from [Hab23, BFK+24], with inverse code rate 2, relative distance 0.1, $\alpha = 0.3$, $c_n = 11$, $d_n = 22$ and the number of opened columns 2617. It is denoted as `Brakedown2(B2)`. For EA code based schemes [BFK+24], we also consider two parameter sets. Both use inverse code rate 2, and $\gamma = 18$, which is an independent density parameter for each of the generator matrices. The number of non-zeros of per column is $\gamma \cdot \log_2(\texttt{code_length})/\texttt{inverse_rate}$. With provable relative distance $\delta = 0.1$, the number of opened columns is 2617, which is denoted as `EA`. Under a conjectured relative distance of $\delta = 0.25$, the number of opened columns is 1012, yielding a substantial performance improvement in proof size, and we denote this variant as `conjectured EA(CEA)`.

6.1 Encoding Time

The performance of the encoding time is shown in Figure 1 and the concrete data for message size from 2^{12} to 2^{28} is reported in Table 6. We benchmark five encoding schemes: `Brakedown1.5`, `Brakedown2`, EA codes, QA codes (ours), and RAA codes. For a message of length N, the encoder of Brakedown takes $O(N)$ time, while EA codes need $O(N \log N)$ time. For rate 1/2, the encoding of our QA code takes N field multiplications and $N \log N$ additions, while RAA encoder performs $4N$ additions and 2 random permutations of a length $2N$ vector.

Across all tested message sizes, our QA code consistently outperforms `Brakedown1.5`, `Brakedown2`, and EA codes. Starting from 2^{22}, the QA encoder is about 3× faster than `Brakedown1.5` encoder and the gap widens as N increases, reflecting the relatively slower growth rate of our QA encoder. Compared with RAA, QA encoding incurs only a modest overhead: its runtime is about 1.6× larger than RAA, whose encoder essentially performs no multiplication.

Table 6: Encoding Time Comparison

	12	14	16	18	20	22	24	26	28
B1.5	1.00 ms	3.17 ms	18.33 ms	89.17 ms	0.41 s	3.2 s	16.3 s	74.4 s	364.4 s
B2	1.17 ms	6.00 ms	30.67 ms	0.147 s	0.89 s	5.5 s	27.3 s	114.3 s	637.2 s
EA	29.33 ms	149.50 ms	0.82 ms	4.24 s	43.36 s	–	–	–	–
QA	0.17 ms	2.00 ms	11.00 ms	50.67 ms	0.25 s	1.1 s	4.8 s	21.9 s	101.8 s
RAA	– ms	1.00 ms	6.17 ms	28.83 ms	0.18 s	0.8 s	3.0 s	12.6 s	62.9 s

This shows that, for sufficiently large message length, our quasi-abelian code achieves a substantially smaller *concrete* encoding time than Spielman's linear-time encodable codes, EA codes, and their variants.

6.2 Polynomial Commitment Schemes

We evaluate the performance of polynomial commitment schemes (PCSs) based on Spielman's code, EA code, and our QA code. The reported metrics include `commit time`, `open time`, `verify time`, and `proof size`. We report both sequential and parallel results, which are shown in Figure 7, Figure 8 respectively. In both settings, our QA-PCS outperforms the Brakedown-PCS and the EA-PCS across all metrics. Concrete sequential and parallel performance numbers are provided in Section A.1.

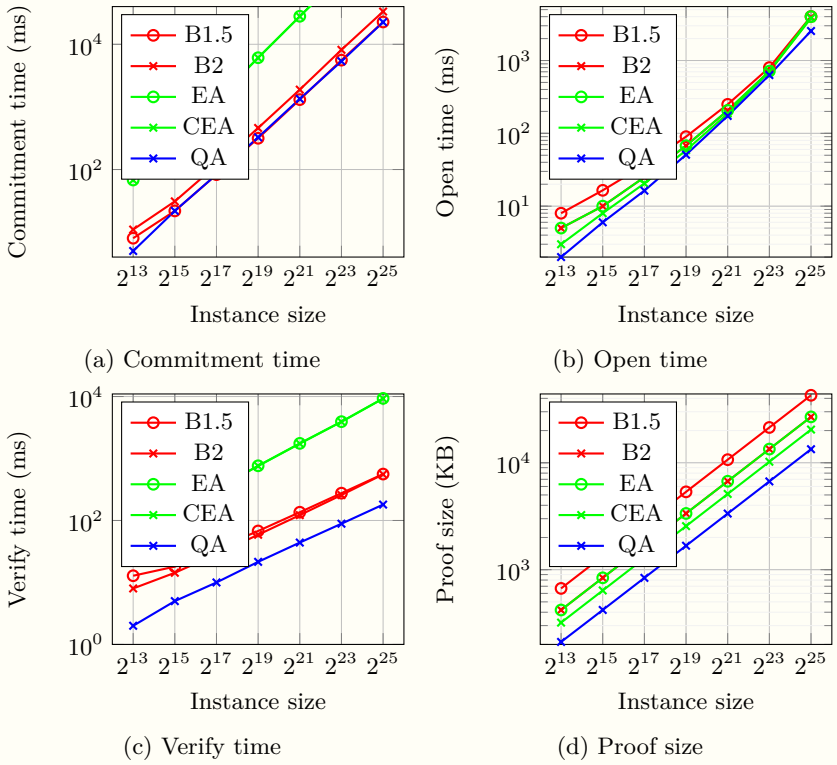
Details of our parallel experiments.

- For the commitment phase, we parallelize both the procedure of encoding of each row of the coefficient matrix and the procedure of computing Merkle hash of each column of the coefficient matrix. For fairness, we did not parallelize the encoding algorithms, although our encoding algorithm could benefit a lot from it (because both Walsh-Hadamard Transform and Hadamard product are native to parallelization).
- For opening phase, we did not use any parallelization. Hence, the numbers in Figure 7(b)and Figure 8(b)are almost the same.
- For verification, we parallelize the procedure of verifying t opened Merkle-tree hash values. In particular, we parallelize the encoding procedure of Spielman's code and EA code. We did NOT parallelize the encoding algorithm of our QA code. Although these mitigate the advantages of QA codes, we still offer the shortest verification time.
- Finally, the parallel execution is performed with a maximum of 32 threads specified by `RAYON_NUM_THREADS` = 32.

In terms of `commit time`, PCS based on QA code significantly outperforms those from `Brakedown2`, EA code, due to our fast encoding algorithm. Moreover, it is also slightly faster than PCS from `Brakedown1.5`. In fact, the encoding procedure of our PCS is faster than `Brakedown1.5`. While `Brakedown1.5` has a lower inverse rate, the hashing costs of `Brakedown1.5` are smaller. Consequently, `Brakedown1.5` and ours are comparable. Concretely for instance size 2^{21}, rate $1/2$, our QA-PCS needs 1344 ms in the sequential setting, which is $1.4\times$ smaller than 1887 ms of `Brakedown2`. As for parallel setting, QA-PCS needs 127 ms, which is $1.45\times$ smaller than 184 ms of `Brakedown2`.

The `open time` is actually dominated by the binding operation in the Rust implementation, which performs a vector matrix multiplication over the scalar field of `Curve25519`. Still, our PCS also achieves substantial improvements, due to the large minimum distance of QA codes, which allows the prover to open fewer Merkle-tree hashing values. Specifically, our scheme opens only 649 columns, while EA code with conjectured minimum distance needs to open 1012 columns. Concretely for instance size 2^{21}, rate $1/2$, our QA-PCS needs ≈ 173 ms in both our sequential and parallel settings, while `Conjectured EA` needs ≈ 186 ms and `Brakedown2` needs ≈ 205 ms.

Figure 7: Sequential performance comparisons of PCS



(a) Commitment time (b) Open time

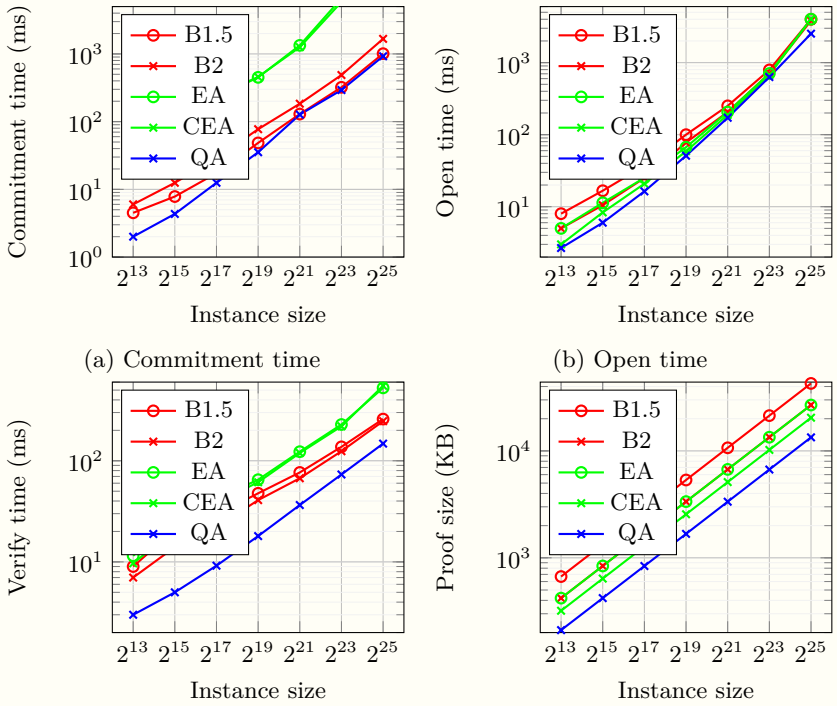
(c) Verify time (d) Proof size

B1.5, B2, EA, CEA, QA stand for Brakedown1.5, Brakedown2, EA, conjectured EA, QA, respectively.

The superiority of our PCS in terms of `verify time` is more obvious, which benefits a lot from both a large minimum distance and fast encoding algorithms. Moreover, PCS from QA code is still the fastest in verification, even if we minimize our advantages by not using a parallel encoding algorithm for QA code and parallelizing Merkle-tree hashing verification. Concretely for instance size 2^{21}, rate 1/2, our QA-PCS needs 44 ms in the sequential setting, which is 2.7× smaller than 122 ms of `Brakedown2`. As for parallel setting, QA-PCS needs 36.5 ms, which is 1.8× smaller than 67.2 ms of `Brakedown2`.

Finally, our PCS offers the smallest `proof size`, which is even $\approx$ 1.52× smaller than EA code with conjectured minimum distance of 0.25.

Figure 8: Parallel performance comparison of PCS



(a) Commitment time (b) Open time

(c) Verify time (d) Proof size

B1.5, B2, EA, CEA, QA stand for Brakedown1.5, Brakedown2, EA, conjectured EA, QA, respectively.

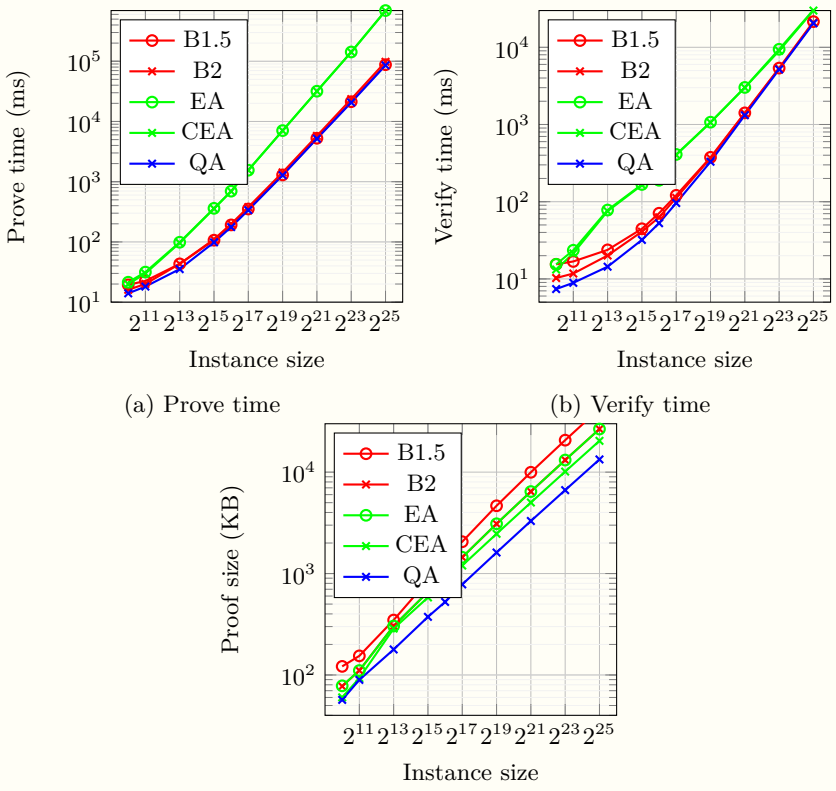
6.3 Application to SpartanNIZK [Set20]

We replace SpartanNIZK [Set20] with PCS from Spielman's code, EA code, and QA code, and report the sequential and parallel performance of the resulting NIZKs (with sublinear proof size, and linear verification) in Figure 9 and Figure 10. Concrete numbers for drawing these figures can be found in Section A.2. Our implementation follows the open source code of [Set20, GLS+23, BFK+24].

Details of our parallel experiments.

- On the prover side, we use the parallel version of the corresponding PCS. Other operations are not parallelized.
- On the verifier side, we use the parallel version of the corresponding PCS. Other operations are not parallelized as well.
- Again, the parallel execution is performed with a maximum of 32 threads specified by `RAYON_NUM_THREADS` = 32.

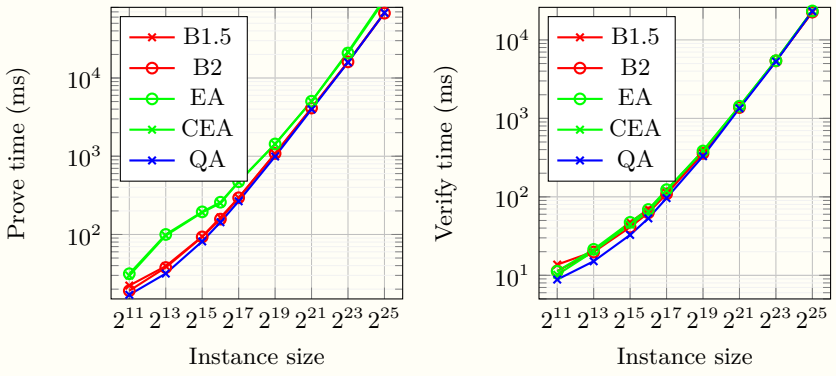
Figure 9: Sequential performance comparisons of NIZKs



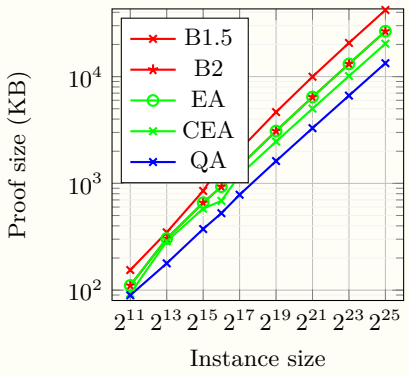
(a) Prove time (b) Verify time

(c) Proof size

Figure 10: Parallel performance comparison of NIZKs



(a) Prove time (b) Verify time



(c) Proof size

Essentially, the `prove time` is dominated by R1CS evaluations, where `commit time` & `open time` of the underlying PCS only contribute a small fraction of `prove time`. Consequently, for a large number of constraints, the `prove time` of NIZK from Spielman's code, EA code, and our QA code are very close. Similarly, the `verify time` from different codes becomes very close, because it is dominated by the R1CS evaluations as well. Still, since our PCS from QA code outperforms others, the NIZK from QA code is the fastest in terms of prover time and verification.

The `proof size` is mainly determined by the proof size of the underlying PCS. As shown in Section 6.2, our PCS has a much smaller proof size than those from Spielman's code and EA code, thanks to the high minimum distance of QA codes. Concretely, for proving R1CS relations with 2^{19} constraints, our NIZK from QA code has proof size 1620.58 KB, which is approximately 1.52× smaller than 2463.14 KB from EA code with conjectured distance, and 1.91× smaller than 3091.5 KB from `Brakedown2`.

We also evaluate the concrete efficiency of code-based SNARKs by replacing SpartanSNARK [Set20] with PCS from Spielman's code, EA code, and QA code. Concrete numbers are referred to Table A.3.

6.4 Application to ECDSA Verification

A typical application of SNARKs is ECDSA signature verification [XZC+22, SLLg], commonly performed over the scalar field of `secp256k1` or `Curve25519`. There are two approaches to verify ECDSA signatures via SNARKs. The first is to directly encode the ECDSA verification circuit over the native scalar field of the curve and run the SNARK over the field. The second is to transform the circuit to a different field required by the underlying SNARK, such as for Ligero [AHIV17], FRI [BBHR18], STARK [BBHR19], Aurora [BCR+19], or Virgo [ZXZS20, ZLW+21]. However, this transformation is extremely expensive, since simulating field operations over a different field introduces significant over-

head. For instance, ECDSA verification over the native scalar field of `secp256k1` or `Curve25519` takes roughly 2^{16} R1CS constraints, whereas the same circuit over the scalar field of the pairing-friendly curve `bn254` requires $\approx 2^{21}$ constraints, $\approx 25\times$ larger. This highlights the practical necessity for field-agnostic SNARKs.

Table 11: Concrete efficiency for ECDSA verification

Instance Size	Protocol	Prove time	Verify time	Proof size
2^{21}	Groth16	149 s	2 ms	128 B
	Ligero	111.45 s	61.89 s	19.32 MB
	Aurora	583.35 s	16.97 s	0.14 MB
2^{21}	SNARK-Brakedown1.5	51.13 s	1.41 s	50.40 MB
	SNARK-Brakedown2	55.39 s	1.24 s	36.96 MB
	SNARK-EA	320.62 s	19.45 s	38.24 MB
	SNARK-EA(Conjectured)	320.22 s	21.57 s	32.04 MB
	SNARK-QA	48.06 s	0.42 s	19.21 MB
2^{16}	NIZK-Brakedown1.5	0.19 s	70.89 ms	1542.40 KB
	NIZK-Brakedown2	0.20 s	62.85 ms	930.96 KB
	NIZK-EA	0.70 s	189.58 ms	931.36 KB
	NIZK-EA(Conjectured)	0.69 s	186.20 ms	686.65 KB
	NIZK-QA	0.18 s	52.69 ms	525.82 KB
2^{16}	SNARK-Brakedown1.5	1.60 s	244.38 ms	7664.42 KB
	SNARK-Brakedown2	1.65 s	186.21 ms	5860.82 KB
	SNARK-EA	7.89 s	2080.11 ms	6623.44 KB
	SNARK-EA(Conjectured)	7.78 s	1984.48 ms	5441.97 KB
	SNARK-QA	1.44 s	76.97 ms	3317.69 KB

Table 11 reports the performance of various succinct schemes for ECDSA verification in the sequential setting. For Groth16, Ligero, and Aurora, the data corresponds to R1CS instances with 2^{21} constraints. For NIZKs and SNARKs from Spielman's code, EA code, and our QA code, the data corresponds to R1CS with 2^{16} constraints over the native scalar field. For completeness, we also provide concrete numbers for code-based SNARKs with instance size 2^{21}, which shows that our SNARK from QA codes has good concrete efficiency.

As shown in the table, our SNARK is $1.11\times$ faster than `Brakedown1.5` in proving time, $2.42\times$ faster than `Brakedown2` in verification, and has proof size $1.64\times$ smaller than the SNARK from EA code with conjectured distance. Moreover, for ECDSA verification, our SNARK from QA codes is $405\times$, $200\times$ faster than Aurora [BCR+19] in proving time and verification, respectively.

Acknowledgment. The authors would like to thank anonymous reviewers at CRYPTO 2026 for suggestions on improving editorial quality of the distance analyses as well as pointing out bugs in the PCS/SNARK implementation. In particular, the authors also thank Peter Scholl for pointing out the third improvement in the distance analyses. The work was supported in part by the National Key Research and Development Program of China under Grants 2022YFA1004900

and 2023YFE0123900, and the National Natural Science Foundation of China under Grants 12361141818, 12426302 and 12271084.

References

ACFY24. Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. STIR: reed-solomon proximity testing with fewer queries. In *CRYPTO 2024*, volume 14929 of *LNCS*, pages 380–413. Springer, 2024. 2

ACFY25. Gal Arnon, Alessandro Chiesa, Giacomo Fenzi, and Eylon Yogev. WHIR: reed-solomon proximity testing with super-fast verification. In *EUROCRYPT 2025*, volume 15604 of *LNCS*, pages 214–243. Springer, 2025. 2

AHIV17. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *CCS 2017*, pages 2087–2104. ACM, 2017. 2, 3, 24, 30

BBC+24. Maxime Bombar, Dung Bui, Geoffroy Couteau, Alain Couvreur, Clément Ducros, and Sacha Servan-Schreiber. FOLEAGE: $\mathbb{F}_4$OLE-based multi-party computation for boolean circuits. In *ASIACRYPT 2024*, volume 15489 of *LNCS*, pages 69–101. Springer, 2024. 4

BBHR18. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. 2, 30

BBHR19. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable zero knowledge with no trusted setup. In *CRYPTO 2019*, volume 11694 of *Lecture Notes in Computer Science*, pages 701–732. Springer, 2019. 30

BCCD23. Maxime Bombar, Geoffroy Couteau, Alain Couvreur, and Clément Ducros. Correlated pseudorandomness from the hardness of quasi-abelian decoding. In *CRYPTO 2023*, volume 14084 of *LNCS*, pages 567–601. Springer, 2023. 4, 5

BCF+25. Martijn Brehm, Binyi Chen, Ben Fisch, Nicolas Resch, Ron D. Rothblum, and Hadas Zeilberger. Blaze: Fast snarks from interleaved RAA codes. In *EUROCRYPT 2025*, volume 15604 of *LNCS*, pages 123–152. Springer, 2025. 2, 3, 4, 6, 7, 24

BCG+17. Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In *ASIACRYPT 2017*, volume 10626 of *LNCS*, pages 336–365. Springer, 2017. 2

BCG+22. Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022*, volume 13508 of *LNCS*, pages 603–633. Springer, 2022. 3

BCL22. Jonathan Bootle, Alessandro Chiesa, and Siqi Liu. Zero-knowledge iops with linear-time prover and polylogarithmic-time verifier. In *EUROCRYPT 2022*, volume 13276 of *LNCS*, pages 275–304. Springer, 2022. 2, 3

BCR+19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT 2019*, volume 11476 of *LNCS*, pages 103–128. Springer, 2019. 2, 24, 30, 31

BCTV14. Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In *USENIX Security Symposium 2014*, pages 781–796. USENIX Association, 2014. 24

Ber06. Daniel J. Bernstein. Curve25519: New diffie-hellman speed records. In *PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006. 3, 24

BFK+24. Alexander R. Block, Zhiyong Fang, Jonathan Katz, Justin Thaler, Hendrik Waldner, and Yupeng Zhang. Field-agnostic snarks from expand-accumulate codes. In *CRYPTO 2024*, volume 14929 of *LNCS*, pages 276–307. Springer, 2024. 2, 3, 6, 7, 13, 23, 24, 25, 28

BLVW19. Zvika Brakerski, Vadim Lyubashevsky, Vinod Vaikuntanathan, and Daniel Wichs. Worst-case hardness for lpn and cryptographic hashing via code smoothing. In *EUROCRYPT 2019*, pages 619–635. Springer, 2019. 11

BM06. L. M. J. Bazzi and Sanjoy K. Mitter. Some randomized code constructions from group actions. *IEEE Trans. Inf. Theory*, 52(7):3210–3219, 2006. 11

DI14. Erez Druk and Yuval Ishai. Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In *ITCS 2014*, pages 169–182. ACM, 2014. 24

DJM98. Dariush Divsalar, Hui Jin, and Robert J McEliece. Coding theorems for" turbo-like" codes. In *Proceedings of the annual Allerton Conference on Communication control and Computing*, volume 36, pages 201–210. University Of Illinois, 1998. 3, 6

FL15. Yun Fan and Liren Lin. Thresholds of random quasi-abelian codes. *IEEE Trans. Inf. Theory*, 61(1):82–90, 2015. 4

FL22. Yun Fan and Liren Lin. Asymptotic properties of quasi-group codes. *CoRR*, abs/2203.00958, 2022. 4, 9, 10, 11, 12, 13, 18, 19

GLS+23. Alexander Golovnev, Jonathan Lee, Srinath T. V. Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and field-agnostic snarks for R1CS. In *CRYPTO 2023*, volume 14082 of *LNCS*, pages 193–226. Springer, 2023. 2, 3, 6, 7, 13, 22, 23, 24, 25, 28

Gro16. Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 305–326. Springer, 2016. 24

GZ06. Philippe Gaborit and Gilles Zémor. Asymptotic improvement of the gilbert-varshamov bound for binary linear codes. In *IEEE ISIT 2006*, pages 287–291. IEEE, 2006. 5

Hab23. Ulrich Haböck. Brakedown's expander code. *IACR Cryptol. ePrint Arch.*, page 769, 2023. 3, 24, 25

KPR+25. Vladimir Kolesnikov, Stanislav Peceny, Rahul Rachuri, Srinivasan Raghuraman, Peter Rindal, and Harshal Shah. Block-accumulate codes: Accelerated linear codes for pcgs and ZK. *IACR Cryptol. ePrint Arch.*, page 1828, 2025. 3

Lab21. SCIPR Lab. libiop: a c++ library for IOP-based zksnarks. GitHub Repository, 2021. Last commit to README.md on Apr 3, 2021. 24

LLX+26. Zhe Li, Hongqing Liu, Chaoping Xing, Yizhou Yao, and Chen Yuan. Faster pseudorandom correlation generators via walsh-hadamard transform. *IACR Cryptol. ePrint Arch.*, page 196, 2026. 4, 5, 6, 13, 18, 19

LXYY25a. Zhe Li, Chaoping Xing, Yizhou Yao, and Chen Yuan. Efficient pseudorandom correlation generators for any finite field. In Serge Fehr and

Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025*, volume 15605 of *LNCS*, pages 145–175. Springer, 2025. 4

LXYY25b. Zhe Li, Chaoping Xing, Yizhou Yao, and Chen Yuan. Efficient pseudorandom correlation generators over $\mathbb{Z}/p^k\mathbb{Z}$. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology - CRYPTO 2025*, volume 16003 of *LNCS*, pages 207–240. Springer, 2025. 4

RR22. Noga Ron-Zewi and Ron D. Rothblum. Proving as fast as computing: succinct arguments with constant prover overhead. In *STOC 2022*, pages 1353–1363. ACM, 2022. 2

RR24. Noga Ron-Zewi and Ron Rothblum. Local proofs approaching the witness length. *J. ACM*, 71(3):18, 2024. 6

RRT23. Srinivasan Raghuraman, Peter Rindal, and Titouan Tanguy. Expand-convolute codes for pseudorandom correlation generators from LPN. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023*, volume 14084 of *LNCS*, pages 602–632. Springer, 2023. 3

Set20. Srinath T. V. Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In *CRYPTO 2020*, volume 12172 of *LNCS*, pages 704–737. Springer, 2020. 2, 24, 28, 30, 37

SLLg. Yi Sun, Tony Liu, Wen-Ding Li, and gubsheep. zk-ecdsa: zksnarks for ecdsa (part 1). Web Article. 30

Spi96. Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Trans. Inf. Theory*, 42(6):1723–1731, 1996. 3, 7, 24

Var57. Rom Rubenovich Varshamov. Estimate of the number of signals in error correcting codes. *Docklady Akad. Nauk, SSSR*, 117:739–741, 1957. 4

Was77. Siri Krishan Wasan. Quasi abelian codes. *Publ. Inst. Math*, 21(35):201–206, 1977. 4

XZC+22. Tiancheng Xie, Jiaheng Zhang, Zerui Cheng, Fan Zhang, Yupeng Zhang, Yongzheng Jia, Dan Boneh, and Dawn Song. zkbridge: Trustless cross-chain bridges made practical. In *CCS 2022*, pages 3003–3017. ACM, 2022. 30

XZS22. Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In *CRYPTO 2022*, volume 13510 of *LNCS*, pages 299–328. Springer, 2022. 2, 3, 6

ZCF24. Hadas Zeilberger, Binyi Chen, and Ben Fisch. Basefold: Efficient field-agnostic polynomial commitment schemes from foldable codes. In *CRYPTO 2024*, volume 14929 of *LNCS*, pages 138–169. Springer, 2024. 2, 3

ZLW+21. Jiaheng Zhang, Tianyi Liu, Weijie Wang, Yinuo Zhang, Dawn Song, Xiang Xie, and Yupeng Zhang. Doubly efficient interactive proofs for general arithmetic circuits with linear prover time. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21*, pages 159–177. ACM, 2021. 30

ZXZS20. Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. Transparent polynomial delegation and its applications to zero knowledge proof. In *SP 2020*, pages 859–876. IEEE, 2020. 30

Supplementary Material

A Detailed Numbers of the Experiments

The `Brakedown1.5` and `Brakedown2` parameters are listed as follows:

Table 12: Brakedown Parameters

	INVERSE RATE	OPEN COLUMNS	ALPHA	CN	DN
Brakedown1.5	1.521	6593	0.178	7	22
Brakedown2	2.0	2617	0.30	11	22

The EA code and conjectured EA code parameters are given as follows:

Table 13: EA Codes Parameters

	INVERSE RATE	OPEN COLUMNS	PROB MULTIPLIER
EA Code	2	2617	18
Conjectured EA	2	1012	18

For QA code, we use inverse rate 2, and the number of opened columns is 649. All these codes are implemented over $\mathbb{F}_{2^{255}-19}$.

A.1 Polynomial Commitment Scheme

Table 14: PCS Comparisons

	2^{13}	2^{15}	2^{17}	2^{19}	2^{21}	2^{23}	2^{25}
1 thd							
Commit(ms)							
B1.5	8.00	21.83	82.83	315.67	1307.50	5585.17	22708.50
B2	11.00	31.00	117.17	460.83	1887.00	8164.33	33332.50
EA	68.00	279.83	1292.50	6101.00	28012.83	126905.83	605926.33
CEA	68.00	285.33	1299.50	6076.17	27852.50	126414.83	603512.50
QA	5.00	21.83	83.17	327.00	1344.33	5354.67	22565.00
Open(ms)							
B1.5	8.00	16.50	36.50	90.00	249.83	799.50	4024.83
B2	5.00	10.00	25.00	67.50	205.33	706.17	3929.17
EA	5.00	10.00	25.00	67.67	206.33	713.50	4010.83
CEA	3.00	8.00	20.50	58.50	186.67	703.83	3812.50
QA	2.00	6.00	16.33	50.83	173.33	631.17	2541.50
Verify(ms)							
B1.5	12.83	18.00	34.83	67.67	135.83	275.83	560.67
B2	8.00	14.33	28.50	58.83	122.00	257.50	554.33
EA	65.00	139.50	322.33	763.00	1754.00	3931.50	9328.50
CEA	63.33	138.00	320.67	757.33	1736.33	3908.83	9243.17
QA	2.00	5.00	10.00	21.50	44.00	88.67	180.50
32 thds							
Commit							
B1.5	4.50	7.83	17.33	48.50	129.33	321.17	1008.00
B2	6.00	12.50	28.83	77.33	184.00	488.00	1669.00
EA	71.33	118.67	230.83	450.67	1337.50	6609.67	36088.67
CEA	69.67	119.00	240.67	457.50	1257.67	5843.50	36530.00
QA	2.00	4.33	12.50	35.33	127.00	293.00	923.00
Open							
B1.5	8.00	16.67	36.83	100.00	251.33	792.17	3996.33
B2	5.00	10.50	25.00	77.83	206.50	704.00	3821.67
EA	5.00	11.33	25.17	67.00	206.00	706.67	4011.50
CEA	3.00	8.33	20.83	58.83	188.33	674.67	4032.17
QA	2.67	6.00	16.33	51.00	171.67	630.83	2524.67
Verify							
B1.5	9.00	20.83	29.33	47.67	76.67	137.33	258.50
B2	7.00	14.00	22.00	40.83	67.17	124.33	246.33
EA	11.50	25.33	42.00	65.00	123.00	228.33	525.83
CEA	9.67	20.83	38.67	61.33	119.33	219.17	551.33
QA	3.00	5.00	9.17	18.00	36.50	73.17	147.83
$\|\mathsf{proof}\|$(KB)							
B1.5	668.22	1336.31	2672.47	5344.75	10689.28	21378.31	42756.34
B2	419.72	839.31	1678.47	3356.75	6713.28	13426.31	26852.34
EA	419.72	839.31	1678.47	3356.75	6713.28	13426.31	26852.34
CEA	319.92	639.15	1277.82	2554.97	5108.88	10216.91	20432.72
QA	211.06	420.12	838.49	1675.13	3347.93	6694.02	13386.08

A.2 Application to SpartanNIZK [Set20]

Table 15: NIZK Comparisons

	2^{11}	2^{13}	2^{15}	2^{16}	2^{17}	2^{19}	2^{21}	2^{23}	2^{25}
1thd									
Prove(ms)									
B1.5	22.52	43.21	106.87	191.63	352.28	1296.13	5268.46	21295.32	87698.86
B2	20.07	43.14	110.62	199.20	378.66	1427.06	5758.64	23494.10	97880.21
EA	31.54	98.78	361.26	696.36	1552.49	7056.40	31797.72	142463.58	694197.79
CEA	29.91	96.46	355.69	690.65	1554.40	7032.31	31777.31	141550.01	685252.72
QA	18.14	35.40	98.28	176.53	336.94	1278.86	5178.91	20644.38	85459.52
Verify(ms)									
B1.5	16.83	23.77	44.60	70.89	120.94	375.51	1414.11	5378.21	21502.10
B2	11.74	20.09	40.97	62.85	112.01	366.17	1391.63	5312.23	20925.96
EA	23.50	77.98	167.29	189.58	409.36	1071.16	3019.93	9421.45	30510.56
CEA	21.44	75.41	164.38	186.20	403.80	1060.66	3004.35	8999.82	29904.04
QA	8.90	14.39	31.99	52.69	96.15	329.63	1317.24	5162.43	20471.32
32thds									
Prove(ms)									
B1.5	22.22	39.37	94.06	165.05	294.44	1070.94	4095.57	16044.55	68516.56
B2	19.10	37.82	93.09	157.05	294.44	1084.19	4129.52	16072.67	67421.92
EA	31.48	100.01	194.58	258.39	472.39	1435.18	5047.28	20847.88	98247.17
CEA	29.86	96.87	191.09	250.88	468.44	1427.22	5032.60	21590.28	103470.22
QA	16.94	31.66	81.49	143.32	265.79	991.84	3975.01	15840.51	67992.75
Verify(ms)									
B1.5	13.69	20.04	47.40	69.42	116.94	359.32	1384.38	5449.15	22663.64
B2	11.36	19.80	41.10	62.63	108.88	357.29	1375.21	5403.43	22602.49
EA	11.26	21.42	47.29	69.02	123.42	384.16	1427.90	5429.53	23382.20
CEA	10.15	20.28	44.32	64.31	120.06	387.58	1423.66	5526.88	23498.54
QA	8.83	15.10	32.72	53.18	96.35	329.60	1341.77	5298.90	23153.23
$\|\mathsf{proof}\|$(KB)									
B1.5	154.33	347.81	850.82	1542.40	2059.46	4652.31	9974.25	20669.34	42055.48
B2	110.63	303.80	659.36	930.96	1445.31	3095.35	6431.22	13167.14	26589.75
EA	110.66	304.18	659.26	931.36	1447.16	3092.00	6440.10	13141.21	26601.70
CEA	92.90	286.46	580.22	686.65	1197.17	2465.13	5012.22	10123.36	20337.25
QA	89.50	178.26	373.64	525.82	783.09	1619.02	3292.23	6636.69	13356.33

A.3 Application to SpartanSNARK [Set20]

Details of our parallel experiments.

- On the prover side, we use the parallel version of the corresponding PCS. Other operations are not parallelized.
- On the verifier side, we use the parallel version of the corresponding PCS. Other operations are not parallelized as well.
- The parallel execution is performed with maximum 32 threads specified by `RAYON_NUM_THREADS` = 32.

Table 16: SNARK Comparisons

		2^{11}	2^{13}	2^{15}	2^{17}	2^{19}	2^{21}	2^{23}
1 thd								
	Prove(s)							
	B1.5	0.104	0.269	0.855	3.053	11.714	49.931	207.491
	B2	0.088	0.255	0.862	3.264	12.877	54.785	229.622
	EA	0.214	0.815	3.512	15.975	70.563	320.620	1511.041
	CEA	0.207	0.802	3.579	15.602	70.373	320.222	1455.345
	QA	0.070	0.212	0.756	2.869	11.417	48.065	208.050
	Verify(ms)							
	B1.5	59.42	97.37	177.75	339.35	670.75	1350.41	2760.16
	B2	43.07	76.08	143.84	284.78	584.61	1180.55	2507.22
	EA	294.64	680.98	1568.65	3748.40	8190.52	19450.98	48231.77
	CEA	287.28	670.94	1772.34	3562.74	8115.17	21573.87	49267.75
	QA	20.09	33.23	59.07	110.23	211.31	418.47	854.48
32 thds								
	Prove(s)							
	B1.5	0.098	0.234	0.717	2.483	9.175	38.892	168.189
	B2	0.078	0.207	0.672	2.368	8.954	38.021	169.919
	EA	0.206	0.438	0.947	2.929	11.408	55.790	265.375
	CEA	0.195	0.390	0.942	2.936	11.064	53.163	263.861
	QA	0.063	0.172	0.594	2.233	8.611	36.116	159.599
	Verify(ms)							
	B1.5	58.35	88.05	136.14	209.90	361.99	659.22	1296.45
	B2	41.99	64.90	103.39	169.31	302.21	607.82	1169.92
	EA	69.19	109.92	209.69	386.15	741.99	1249.02	2776.21
	CEA	57.01	93.48	186.61	373.22	654.17	1282.73	2915.50
	QA	20.67	32.02	52.17	92.07	170.70	331.74	676.73
	$\|\mathsf{proof}\|$(MB)							
	B1.5	1.16	2.45	5.24	11.50	24.38	50.39	102.43
	B2	0.98	2.06	4.23	8.83	18.18	36.95	74.52
	EA	1.15	2.49	4.79	9.57	19.16	38.28	76.62
	CEA	0.99	2.08	4.05	8.03	16.02	32.04	64.09
	QA	0.64	1.21	2.41	4.81	9.60	19.20	38.37

B Missing Proofs of Section 3.1

Proof (Proof of Lemma 3.7). Let $\mathcal{C} \subseteq \mathbb{F}_p[\mathbb{G}]$ be a group code. Let $I \subseteq \mathbb{G}$ be any information set of $\mathcal{C}$. Since $\mathcal{C}$ is a group code, $x \cdot \mathcal{C} = \mathcal{C}$ for any $x \in \mathbb{G}$. Thus, xI is also an information set of $\mathcal{C}$. Thus, $\{xI, x \in \mathbb{G}\}$ forms a balanced system of information index sets of $\mathbb{G}$. □

Proof (Proof of Lemma 3.8). We present the proof for completeness. We first prove the following claim

Claim. If $\mathcal{B} \subseteq \mathbb{F}_p^N$ and $\omega = \sum_{b \in \mathcal{B}} \frac{\mathsf{wt}(b)}{N|\mathcal{B}|}$, then

$$|\mathcal{B}| \leq p^{N h_p(\omega)}.$$

Proof. Let $M = |\mathcal{B}|$. Let $(X_1, \ldots, X_N)$ be the joint random variable such that

$$f(a_1, \ldots, a_N) := \Pr[(X_1, \ldots, X_N) = (a_1, \ldots, a_N)] = \frac{1}{M} \quad \forall (a_1, \ldots, a_N) \in \mathcal{B},$$

and 0 otherwise. It is clear that $H_p(X_1, \ldots, X_N) = \log_p M$ where

$$H_p(X_1, \ldots, X_N) = \sum_{(a_1, \ldots, a_N) \in \mathcal{B}} -f(a_1, \ldots, a_N) \log_p f(a_1, \ldots, a_N)$$

is the entropy function. Moreover, according to the property of entropy function, we have

$$H_p(X_1, \ldots, X_N) \leq \sum_{i=1}^{N} H_p(X_i).$$

It suffices to bound $H_p(X_i)$. Let ω_i be the average weight of i-th index in $\mathcal{B}$, i.e., $\omega_i = \sum_{(\ldots, b_i, \ldots) \in B} \frac{\mathsf{wt}(b_i)}{|B|}$. It is clear that

$$\begin{aligned}
H_p(X_i) &= -\Pr[X_i = 0] \log_p \Pr[X_i = 0] - \sum_{a \in \mathbb{F}_p^*} \Pr[X_i = a] \log_p \Pr[X_i = a] \\
&\leq -\Pr[X_i = 0] \log_p \Pr[X_i = 0] - \Pr[X_i \neq 0] \log_p \frac{\Pr[X_i \neq 0]}{p-1} \\
&= -(1-\omega_i) \log_p (1-\omega_i) - \omega_i \log_p \frac{\omega_i}{p-1}.
\end{aligned}$$

The last inequality is due to the convexity of $x \log x$ and the Jensen inequality. Observe that $\sum_{i=1}^{N} \omega_i = \omega N$ and using Jensen inequality again to obtain

$$\sum_{i=1}^{N} -(1-\omega_i) \log_p (1-\omega_i) \leq -N(1-\omega) \log(1-\omega)$$

and

$$\begin{aligned}
\sum_{i=1}^{N} -\omega_i \log_p \frac{\omega_i}{p-1} &= \sum_{i=1}^{N} \omega_i \log_p (p-1) - \sum_{i=1}^{N} \omega_i \log_p \omega_i \\
\leq N \log_p (p-1) + \sum_{i=1}^{N} -\omega_i \log_p \omega_i &\leq N \log_p (p-1) - N\omega \log_p \omega.
\end{aligned}$$

Combining everything together, we prove the claim. □

Since $\mathcal{C}$ is a balanced code, we can find the information set $I_1, \dots, I_s$ satisfying Definition 3.6. Let ρ_i be the projection in Definition 3.6. Then, we define the following map

$$\rho : \mathcal{C}^s \to \mathbb{F}_p^{I_1} \times \cdots \times \mathbb{F}_p^{I_s}, \quad \rho(c_1, \dots, c_s) = (\rho_1(c_1), \dots, \rho_s(c_s)).$$

Since ρ_i is a bi-jection, we have $|\rho(\mathcal{B})| = M^s$ for $|B| = M$. Fix $b = (b_1, \dots, b_n) \in \mathcal{B}$. We note that each index i is contained in exactly t information sets $I_{j_1}, \dots, I_{j_t}$. For each j_r, $(\dots, b, \dots) \in \mathcal{B}^s$ with b in the j_r-th index will contribute to the total weight of $\rho(\mathcal{B}^s)$. There are M^{s-1} such $(\dots, b, \dots)$ and thus contribute M^{s-1} weight if b_i is nonzero. We enumerate every b_i and find that the contribution of b in $\rho(\mathcal{B}^s)$ is $M^{s-1}\mathsf{wt}(b)t$. Thus, the total weight of $\rho(\mathcal{B}^s)$ is $M^s\omega Nt = M^s\omega ks$ as $Ns = kt$. Thus, the average weight of $\rho(\mathcal{B}^s)$ is ω. Since our claim says that $M^s \le p^{-ksh_p(\omega)}$, the desired result follows. □

Proof (Proof of Corollary 3.10). Let $I_1, \dots, I_s$ be the information sets of $\mathcal{C}$ defined in Definition 3.6. Then, it is easy to see that $I_1^h, \dots, I_s^h$ forms the balanced system of $\mathcal{C}^h$. □