Math Companion to Soundcalc

December 5, 2025

1 Notation and Preliminaries

1.1 Fields

Fields of size q are denoted as $\mathbb{F}_q$ or simply $\mathbb{F}$.

1.2 Reed-Solomon codes

We use the following notation:

- $RS[\mathbb{F}, S, \rho]$: Reed-Solomon code over the field $\mathbb{F}$ with evaluation domain S and rate ρ.
- $\deg(f)$: degree of the polynomial f.

2 FRI-based VM security level calculation

This section calculates the security level for a FRI-based VM in section 2.3.

2.1 FRI parameters

Global parameters used in the FRI analysis:

- m_J — Johnson parameter.
- r_{FRI} — number of FRI rounds.
- Folding factors $\widehat{\text{folds}} = [k_0, k_1, \ldots, k_{r_{FRI}-1}]$;
- t — number of queries.
- θ.
- δ.
- ρ — rate of the Reed-Solomon code.
- l_t — trace length.
- L — list size.
- $b_{\text{grind},Q}$ — grinding parameter for the query phase.
- n — witness size.
- b_{hash} — number of bits in the hash function output.
- b_{proof} — proof size in bits.
- B — batch size.

Notation specific to the Johnson bound:

-

2.2 Fixed constants

We fix the following constants for the soundness calculator:

- $m_J = 16$. Set in

  `fri.py/get_johnson_parameter_m()`

2.3 Security level for a FRI-based VM

The security level is calculated in

`zkvms/fri_based_vm.py/get_security_levels()`

.

It is done separately for two different regimes: UDR and JBR — using the same procedure:

1. Calculate the FRI round-by-round soundness errors $\mathbf{e}_{\mathrm{FRI},U}, \mathbf{e}_{\mathrm{FRI},J}$ using the formula from section 2.3.1 and section 2.3.2.
2. Obtain optimal δ_U, δ_J parameters using the formula from section 2.3.3 and section 2.3.4.
3. Obtain the list sizes L_U, L_J for the respective δ_U, δ_J.
4. Obtain the DEEP-ALI soundness errors $\epsilon_{\mathrm{D-A},U}, \epsilon_{\mathrm{D-A},J}$ using the formulas from Section **??**.
5. Compute the total soundness errors as

$$\epsilon_U = \min(\mathbf{e}_{\mathrm{FRI},U}, \epsilon_{\mathrm{D-A},U}), \quad \epsilon_J = \min(\mathbf{e}_{\mathrm{FRI},J}, \epsilon_{\mathrm{D-A},J}).$$

Then the full security level in bits is the maximum of the two regimes:

$$\text{Security level} = \max(-\log_2 \epsilon_U, -\log_2 \epsilon_J).$$

2.3.1 RBR soundness in UDR

`fri_based_vm.py/get_security_levels_for_regime()`

2.3.2 RBR soundness in JBR

`fri_based_vm.py/get_security_levels_for_regime()`

2.3.3 Optimal distance in UDR

`unique_decoding.py/get_max_delta()`

2.3.4 Optimal distance in JBR

`johnson_bound.py/get_max_delta()`

2.3.5 List sizes

`unique_decoding.py/get_max_list_size()`

`johnson_bound.py/get_max_list_size()`

2.3.6 DEEP-ALI errors

`fri_based_vm.py/get_DEEP_ALI_errors()`

2.4 Soundness formula

This is calculated in

```
fri.py/get_FRI_query_phase_error()
```

.

Query phase error:

$$\epsilon_{\text{query}} = (1-\theta)^t \cdot 2^{-b_{\text{grind},Q}} \tag{1}$$

The query phase error without grinding is computed as per [?][1]

2.5 Proof size

This calculation is performed in

```
fri.py/get_FRI_proof_size_bits()
```

. The FRI proof contains two parts: Merkle roots, and one "openings" per query, where an "opening" is a Merkle path for each folding layer. For each layer we count the size that this layer contributes, which includes the root and all Merkle paths.

Initial round: one root and one path per query. We assume that for the initial functions, there is only one Merkle root, and each leaf i for that root contains symbols i for all initial functions.

Folding rounds: we assume that "siblings" for the following layers are grouped together in one leaf. This is natural as they always need to be opened together.

The proof size is calculated as follows:

$$\begin{aligned} b_{\text{proof}} = & \underbrace{b_{\text{hash}} + t \cdot MP(\frac{n}{\widehat{\text{folds}}[0]}, B, \log_2|\mathbb{F}|, b_{\text{hash}}) +}_{\text{Initial round}} \\ & + \underbrace{\sum_{1 \le i \le r_{FRI}-2} \left(b_{\text{hash}} + t \cdot MP(\frac{n}{\prod_{1\le j \le i} \widehat{\text{folds}}[j]}, B, \log_2|\mathbb{F}|, b_{\text{hash}}) \right) +}_{\text{Folding rounds but last}} \\ & + \underbrace{\left(b_{\text{hash}} + t \cdot MP(\frac{n}{\widehat{\text{folds}}[r_{FRI}-1] \prod_{1 \le j \le r_{FRI}-1} \widehat{\text{folds}}[j]}, B, \log_2|\mathbb{F}|, b_{\text{hash}}) \right)}_{\text{Last folding round}} \end{aligned} \tag{2}$$

where $MP(n, s, q, b)$ is the Merkle path size calculated as

$$MP(n,s,q,b) = \underbrace{sq}_{\text{leaf size}} + \underbrace{sq}_{\text{sibling}} + \underbrace{\lceil \log_2 n \rceil \cdot b}_{\text{co-path}} \tag{3}$$

[1]Code refers to (7) and Th2 of [Hab22]

3 WHIR-based VM security level calculation

3.1 Notation and parameters

- Number of iterations: $M =$ `num_iterations`
- Iteration index: $i \in \{0, \dots, M-1\}$
- Folding round index: $s \in \{0, \dots, k\}$
- Field size: $|\mathbb{F}| = F$
- Constraint degree: $d =$ `constraint_degree`
- Folding factor: $k =$ `folding_factor`
- Log-degree in iteration i: $m_i =$ `log_degrees`$[i]$, and we set $m_i = m_0 - k \cdot i$
- Log-inverted rate in iteration i: $\mu_i =$ `log_inv_rates`$[i]$, and we set $\mu_i = \log_2 \frac{1}{\rho} + (k-1) \cdot i$
- Number of OOD samples in iteration i: $w_i =$ `num_ood_samples`$[i]$
- Number of queries in iteration i: $t_i =$ `num_queries`$[i]$
- Grinding bits:

$$g_{\text{batch}}, \quad g_{i,s}^{\text{fold}}, \quad g_i^{\text{ood}}, \quad g_i^{\text{qry}}.$$

3.2 Batching Error ϵ_{batch}

`get_batching_error()`

In the Johnson Bound Regime (JBR) we proceed as follows:

- Find η:

$$\eta = \frac{\sqrt{\rho}}{32}$$

- Compute linear error:

$$\epsilon_{\text{batch,lin},J} = \frac{2^{m_0}}{q(2\min(\eta, \sqrt{\rho}/20))^5}$$

- Compute powers error:

$$\epsilon_{\text{batch,pow},J} = \epsilon_{\text{batch,lin},J} \cdot (B-1)$$

Base error:

$$\varepsilon_{\text{batch},JBR}^{\text{base}} = \begin{cases} \epsilon_{\text{batch,pow},J} & \text{(power batching)}, \\ \epsilon_{\text{batch,lin},J} & \text{(linear batching)}. \end{cases}$$

After grinding:

$$\epsilon_{\text{batch}} = \varepsilon_{\text{batch}}^{\text{base}} \cdot 2^{-g_{\text{batch}}}.$$

3.3 Folding Error in JBR

`epsilon_fold()`

For iteration $i \in [M]$ and folding round $s \in \{1, 2, \dots, k\}$:

- Get list size

$$\ell_{i,s-1} = \texttt{regime.get\_max\_list\_size}(i, s-1, JBR).$$

- Get rate and dimension:

$$(\rho, m) = \texttt{get\_code\_for\_iteration\_and\_round(i,s)}$$

- Base error (two terms):

$$\varepsilon_{i,s}^{\text{fold,base}} = d \cdot \frac{\ell_{i,s-1}}{q} + \epsilon_{\text{batch,pow},J}(\rho, m, q, 2).$$

- After grinding:

$$\epsilon_{i,s}^{\text{fold}} = \varepsilon_{i,s}^{\text{fold,base}} \cdot 2^{-g_{i,s}^{\text{fold}}}.$$

3.4 Bits of Security in UDR and JBR

Computed in `get_security_levels_for_regime()`.

For any error term:

$$\lambda(\epsilon) = \lfloor -\log_2 \epsilon \rfloor.$$

Overall security:

$$\lambda_{\text{total}} = \min\{\lambda_{\text{batch}}, \lambda^{\text{fold}}_{i,s}, \lambda^{\text{out}}_{i}, \lambda^{\text{shift}}_{i}, \lambda^{\text{fin}}\}.$$

4 Proof Size Calculations

This section summarizes the proof-size formula computed in `get_proof_size_bits()`.

4.1 Initial Function Size

$$S_{\text{if}} = b_{\text{hash}}.$$

4.2 Initial Sumcheck Size

Thus the folding proof size per round is

$$S_{\text{sumcheck}} = kd \log_2 |\mathbb{F}|.$$

4.3 OOD Proof Size for Iteration i

$$S_{\text{ood},i} = b_{\text{hash}} + w_i \cdot \log_2 |\mathbb{F}| + kd \log_2 |\mathbb{F}|$$

$$S_{\text{ood},M} = +2^{m_M} \log_2 |\mathbb{F}|$$

4.4 Query Proof Size for Iteration i

Thus

$$S_{\text{qry},i} = t_i \cdot MP(2^{m_i+\mu_i-k}, 2^k, \log_2 q, b_{\text{hash}}).$$

4.5 Total Proof Size

Collecting all terms and summing over all M iterations:

$$\boxed{S_{\text{total}} = S_{\text{if}} + S_{\text{sumcheck}} + \sum_{i\in[M]} (S_{\text{ood},i} + S_{\text{qry},i})}$$

Expanding,

$$S_{\text{total}} = b_{\text{hash}} + kd \log_2 |\mathbb{F}| + 2^{m_M} \log_2 |\mathbb{F}| + \sum_{i\in[M-1]} (b_{\text{hash}} + w_i \cdot \log_2 |\mathbb{F}| + kd \log_2 |\mathbb{F}|) + \sum_{i\in[M]} \left(t_i \cdot MP(2^{m_i+\mu_i-k}, 2^k, \log_2 q, b_{\text{hash}})\right)$$