On Interactive Oracle Proofs for Boolean R1CS Statements

Ignacio Cascudo[1⋆] and Emanuele Giunta[1,2,3]

[1] IMDEA Software Institute, Madrid, Spain.
{ignacio.cascudo, emanuele.giunta}@imdea.org
[2] Universidad Politécnica de Madrid
[3] Scuola Superiore di Catania, Italy.

Abstract. The framework of interactive oracle proofs (IOP) has been used with great success to construct a number of efficient transparent zk-SNARKs in recent years. However, these constructions are based on Reed-Solomon codes and can only be applied *directly* to statements given in the form of arithmetic circuits or R1CS over *large* enough fields $\mathbb{F}$.
This motivates the question: what is the best way to apply these IOPs to statements that are naturally written as R1CS over *small* fields, and more concretely, the *binary* field $\mathbb{F}_2$? While one can just see the system as one over an extension field $\mathbb{F}_{2^e}$ containing $\mathbb{F}_2$, this seems wasteful, as it uses e bits to encode just one "information" bit. In fact, in FC21 the work BooLigero devised a way to apply the well-known Ligero while being able to encode $\sqrt{e}$ bits into one element of $\mathbb{F}_{2^e}$.
In this paper, we introduce a new protocol for $\mathbb{F}_2$-R1CS which among other things relies on a more efficient embedding which (for practical parameters) allows to encode $\geq e/4$ bits into an element of $\mathbb{F}_{2^e}$. Our protocol makes then *black box use* of *lincheck* and *rowcheck* protocols for the larger field. Using the lincheck and rowcheck introduced in Aurora and Ligero respectively we obtain $1.31-1.65\times$ smaller proofs for Aurora and $3.71\times$ for Ligero. We also estimate the reduction of prover time by a factor of $24.7\times$ for Aurora and between $6.9-32.5\times$ for Ligero without interactive repetitions.
Our methodology uses the notion of reverse multiplication friendly embeddings introduced in the area of secure multiparty computation, combined with a new IOPP to test linear statements modulo a subspace $V \leq \mathbb{F}_{2^e}$ which may be of independent interest.

1 Introduction

A zero-knowledge proof is a protocol in which a *prover* convinces a *verifier* that a statement is true, while conveying no other information apart from its truth. Zero-knowledge proofs have been among the most useful and studied primitives in cryptography since their advent in the 80s. Their popularity has increased even more in recent times, propelled by new applications motivated by blockchain

⋆ Research partially funded by the Spanish Government under the project SecuRing (ref. PID2019-110873RJ-I00) and by a research grant from Nomadic Labs and the Tezos Foundation.

technologies. This context has highlighted the relevance of a particular flavour of zero-knowledge proof, known as zero-knowledge succinct non-interactive argument of knowledge, or zk-SNARK.

The flexibility and efficiency of zk-SNARKs allow to provide practical arguments of knowledge for relations that lack any kind of algebraic structure, for instance the preimage relation for a one-way function. However, it is well known [Wee05] that under standard complexity assumptions, succinct non-interactive arguments do not exist unless some kind of setup is assumed, such as a common reference string. This either requires a trusted third party or the execution of heavy MPC protocols if the setup relies on secret randomness.

For this reason, *transparent* SNARKs have been proposed, whose setup involves only publicly generated randomness. Many constructions of transparent setup SNARKs have been proposed in recent years, both based on asymmetric [BCC+16], [WTS+18], [BBB+18], [BFS20] and symmetric [AHIV17], [BBHR18b], [BCR+19], [COS20], [Set20], [BFH+20] cryptographic techniques.

In this work we focus on this latter type of constructions and remark that all cited works in this category are built in (variants of) the Interactive Oracle Proof framework presented in [BCS16] and independently in [RRR16] as "interactive PCP". Moreover they all address directly or indirectly the NP-complete rank 1 constraint system satisfiability problem. An easier to state variant asks to prove, given $A, B, C \in \mathbb{F}^{m,n}$ and $\mathbf{b} \in \mathbb{F}^m$, the existence of a vector $\mathbf{z} \in \mathbb{F}^n$ such that $A\mathbf{z} * B\mathbf{z} = C\mathbf{z} + \mathbf{b}$, where $*$ is the component-wise multiplication of vectors in $\mathbb{F}^m$.

An IOP is an interactive proof where the verifier has oracle access to some strings provided by the prover. Its relation to zk-SNARKs stems from the results in [BCS16] where it was shown that any IOP can be efficiently compiled into a non-interactive argument in the random oracle model by using Merkle trees [Mer90]. Moreover the transformation, which can be seen as a generalization of the reduction in [Mic94] from PCP, preserves zero knowledge and knowledge soundness. In particular, IOPs can be used to construct zk-SNARKs.

Unfortunately, the IOP constructions above cannot be *directly* instantiated for every field choice as they extensively use Reed-Solomon codes, that requires the existence of enough points in $\mathbb{F}$ and, even worse, the soundness error is often greater than $|\mathbb{F}|^{-1}$ due to polynomial identity tests which implies $|\mathbb{F}| > 2^\lambda$ with λ security parameter. This leaves out, for example, the case of R1CS over $\mathbb{F}_2$. This case is actually interesting as some hash functions and encryption schemes can be interpreted as boolean circuits with relative ease, and then translated to a R1CS. A straight-forward way to overcome this problem, mentioned in [AHIV17], is to simply embed $\mathbb{F}_2$ in a larger field $\mathbb{F}_{2^e}$, for large enough e (where at least $e > \lambda$) and add constraints of the kind $z_i^2 = z_i$ for $i = 1, \dots, n$ to ensure that the witness entries belongs to $\mathbb{F}_2$,[4] and then execute the protocol for R1CS over the larger field.

However this approach seems wasteful, as elements of $\mathbb{F}_{2^e}$ which in principle could encode up to e bits of information are used to represent only one element of $\mathbb{F}_2$. Also, operations over $\mathbb{F}_{2^e}$ are more expensive than those over $\mathbb{F}_2$. Finally one needs the aforementioned additional constraints on the witness, which increase the size of the system.

Since $\mathbb{F}_{2^e}$ is an e-dimensional vector space over $\mathbb{F}_2$, one attempt to improve this would be to interpret vectors in $\mathbb{F}_2^e$ as elements over the larger field $\mathbb{F}_{2^e}$. While this would work for systems that only involve additions (XORs), it fails in general when multiplications (ANDs) are needed too. [5] The technical issue is that for $e > 1$, the ring $\mathbb{F}_2^e$, considered with component-wise addition and multiplication, cannot be embedded via a ring homomorphism into $\mathbb{F}_{2^e}$ (nor any other finite field) since $\mathbb{F}_2^e$ contains zero divisors while fields do not.

A better approach was presented in BooLigero [GSV21] for the case of Ligero [AHIV17]. Their technique allows to encode e bits into roughly $\sqrt{e}$ field elements in $\mathbb{F}_{2^e}$, so that one can use Ligero over $\mathbb{F}_{2^e}$ to treat $\sqrt{e}$ times larger statements over $\mathbb{F}_2$ than the "naïve" method, with roughly the same R1CS size. This however motivates the following question: *can we find embeddings of $\mathbb{F}_2^k$ into $\mathbb{F}_{2^e}$ with a larger embedding rate k/e which allow to produce more efficient IOPs for R1CS over $\mathbb{F}_2$ given an IOP for R1CS over $\mathbb{F}_{2^e}$?*

1.1 Our contributions

In this work we answer the above question in the affirmative using a more efficient embedding that allows us to encode $k \geq e/4$ bits into an element of $\mathbb{F}_{2^e}$. We then present a construction of an IOP for $\mathbb{F}_2$-R1CS satisfiability which makes black-box use of any IOP satisfying mild assumptions for R1CS over larger fields. This leads us to reducing Aurora's argument size up to $1.31 - 1.65\times$ and Ligero's argument size up to $3.71\times$.

More concretely, we can use any *Reed Solomon encoded IOP*, a variant of IOP introduced in [BCR+19], that provides two commonly used sub-protocols: a *generalised lincheck*, which tests linear relations of the form $A_1\mathbf{x}_1 + \ldots + A_n\mathbf{x}_n = \mathbf{b}$ when the verifier has only oracle access to Reed Solomon codewords encoding $\mathbf{x}_i$, and a *rowcheck*, which tests quadratic relations $\mathbf{x} * \mathbf{y} = \mathbf{z}$ when the verifier has oracle access to encodings of $\mathbf{x}, \mathbf{y}, \mathbf{z}$. This includes Ligero[6], Aurora [BCR+19][7]

[4] This is necessary as, for example, $x^2 + x + 1 = 0$ is satisfiable over $\mathbb{F}_4$ but not over $\mathbb{F}_2$, despite the fact that the constraint only involves constants over $\mathbb{F}_2$. Note that interpreting field multiplication as logical AND, the above constrain is equivalent to $x \cdot (x - 1) = 1$, i.e. both x and its negation are true.

[5] This not only includes coordinate-wise products of secret vectors, but also the linear operations $A\mathbf{x}$ in the R1CS system, where A is a public matrix over the larger field.

[6] See [BCR+19] for how to see Ligero as an IOP with these characteristics

[7] We cannot however apply our techniques to IOPs with preprocessing, see comment in Section 1.3.

and Ligero++ [BFH$^+$20] up to minor manipulations to transform their lincheck, see the full version [CG21].

In a nutshell, our embedding technique relies primarily on two components: first, the use of reverse multiplication friendly embeddings (RMFE), introduced in the MPC literature in [CCXY18] and independently in [BMN18] and used in several subsequent works [DLN19, CG20, PS21, DGOT21, ACE$^+$21]. Such algebraic device maps a vector from $\mathbb{F}_2^k$ into an element of a larger field $\mathbb{F}_q = \mathbb{F}_{2^e}$ in a manner such that field additions and products of two encodings in $\mathbb{F}_q$ still encode the component-wise additions and products of the originally vectors from $\mathbb{F}_2^k$, even though the map is not a ring homomorphism. For $k < 100$ we can get RMFEs with $e \approx 3.3k$ (or $e = 4k$ if we insist on e being a power of 2). Second, the notion of *modular lincheck*, an IOPP which we introduce in Section 3.3 and that we believe is of independent interest, to test linear relations modulo an $\mathbb{F}_2$ vector space V contained in $\mathbb{F}_q$, i.e. equations of the form $A\mathbf{x} = \mathbf{b} \mod V^n$ (meaning that each coordinate of the vector $A\mathbf{x} - \mathbf{b}$ is in V).

In conclusion for each of the aforementioned schemes we compare known adaptations to $\mathbb{F}_2$-R1CSs with our general reduction both in terms of argument size and prover complexity. Regarding the proof size we estimate it numerically, see our Python implementation at [Git21]. Regarding prover time we estimate it asymptotically, predicting an improvement factor of $24.7\times$ for Aurora and between $6.9 - 32.5\times$ for Ligero without interactive repetitions.

1.2 Techniques

Reverse multiplication friendly embeddings. A $(k, e)_p$-RMFE is a pair of $\mathbb{F}_p$-linear maps $\varphi : \mathbb{F}_p^k \to \mathbb{F}_{p^e}$ and $\psi : \mathbb{F}_{p^e} \to \mathbb{F}_p^k$ satisfying $\mathbf{x} * \mathbf{y} = \psi(\varphi(\mathbf{x}) \cdot \varphi(\mathbf{y}))$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}_p^k$, where $*$ denotes the component-wise product. The properties automatically imply that φ is injective, hence the name *embedding*. Note that φ is not necessarily a ring homomorphism, i.e. $\varphi(\mathbf{x} * \mathbf{y}) \neq \varphi(\mathbf{x}) \cdot \varphi(\mathbf{y})$ in general. In this paper we extend the notation blockwise to $\Phi : (\mathbb{F}_p^k)^n \to \mathbb{F}_{p^e}^n$ given by $\Phi(\mathbf{x}_1, \ldots, \mathbf{x}_n) = (\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_n))$ and $\Psi : \mathbb{F}_{p^e}^n \to (\mathbb{F}_p^k)^n$ given by $\Psi(x_1, \ldots, x_n) = (\psi(x_1), \ldots, \psi(x_n))$. These satisfy then $\mathbf{x} * \mathbf{y} = \Psi(\Phi(\mathbf{x}) * \Phi(\mathbf{y}))$ for all $\mathbf{x}, \mathbf{y} \in (\mathbb{F}_p^k)^n = \mathbb{F}_p^{kn}$, where the component-wise product on the right side is on $\mathbb{F}_{p^e}^n$.

From $\mathbb{F}_2$-R1CS to a system of statements over $\mathbb{F}_q$. A key ingredient of our result is how to translate the system $A_1\mathbf{w} * A_2\mathbf{w} = A_3\mathbf{w} + \mathbf{b}$ over $\mathbb{F}_2$ into an equivalent set of relations over $\mathbb{F}_q$ that can be efficiently checked. Even with the RMFE in hand, this is not trivial because φ (consequently Φ) is neither a ring homomorphism nor surjective.

Defining $\mathbf{x}_i = A_i\mathbf{w}$, we can split the above statement into the three *linchecks* $A_i\mathbf{w} = \mathbf{x}_i$ and the *rowcheck* $\mathbf{x}_1 * \mathbf{x}_2 = \mathbf{x}_3 + \mathbf{b}$. The prover will start by embedding $\widetilde{\mathbf{w}} = \Phi(\mathbf{w}) \in \mathbb{F}_q^{n/k}$ and $\widetilde{\mathbf{x}}_i = \Phi(\mathbf{x}_i)$. We then need to deal with the following:

First of all, because Φ is not surjective, we need additional constraints to ensure $\widetilde{\mathbf{w}}, \widetilde{\mathbf{x}}_i$ lie in the image of Φ. We can write these in the form $I_{n/k} \cdot \widetilde{\mathbf{w}} \in (\operatorname{Im} \varphi)^{n/k}$ and $I_{m/k} \cdot \widetilde{\mathbf{x}}_i \in (\operatorname{Im} \varphi)^{m/k}$ (where I_ℓ is the ℓ by ℓ identity matrix).

Then, because Φ is not a ring homomorphism, we can *not* simply translate $\mathbf{x}_1 * \mathbf{x}_2 = \mathbf{x}_3 + \mathbf{b}$ into $\widetilde{\mathbf{x}}_1 * \widetilde{\mathbf{x}}_2 = \widetilde{\mathbf{x}}_3 + \Phi(\mathbf{b})$, as this is not true in general. Instead, we need to use the RMFE "product recovery map" ψ. Setting $\mathbf{t} = \widetilde{\mathbf{x}}_1 * \widetilde{\mathbf{x}}_2$, we show that the rowcheck statement is equivalent to the modular linear relation $\mathbf{t} - u \cdot \widetilde{\mathbf{x}}_3 = u \cdot \Phi(\mathbf{b}) \mod (\operatorname{Ker}\psi)^{m/k}$ where $u = \varphi(\mathbf{1}) \in \mathbb{F}_q$, $\mathbf{1}$ is the all-one vector and Ker denotes the kernel.

Similarly, we show that each lincheck $A_i\mathbf{w} = \mathbf{x}_i$ can be translated into $\widetilde{A_i}\widetilde{\mathbf{w}} - \widetilde{I_m}\widetilde{\mathbf{x}}_i \in (\operatorname{Ker} S \circ \psi)^m$, where $\widetilde{A_i}, \widetilde{I_m}$ are the result of applying Φ to A_i, I_m row-wise and S is the map summing the k components of a vector in $\mathbb{F}_2^k$.

Modular linear test The sketched characterization above implies that providing a way to test linear modular relations over $\mathbb{F}_q$ yields the desired IOP as the prover could provide oracle access to encodings of $\widetilde{\mathbf{w}}, \widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2, \widetilde{\mathbf{x}}_3, \mathbf{t}$ and then convince the verifier that all those constraints are satisfied. To test $\mathbf{x} = \mathbf{0} \mod V^n$, a standard approach would consist in proving that a random linear combination of its coordinates belongs to V. However, we are dealing with a $\mathbb{F}_2$-vector space, and this translates into a soundness error of 1/2. In order to decrease it to $2^{-\lambda}$, we could check λ independent linear combinations, which involves λn random bits. In Section 3.3 we describe how to reduce the required random bits to $\Theta(\lambda)$ by using a certain family of almost universal linear hash functions, and achieve zero knowledge by adding a masking term.

Optimizations The above techniques require a total of 8 modular linchecks and a rowcheck. In Section 4, we introduce several modifications, the main of which is to reduce the number of modular linchecks to just 3. The observation is that we can test several equations of the form $A\mathbf{x}_i = \mathbf{b}_i \mod V^{n_i}$ (with common V) all at once by checking $\sum R_i(A\mathbf{x_i} - \mathbf{b}_i) \in V^\lambda$ for appropriately chosen matrices R_i. Additionaly, we compress messages sent by the prover using the structure of these vector spaces V, which comes from our use of an RMFE.

1.3 Other related work

Our work provides a significant reduction of the proof size with respect to BooLigero [GSV21]. Applying our construction to Ligero for an $\mathbb{F}_2$-R1CS consisting of 2^{20} constraints we measure proofs $3.71\times$ shorter than plain Ligero and $3.03\times$ smaller than BooLigero. We also stress that in contrast to [GSV21] we present a general reduction that can be applied to a larger class of protocols.
Regarding the use of RMFE, to the best of our knowledge only the recent work [DGOT21] applied this tool in the IOP framework (see their Appendix A). However, their use is restricted to their own protocol, which follows the MPC-in-the-head paradigm introduced in [IKOS07], and cannot be applied directly to other existing IOPs such as Aurora. Furthermore, this optimisation is only considered in the multi-instance case while in our work we manage to integrate the RMFE also for a single instance.
We also remark that even though our construction captures essentially any IOPs that provides a lincheck and a rowcheck, it still cannot be applied out of the box

to zk-SNARKS with preprocessing such as Fractal [COS20] or Spartan [Set20]. The reason is that we use the given linchecks to test a randomised relation depending on the random coins of the verifier. This significantly affects the usefulness of any pre-computation. We believe however that this issue can be overcome in a non black-box way with different techniques, a problem that we leave for future work.

2 Preliminaries

The set $\{1, \dots, n\}$ is called $[n]$. Vectors are denoted with boldface font. $\mathbf{v} * \mathbf{w}$ denotes the coordinate-wise product of two vectors of the same length, and $\|\mathbf{v}\|$ is the Hamming weight of $\mathbf{v}$. $\mathbf{1}_k$ is the vector of k 1's. Matrices are denoted with capital letters, $A^\top$ is the transpose of A and I_n is the n by n identity matrix. Given q a prime power, $\mathbb{F}_q$ is a field of q elements. When $q = p^e$, $\mathbb{F}_p$ can be seen as a subset of $\mathbb{F}_q$ and $\mathbb{F}_q$ can be treated as an $\mathbb{F}_p$ vector space of dimension e. $V \leq \mathbb{F}_q$ means that V is an $\mathbb{F}_p$-vector subspace of $\mathbb{F}_q$. $a = b \mod V$ means that $a - b \in V$, and for vectors of length m, $\mathbf{a} = \mathbf{b} \mod V^m$ iff $a_i = b_i \mod V$ for all $i \in [m]$. Given an $\mathbb{F}_p$-linear map $L : V \to W$ its kernel is $\operatorname{Ker} L = \{\mathbf{x} \in V : L(\mathbf{x}) = 0\}$ and its image is $\operatorname{Im} L = \{\mathbf{y} \in W : \mathbf{y} = L(\mathbf{x}) \text{ for some } \mathbf{x} \in V\}$. Given a polynomial $\widehat{f} \in \mathbb{F}_q[x]$ and $L \subseteq \mathbb{F}_q$ we denote $\widehat{f}_{|L} = (\widehat{f}(\alpha))_{\alpha \in L}$ its evaluation over L. The Reed-Solomon code over L of rate $\rho \in [0, 1]$ is the set $\mathsf{RS}_{\mathbb{F}_q,L,\rho} := \{\widehat{f}_{|L} : \widehat{f} \in \mathbb{F}_q[x],\ \deg \widehat{f} < \rho|L|\}$. We will typically encode vectors $\mathbf{v}$ of length $m < \rho|L|$ as codewords from $\mathsf{RS}_{\mathbb{F}_q,L,\rho}$ by sampling a $f \in \mathsf{RS}_{L,\rho}$ such that $\widehat{f}_{|H} = \mathbf{v}$. $\mathbb{F}_q^H$ denotes the set of vectors over $\mathbb{F}_q$ with coordinates indexed by H and $\mathbb{F}_q^{H_1 \times H_2}$ is the set of matrices with rows and columns indexed by H_1 and H_2 respectively. Finally $\mathrm{FFT}(\mathbb{F}, n)$ denotes the number of field operations required to perform a fast Fourier transform over a set of size n, see [GM10].

2.1 Reverse multiplication friendly embedding

We now recall the notion of reverse multiplication friendly embedding from [CCXY18]. Its purpose is to 'reconcile' the coordinate-wise multiplicative structure of a ring $\mathbb{F}_p^k$ and the finite field structure of an extension $\mathbb{F}_{p^e}$ of $\mathbb{F}_p$.

Definition 1. *Given a prime power p and $k, e \in \mathbb{N}$ a* ***Reverse Multiplication-Friendly Embedding****, denoted $(k, e)_p$-RMFE, is a pair of $\mathbb{F}_p$-linear maps $\varphi : \mathbb{F}_p^k \to \mathbb{F}_{p^e}$, $\psi : \mathbb{F}_{p^e} \to \mathbb{F}_p^k$ such that for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}_p^k$, it holds that*

$$\mathbf{x} * \mathbf{y} = \psi(\varphi(\mathbf{x}) \cdot \varphi(\mathbf{y})).$$

That is, one can embed $\mathbb{F}_p^k$ into $\mathbb{F}_{p^e}$ via a linear map φ so that the product in $\mathbb{F}_{p^e}$ of the images of any two vectors $\mathbf{x}, \mathbf{y}$ carries information about their component-wise product $\mathbf{x} * \mathbf{y}$, and this can be recovered applying ψ to that field product. However, φ is in general not a ring homomorphism and therefore $\psi \neq \varphi^{-1}$. For notational convenience, we extend both φ and ψ to maps Φ, Ψ as follows. Given vectors $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in (\mathbb{F}_p^k)^n$ and $\mathbf{z} = (z_1, \ldots, z_n) \in (\mathbb{F}_{p^e})^n$ we define

$$\Phi(\mathbf{x}) := (\varphi(\mathbf{x}_1), \ldots, \varphi(\mathbf{x}_n)) \in (\mathbb{F}_{p^e})^n, \qquad \Psi(\mathbf{z}) := (\psi(z_1), \ldots, \psi(z_n)) \in (\mathbb{F}_p^k)^n.$$

The following properties of these extended functions will be key in Section 3.1 to transform a $\mathbb{F}_2$-R1CS system into a system of equations over $\mathbb{F}_{2^e}$. Note in particular (2) and (4) characterize respectively coordinatewise and inner products over $\mathbb{F}_p$ in terms of the corresponding operations over $\mathbb{F}_{p^e}$. The lemma follows quite directly from the definitions and a proof appears in the full version [CG21].

Lemma 1. *The following holds for all positive $n \in \mathbb{N}$:*

1. *The maps φ and Φ are injective. The maps ψ and Ψ are surjective.*
2. *For all $\mathbf{x}$, $\mathbf{y} \in (\mathbb{F}_p^k)^n$, $\mathbf{x} * \mathbf{y} = \Psi(\Phi(\mathbf{x}) * \Phi(\mathbf{y}))$ where the $*$ product in the right-hand side is component-wise in $(\mathbb{F}_{p^e})^n$, i.e. in each component we use the field product in $\mathbb{F}_{p^e}$.*
3. *Let $u = \varphi(\mathbf{1}_k) \in \mathbb{F}_{p^e}$.[8] Then for all $\mathbf{x} \in (\mathbb{F}_p^k)^n$ we have $\mathbf{x} = \Psi(u \cdot \Phi(\mathbf{x}))$.*
4. *Let $S : \mathbb{F}_p^k \to \mathbb{F}_p$ be given by $S(x_1, x_2, \ldots, x_k) = x_1 + x_2 + \cdots + x_k$. Then for all $\mathbf{x}$, $\mathbf{y} \in (\mathbb{F}_p^k)^n$, the inner product $\mathbf{x}^\top \mathbf{y}$ can be written as*

$$\mathbf{x}^\top \mathbf{y} = S \circ \psi(\Phi(\mathbf{x})^\top \Phi(\mathbf{y}))$$

As for the existence of RMFEs, in our case of interest $p = 2$ one can obtain the following parameters by concatenation of polynomial interpolation techniques [CCXY18, CG20] (for asymptotics and other results see the full version [CG21]):

Lemma 2. *For all $r \leq 33$, there exists a $(3r, 10r)_2$-RMFE. For all $a \leq 17$ there exists a $(2a, 8a)_2$-RMFE. For all $b \leq 65$ there exists a $(3b, 12b)_2$-RMFE.*

This yields RMFEs with parameters $(48, 192)$, $(48, 160)$ and $(32, 128)$, setting $r = a = b = 16$, that we will concretely use to evaluate our reduction.

2.2 R1CS, Lincheck and Rowcheck

We now recall the main relations used in recent IOP-based[9]SNARKs like [BCR+19, AHIV17]. The first one is the *rank 1 constraints system*, or R1CS, that defines an NP-complete language closely related to arithmetic circuit satisfiability. Here we present an equivalent affine version that requires for $A_1, A_2, A_3 \in \mathbb{F}^{m,n}$ and $\mathbf{b} \in \mathbb{F}^m$ to exhibit a vector $\mathbf{w} \in \mathbb{F}^n$ such that $A_1\mathbf{w} * A_2\mathbf{w} = A_3\mathbf{w} + \mathbf{b}$. Formally

Definition 2. *We define the affine R1CS relation as the set*

$$\mathcal{R}_{\mathsf{R1CS}} = \{((\mathbb{F}, m, n, A_1, A_2, A_3, \mathbf{b}), \mathbf{w}) \ : \ A_i \in \mathbb{F}^{m,n}, \ A_1\mathbf{w} * A_2\mathbf{w} = A_3\mathbf{w} + \mathbf{b}\}.$$

[8] Note that u is not necessarily equal to 1.
[9] See [BCS16, BCR+19] for the rigorous definition, or the full version [CG21] for a simplified explanation.

Instead of directly providing a proof system for R1CS, two intermediate relations, *lincheck* and *rowcheck*, are defined and for which [BCR$^+$19] constructs RS-encoded IOPPs.[10] These are then used as building blocks to produce a RS-encoded IOP for the R1CS relation, which in turn can be combined with a low degree test, such as FRI [BBHR18a] or [BGKS20], to make a standard IOP for R1CS. The complexity of this reduction depends on the so-called *max rates*, two parameters related to the degrees of polynomials and the relations which are tested.

The lincheck relation requires that the witnesses $f_1, f_2 \in \mathsf{RS}_{L,\rho}$ encode over $H_1, H_2 \subseteq \mathbb{F}_q$ two vectors $\mathbf{x}_1, \mathbf{x}_2$ (i.e. $\widehat{f}_{i|H_i} = \mathbf{x}_i$) which satisfy a given linear constraint $M\mathbf{x}_1 = \mathbf{x}_2$. The rowcheck relation requires that witnesses $f_1, f_2, f_3 \in \mathsf{RS}_{L,\rho}$ encode over $H \subseteq \mathbb{F}_q$ three vectors $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ such that $\mathbf{x}_1 * \mathbf{x}_2 = \mathbf{x}_3$. For efficiency reasons, depending on the concrete instantiations of Aurora and FRI, in both definitions below L, H_1, H_2, H are taken to be $\mathbb{F}_2$-affine subspaces of $\mathbb{F}_q$.

Definition 3. *We define $\mathcal{R}_{\mathsf{Lin}}$ as the set of tuples $((\mathbb{F}_q, L, H_1, H_2, \rho, M), (f_1, f_2))$ such that $L, H_i \subseteq \mathbb{F}_q$ are affine subspaces, $H_i \cap L = \varnothing$ for $i \in \{1, 2\}$, $f_i \in \mathsf{RS}_{L,\rho}$, $M \in \mathbb{F}_q^{H_1 \times H_2}$ and the linear relationship $\widehat{f}_{1|H_1} = M \cdot \widehat{f}_{2|H_2}$ holds.*

Definition 4. *We define $\mathcal{R}_{\mathsf{Row}}$ as the set of tuples $((\mathbb{F}_q, L, H, \rho), (f_1, f_2, f_3))$ such that $L, H \subseteq \mathbb{F}_q$ are disjoint affine subspaces, $f_i \in \mathsf{RS}_{L,\rho}$ for $i \in \{1, 2, 3\}$ and the quadratic relationship $\widehat{f}_{1|H} * \widehat{f}_{2|H} = \widehat{f}_{3|H}$ holds.*

RS-encoded IOPPs $(\mathsf{P}_{\mathsf{Lin}}, \mathsf{V}_{\mathsf{Lin}})$ and $(\mathsf{P}_{\mathsf{Row}}, \mathsf{V}_{\mathsf{Row}})$ for the two relations above are provided in [BFH$^+$20, BCR$^+$19] and in [AHIV17] up to minor adaptations in the second case. We will need a generalisation of $\mathcal{R}_{\mathsf{Lin}}$ that tests relations of the form $M_1\mathbf{x}_1 + \ldots + M_h\mathbf{x}_h = \mathbf{b}$ (for $h = 2$, $M_1 = -I$ and $\mathbf{b} = \mathbf{0}$ we get back the standard lincheck).

Definition 5. *$\mathcal{R}_{\mathsf{Lin}_h}$ is the set of tuples $((\mathbb{F}_q, L, H_0, H_i, \rho, M_i, \mathbf{b})_{i=1}^h, (f_i)_{i=1}^h)$ such that $L, H_0, H_i \leq \mathbb{F}_q$, $L \cap H_0 = L \cap H_i = \varnothing$ for all $i \in \{1, \ldots, h\}$, $f_i \in \mathsf{RS}_{L,\rho}$, $M_i \in \mathbb{F}_q^{H_0 \times H_i}$ and the linear relationship $\sum_{i=1}^h M_i \cdot \widehat{f}_{i|H_i} = \mathbf{b}$ holds.*

The lincheck protocol presented in Aurora can be generalised to capture this variant, as shown in the full version of this paper.

3 Simplified Construction

In the rest of the paper we aim at describing an efficient RS-encoded IOP for $\mathbb{F}_2$-R1CS. As the only tools we assume are a lincheck and a rowcheck over a large enough field, our first step in Section 3.1 is to characterise $\mathbb{F}_2$-R1CS in terms of one quadratic relation over $\mathbb{F}_q$ and a set of linear relations modulo some vector

[10] Reed-Solomon IOPs are IOPs where soundness is guaranteed only when the messages sent by the prover are oracles to codewords of a Reed-Solomon code. Reed-Solomon IOPPs (proofs of proximity) additionally provide oracle access to the witness, also a set of Reed-Solomon codewords, to the verifier.

space $V \leq \mathbb{F}_q$. An RS-encoded IOPP to test the latter conditions is provided in Section 3.3. Finally a simple solution that uses naively the above IOPP is provided in Section 3.4. Even if suboptimal, we see this as a useful stepping stone to better present the efficient version in Section 4.2.

3.1 Characterisation of R1CS

In the following we assume (φ, ψ) to be a $(k, e)_2$-RMFE, where $q = 2^e$, and recall that Φ, Ψ denote the block-wise application of φ and ψ, cf. Section 2.1.

Theorem 1. *Let $A_1, A_2, A_3 \in \mathbb{F}_2^{m,n}$, $\mathbf{b} \in \mathbb{F}_2^m$ with m, n multiples of k. Then there exists $\mathbf{w} \in \mathbb{F}_2^n$ such that $((\mathbb{F}_2, m, n, A_1, A_2, A_3, \mathbf{b}), \mathbf{w}) \in \mathcal{R}_{\mathsf{R1CS}}$ if and only if there exist $\widetilde{\mathbf{w}} \in \mathbb{F}_q^{n/k}$ and $\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2, \widetilde{\mathbf{x}}_3, \mathbf{t} \in \mathbb{F}_q^{m/k}$ satisfying*

$$\begin{aligned}
\widetilde{\mathbf{x}}_1 * \widetilde{\mathbf{x}}_2 &= \mathbf{t} && && (1)\\
\widetilde{\mathbf{w}} &= \mathbf{0} \mod (\operatorname{Im}\varphi)^{n/k} && && (2)\\
\widetilde{\mathbf{x}}_i &= \mathbf{0} \mod (\operatorname{Im}\varphi)^{m/k} && \forall i \in \{1,2,3\} && (3)\\
\widetilde{A}_i \widetilde{\mathbf{w}} - \widetilde{I}_m \widetilde{\mathbf{x}}_i &= \mathbf{0} \mod (\operatorname{Ker} S \circ \psi)^m && \forall i \in \{1,2,3\} && (4)\\
\mathbf{t} - u \widetilde{\mathbf{x}}_3 &= u \widetilde{\mathbf{b}} \mod (\operatorname{Ker}\psi)^{m/k} && && (5)
\end{aligned}$$

where $\widetilde{\mathbf{b}} = \Phi(\mathbf{b}) \in \mathbb{F}_q^{m/k}$, $u = \varphi(\mathbf{1}_k) \in \mathbb{F}_q$, $\widetilde{A}_i \in \mathbb{F}_q^{m,n/k}$ is the matrix obtained by applying Φ row-wise to A_i, and $\widetilde{I}_m \in \mathbb{F}_q^{m,m/k}$ is the matrix obtained by applying Φ row-wise to the identity matrix $I_m \in \mathbb{F}_2^{m,m}$. Moreover if $\mathbf{w}$ is a witness for the R1CS then $\widetilde{\mathbf{w}} = \Phi(\mathbf{w})$, $\widetilde{\mathbf{x}}_i = \Phi(A_i\mathbf{w})$, $\mathbf{t} = \widetilde{\mathbf{x}}_1 * \widetilde{\mathbf{x}}_2$ satisfy the conditions above.

The proof appears in the full version [CG21], but we remark Equations (2), (3) are equivalent to saying $\widetilde{\mathbf{w}} = \Phi(\mathbf{w})$, $\widetilde{\mathbf{x}}_i = \Phi(\mathbf{x}_i)$ for some $\mathbf{w}$, $\mathbf{x}_i$; Equations (1) and (5) encode $\mathbf{x}_1 * \mathbf{x}_2 = \mathbf{x}_3 + \mathbf{b}$ (the rowcheck) and the latter is derived using properties (2) and (3) in Lemma 1; while Equations (4) encode $A_i\mathbf{w} = \mathbf{x}_i$ (the lincheck) and are derived from property (4) in Lemma 1.

3.2 Linear Hashing

We now adapt linear checks to small fields. A common technique to test $A\mathbf{x} = \mathbf{b}$ over $\mathbb{F}_q$ is to sample a random vector $\mathbf{r} \in \mathbb{F}_q^m$ and check $\mathbf{r}^\top A\mathbf{x} = \mathbf{r}^\top \mathbf{b}$. Alternatively one can set $\mathbf{r} = (1, r, \ldots, r^{m-1})$ for $r \leftarrow^{\$} \mathbb{F}_q$ to save randomness. The soundness errors of these approaches are respectively $1/q$ and $(m-1)/q$, which are too large if q is small as in our case. Therefore they need to be adapted. With this aim in mind, let $\vartheta : \mathbb{F}_2^\lambda \to \mathbb{F}_{2^\lambda}$ be an isomorphism of $\mathbb{F}_2$-linear spaces[11]. For any $\alpha \in \mathbb{F}_{2^\lambda}$ define $R_\alpha^{(m)} : \mathbb{F}_2^{\lambda m} \to \mathbb{F}_2^\lambda$ such that

$$R_\alpha^{(m)}(\mathbf{x}_1, \ldots, \mathbf{x}_m) \;=\; \vartheta^{-1}\big(\alpha\vartheta(\mathbf{x}_1) + \ldots + \alpha^m \vartheta(\mathbf{x}_m)\big).$$

[11] Observe here we do not worry about their multiplicative structures

Seeing this function as a matrix in $\mathbb{F}_2^{\lambda,\lambda m}$, we can apply it to vectors in $\mathbb{F}_q^{\lambda m}$, i.e., if $R_\alpha^{(m)} = (r_{i,j}) \in \mathbb{F}_2^{\lambda,\lambda m}$ and $\mathbf{x} = (x_j)_{j=1}^{\lambda m} \in \mathbb{F}_q^{\lambda m}$ then $R_\alpha^{(m)}\mathbf{x} = \left(\sum_{j=1}^{\lambda m} r_{i,j}x_j\right)_{i=1}^{\lambda}$. This family of linear functions satisfies the following properties

Proposition 1. *Let $V \leq \mathbb{F}_q$ be an $\mathbb{F}_2$ vector subspace, $\mathbf{y} \in \mathbb{F}_q^\lambda$, $\mathbf{x} \in \mathbb{F}_q^{\lambda m} \setminus V^{\lambda m}$ and $\alpha \sim U(\mathbb{F}_{2^\lambda})$, then* $\Pr\left[R_\alpha^{(m)}\mathbf{x} = \mathbf{y} \mod V^\lambda\right] \leq 2^{-\lambda} \cdot m$

Proposition 2. *Let $V \leq \mathbb{F}_q$ be an $\mathbb{F}_2$ vector subspace, $\mathbf{y} \in \mathbb{F}_q^\lambda$, $\mathbf{x}_i \in \mathbb{F}_q^{\lambda m_i}$ for $i \in [h]$ such that $\mathbf{x}_j \notin V^{\lambda m_j}$ for some j. Then $\alpha_i \sim U(\mathbb{F}_{2^\lambda})$ implies*

$$\Pr\left[R_{\alpha_1}^{(m_1)}\mathbf{x}_1 + \ldots + R_{\alpha_h}^{(m_h)}\mathbf{x}_h = \mathbf{y} \mod V^\lambda\right] \leq 2^{-\lambda} \cdot \max\{m_i : i \in [h]\}.$$

3.3 Modular Lincheck

In this section we provide an RS-encoded IOPP that generalises the Lincheck to linear relations of the form $M_1\mathbf{x}_1 + \ldots + M_h\mathbf{x}_h = \mathbf{b}$ modulo an $\mathbb{F}_2$ vector space $V \leq \mathbb{F}_q$, where the verifier has oracle access to an encoding of $\mathbf{x}_i$ for each i.

Definition 6. *The Modular Lincheck relation is the set $\mathcal{R}_{\mathsf{MLin}_h}$ of all tuples $((\mathbb{F}_q, L, H_0, H_i, \rho, M_i, \mathbf{b}, V)_{i=1}^h, (f_i)_{i=1}^h)$ such that $L, H_0, H_i \subseteq \mathbb{F}_q$ are affine $\mathbb{F}_2$-spaces with $L \cap H_i = \varnothing$, $\rho \in [0,1)$, $M_i \in \mathbb{F}_q^{H_0 \times H_i}$, $f_i \in \mathsf{RS}_{L,\rho}$ and $\sum_{i=1}^h M_i \widehat{f}_{i|H_i} = \mathbf{b} \mod V^{H_0}$.*

Consider the simpler statement $\mathbf{x} = \mathbf{0} \mod V^H$, i.e. $\mathbf{x} \in V^H$, and the following proof: the verifier samples a random $R \sim U(\mathbb{F}_2^{H_0' \times H})$, and receives $\mathbf{v} = R\mathbf{x}$ from the prover; the verifier then checks $\mathbf{v} \in V^{H_0'}$ and then runs a lincheck to test $\mathbf{v} = R\mathbf{x}$. In order to make this zero knowledge, we add a masking codeword g sampled from $\mathsf{Mask}(L, \rho, H_0', V) = \{f \in \mathsf{RS}_{L,\rho} : \widehat{f}_{|H_0'} \in V^{H_0'}\}$ so that the sender first sends an oracle to g, receives R, and sends $\mathbf{v} = R\mathbf{x} + \widehat{g}_{|H_0'}$ in plain. In the general case we replace $\mathbf{x}$ with $\sum_{i=1}^h M_i \widehat{f}_{i|H_i} - \mathbf{b}$ and, for efficiency reasons, the random matrix R with R_α obtaining the protocol in Figure 1.

From the above observations, the protocol has the following properties, where soundness comes from Proposition 2. See the full paper for a rigorous proof.

Theorem 2. *Protocol 1 is an RS-encoded IOPP for the relation $\mathcal{R}_{\mathsf{MLin}_h}$ that upon setting $|H_0'| = \lambda$ has the following parameters:*

Rounds	$= 2$
Proof Length	$= 3\|L\|$ *elements of* $\mathbb{F}_q$
Randomness	$= \lambda + 2\log q$ *bits*
Soundness Error	$= \lceil m/\lambda \rceil 2^{-\lambda} + \lambda q^{-1}$
Prover Time	$= \mathrm{FFT}(\mathbb{F}_q, \|L\|) + \sum_{i=1}^h \\|M_i\\| + \\|\mathbf{b}\\| + \lambda \sum_{i=1}^n \|H_i\| + T^{\mathsf{P}}_{\mathsf{Lin}_{h+1}}$
Verifier Time	$= \lambda \dim V + \sum_{i=1}^h \\|M_i\\| + \\|\mathbf{b}\\| + T^{\mathsf{V}}_{\mathsf{Lin}_{h+1}}$
Max Rates	$= \left(\rho + \lambda\|L\|^{-1}, \rho + (\lambda + \|H\|)\|L\|^{-1}\right)$

where $H = \mathsf{span}\,(H_1, \ldots, H_h, H_0')$ and $T^{\mathsf{P}}_{\mathsf{Lin}_{h+1}}, T^{\mathsf{V}}_{\mathsf{Lin}_{h+1}}$ denotes the costs of running respectively $\mathsf{P}_{\mathsf{Lin}_{h+1}}$ and $\mathsf{V}_{\mathsf{Lin}_{h+1}}$.

$\mathsf{P}_{\mathsf{Mlin}_h}((\mathsf{pp}, M_i, \mathbf{b}, V, f_i)_{i=1}^h)$		$\mathsf{V}_{\mathsf{Mlin}_h}^{f_1,\dots,f_h}((\mathsf{pp}, M_i, \mathbf{b}, V)_{i=1}^h)$
Agree on $H_0' \subseteq \mathbb{F}_q : H_0' \cap L = \varnothing$		Agree on $H_0' \subseteq \mathbb{F}_q : H_0' \cap L = \varnothing$
$M_{h+1} \leftarrow I_{H_0'},\ H_{h+1} \leftarrow H_0'$		$M_{h+1} \leftarrow I_{H_0'},\ H_{h+1} \leftarrow H_0'$
$\rho' \leftarrow \rho + \lvert H_0'\rvert\lvert L\rvert^{-1}$		$\rho' \leftarrow \rho + \lvert H_0'\rvert\lvert L\rvert^{-1}$
$\mathsf{pp}' \leftarrow (\mathbb{F}_q, L, H_0', H_i, \rho')_{i=1}^{h+1}$		$\mathsf{pp}' \leftarrow (\mathbb{F}_q, L, H_0', H_i, \rho')_{i=1}^{h+1}$
$f_{h+1} \xleftarrow{\$} \mathsf{Mask}(L, \rho', H_0', V)$	$\xrightarrow{f_{h+1}}$	
	$\xleftarrow{\alpha}$	$\alpha \xleftarrow{\$} \mathbb{F}_{2^\lambda}$
$\mathbf{v} \leftarrow R_\alpha \left[\sum_{i=1}^h M_i \widehat{f}_{i\lvert H_i} - \mathbf{b}\right] + \widehat{f}_{h+1\lvert H_0'}$	$\xrightarrow{\mathbf{v}}$	
		If $\mathbf{v} \notin V^{H_0'}$ return $\bot$
$M' \leftarrow ((R_\alpha M_i)_{i=1}^h, I_{H_0'})$		$M' \leftarrow ((R_\alpha M_i)_{i=1}^h, I_{H_0'})$
Execute:		Execute:
$\mathsf{P}_{\mathsf{Lin}_{h+1}}(\mathsf{pp}', M', R_\alpha\mathbf{b} + \mathbf{v}, (f_i)_{i=1}^{h+1})$		$\mathsf{V}_{\mathsf{Lin}_{h+1}}^{f_1,\dots,f_{h+1}}(\mathsf{pp}', M', R_\alpha\mathbf{b} + \mathbf{v})$

Fig. 1. RS-encoded IOPP for $\mathcal{R}_{\mathsf{Mlin}_h}$ with $\mathsf{pp} = (\mathbb{F}_q, L, H_0, (H_i)_{i=1}^h, \rho)$

3.4 An RS-encoded IOP for R1CS from Modular Lincheck

Given RS-encoded IOPP for Modular Lincheck and Rowcheck we briefly sketch how to build a simple RS-encoded IOP for $\mathbb{F}_2$-R1CS. By Theorem 1 we know that a given system, defined by $A_1, A_2, A_3 \in \mathbb{F}_2^{m,n}$, $\mathbf{b} \in \mathbb{F}_2^m$ is satisfied if and only if there exists $\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2, \widetilde{\mathbf{x}}_3, \mathbf{t} \in \mathbb{F}_q^{m/k}$ and $\widetilde{\mathbf{w}} \in \mathbb{F}_q^{n/k}$ that satisfy equations 1-5.

Thus we let the prover initially compute the extended witness $\mathbf{x}_i = A_i\mathbf{w}$, apply block-wise the RMFE to get $\widetilde{\mathbf{x}}_i = \Phi(\mathbf{x}_i)$, $\widetilde{\mathbf{w}} = \Phi(\mathbf{w})$ and finally set $\mathbf{t} = \widetilde{\mathbf{x}}_1 * \widetilde{\mathbf{x}}_2$. Next, it picks two affine subspaces $H_1, H_2 \subseteq \mathbb{F}_q$ of sizes $m/k, n/k$ and sample five codewords $f_{\widetilde{\mathbf{x}}_i}, f_{\mathbf{t}}, f_{\widetilde{\mathbf{w}}}$ such that $\widehat{f}_{\widetilde{\mathbf{x}}_i\lvert H_1} = \widetilde{\mathbf{x}}_i$, $\widehat{f}_{\mathbf{t}\lvert H_1} = \mathbf{t}$ and $\widehat{f}_{\widetilde{\mathbf{w}}\lvert H_2} = \widetilde{\mathbf{w}}$.
Finally it provides oracle access to these codewords to the verifier and they both run:

- One rowcheck to test $\widetilde{\mathbf{x}}_1 * \widetilde{\mathbf{x}}_2 = \mathbf{t}$.
- Four modular lincheck to test $I_{m/k} \cdot \widetilde{\mathbf{x}}_i \in (\operatorname{Im}\varphi)^{H_1}$ and $I_{n/k} \cdot \widetilde{\mathbf{w}} \in (\operatorname{Im}\varphi)^{H_2}$.
- Three modular lincheck to test that $\tilde{A}_i \cdot \widetilde{\mathbf{w}} - \tilde{I}_m \cdot \widetilde{\mathbf{x}}_i \in (\operatorname{Ker} S \circ \psi)^m$.
- One modular lincheck to check $I_{m/k} \cdot \mathbf{t} - (uI_{m/k}) \cdot \widetilde{\mathbf{x}}_3 = u\widetilde{\mathbf{b}} \mod (\operatorname{Ker}\psi)^{H_1}$.

Correctness and soundness of the above protocol follows from Theorem 1, while Zero Knowledge against β queries can be achieved setting the rate of $f_{\widetilde{\mathbf{x}}_i}, f_{\mathbf{t}}$ to $\frac{m/k+\beta}{\lvert L\rvert}$ and the rate of $f_{\widetilde{\mathbf{w}}}$ to $\frac{n/k+\beta}{\lvert L\rvert}$.

4 Efficient Construction

4.1 Batching Modular Linchecks and Packing Vectors

The protocol above requires a total of 8 modular Linchecks. In this section we show how to reduce the number of required modular linchecks to three, by batching proofs of relations modulo the same vector space: we aim at designing an RS-encoded IOPP for a relation of the form: $\forall i \in [h], A_i\mathbf{x}_i = \mathbf{b}_i \mod V^{m_i}$.

We propose the following: as before the prover begins by sending a codeword that encodes a masking term $\mathbf{y} \sim U(V^\lambda)$. The verifier then chooses h matrices $R_{\alpha_1}, \ldots, R_{\alpha_h}$ and the prover replies by sending $\mathbf{v} = \sum_{i=1}^{h} R_{\alpha_i}(A_i\mathbf{x}_i - \mathbf{b}_i) + \mathbf{y}$. Finally the verifier checks if $\mathbf{v} \in V^\lambda$ and both parties executes a lincheck to test the above relation. Informally security follows as in the single modular lincheck from Section 3.3, except that for soundness we use Proposition 2.

To further improve the complexities, we now show how to reduce the size of vectors sent in plain by the prover in the (batched) modular lincheck. Recalling $u = \varphi(\mathbf{1}_k)$ we point out $\operatorname{Ker}\psi$ and $u \cdot \operatorname{Im}\varphi$ intersect only in 0, because $\psi(u \cdot \varphi(\mathbf{v})) = \mathbf{1}_k * \mathbf{v} = \mathbf{v}$. Therefore $\mathbb{F}_q$ is the direct sum of $\operatorname{Ker}\psi$ and $(u \cdot \operatorname{Im}\varphi)$. Then given $\mathbf{x} \in (\operatorname{Im}\varphi)^n$ and $\mathbf{y} \in (\operatorname{Ker}\psi)^n$, we just need to send $\mathbf{z} = u\mathbf{x} + \mathbf{y}$. Given $\mathbf{z}$ one can extract $\mathbf{x} = \Phi(\Psi(\mathbf{z}))$ and $\mathbf{y} = \mathbf{z} - u\mathbf{x}$, where the former equation is justified by observing that, if we call $\mathbf{v} \in \mathbb{F}_2^{kn}$ such that $\mathbf{x} = \Phi(\mathbf{v})$, then $\Phi(\Psi(\mathbf{z})) = \Phi(\Psi(u\mathbf{x} + \mathbf{y})) = \Phi(\Psi(u \cdot \Phi(\mathbf{v}))) = \Phi(\mathbf{v}) = \mathbf{x}$, where the second equality following from $\mathbf{y} \in (\operatorname{Ker}\psi)^n$ and the third one from Lemma 1.

4.2 An Efficient RS-encoded IOP for R1CS

With the two ideas presented so far we can now improve the protocol sketched in Section 3.4. We batch linchecks in three groups, testing equations modulo $\operatorname{Im}\varphi$, $\operatorname{Ker} S \circ \psi$ and $\operatorname{Ker}\psi$ respectively. Moreover we observe that the masking terms of these tests can be aggregated. To do so we choose three disjoint affine subspaces H_1', H_2', H_3' of size λ and sample g from the set $\mathsf{BMask}\,(L, \rho, H_1', H_2', H_3', \varphi, \psi)$ defined as

$$\left\{ f \in \mathsf{RS}_{L,\rho} \;:\; \widehat{f}_{|H_1'} \in (\operatorname{Im}\varphi)^{H_1'} \;,\; \widehat{f}_{|H_2'} \in (\operatorname{Ker} S \circ \psi)^{H_2'} \;,\; \widehat{f}_{|H_3'} \in (\operatorname{Ker}\psi)^{H_3'} \right\}.$$

In the following protocol we let $\rho_1 = (m/k + \beta)|L|^{-1}$, $\rho_2 = (n/k + \beta)|L|^{-1}$ and $\rho_3 = (3\lambda + \beta)|L|^{-1}$ be the three rates used.

Theorem 3. *Protocol 2 is an RS-encoded IOP for the relation $\mathcal{R}_{\mathsf{R1CS}}$ which, using Aurora's lincheck and rowcheck, achieves the following parameters*

Rounds	=	3
Proof Length	=	$8\lvert L\rvert$ *elements of* $\mathbb{F}_q$
Randomness	=	$8\lambda + 5\log q$ *bits*
Soundness Error	=	$\max(\lceil m/\lambda\rceil, \lceil n/k\lambda\rceil) \cdot 2^{-\lambda} + \lambda q^{-1}$
Prover Time	=	$O(\lvert L\rvert \log(m+n) + \sum_{i=1}^{3} \lVert A_i\rVert + \lVert\mathbf{b}\rVert) + 35 \cdot \mathrm{FFT}(\mathbb{F}_q, \lvert L\rvert)$
Verifier Time	=	$O(\sum_{i=1}^{3} \lVert A_i\rVert + \lVert\mathbf{b}\rVert + n + m)$
Max Rates	=	$\left(\frac{\max(m/k, n/k, 3\lambda) + 2\beta}{\lvert L\rvert}, \frac{\max(2m/k, 2n/k, 3\lambda) + 2\beta}{\lvert L\rvert}\right)$

$\mathsf{P}_{\mathsf{R1CS}}(\mathbb{F}_q, m, n, A_1, A_2, A_3, \mathbf{b}, \mathbf{w})$		$\mathsf{V}_{\mathsf{R1CS}}(\mathbb{F}_q, m, n, A_1, A_2, A_3, \mathbf{b})$
$u := \varphi(\mathbf{1}_k)$		$u := \varphi(\mathbf{1}_k)$
$\widetilde{I}_m \leftarrow (\Phi(\mathbf{e}_j)^\top)_{j=1}^m$		$\widetilde{I}_m \leftarrow (\Phi(\mathbf{e}_j)^\top)_{j=1}^m$
$\widetilde{A}_i \leftarrow (\Phi(\mathbf{a}_{i,j})^\top)_{j=1}^m$		$\widetilde{A}_i \leftarrow (\Phi(\mathbf{a}_{i,j})^\top)_{j=1}^m$
$\widetilde{\mathbf{b}} \leftarrow \Phi(\mathbf{b}),\ \widetilde{\mathbf{w}} \leftarrow \Phi(\mathbf{w})$		$\widetilde{\mathbf{b}} \leftarrow \Phi(\mathbf{b})$
$\widetilde{\mathbf{x}}_i \leftarrow \Phi(A_i\mathbf{w}),\ \mathbf{t} \leftarrow \widetilde{\mathbf{x}}_1 * \widetilde{\mathbf{x}}_2$		
$f_{\widetilde{\mathbf{w}}} \xleftarrow{\$} \{f \in \mathsf{RS}_{L,\rho_2} : \widehat{f}_{\mid H_2} = \widetilde{\mathbf{w}}\}$		
$f_{\widetilde{\mathbf{x}}_i} \xleftarrow{\$} \{f \in \mathsf{RS}_{L,\rho_1} : \widehat{f}_{\mid H_1} = \widetilde{\mathbf{x}}_i\}$		
$f_{\mathbf{t}} \xleftarrow{\$} \{f \in \mathsf{RS}_{L,\rho_1} : \widehat{f}_{\mid H_1} = \mathbf{t}\}$		
$g \xleftarrow{\$} \mathsf{BMask}(L, \rho_3, H_1', H_2', H_3', \varphi, \psi)$	$\xrightarrow{f_{\widetilde{\mathbf{w}}}, f_{\widetilde{\mathbf{x}}_i}, f_{\mathbf{t}}, g}$	
		$(\alpha_i)_{i=1}^4 \xleftarrow{\$} \mathbb{F}_{2^\lambda}$
	$\xleftarrow{\alpha_i, \gamma_i, \delta}$	$(\gamma_i)_{i=1}^3 \xleftarrow{\$} \mathbb{F}_{2^\lambda},\ \delta \xleftarrow{\$} \mathbb{F}_{2^\lambda}$
$M_1 \leftarrow ((R_{\alpha_i})_{i=1}^4, I_\lambda)$		Compute M_1
$M_2 \leftarrow (\sum_{j=1}^3 R_{\gamma_j}\widetilde{A}_j, (-R_{\gamma_i}\widetilde{I}_m)_{i=1}^3, I_\lambda)$		Compute M_2
$M_3 \leftarrow (R_\delta, -uR_\delta, I_\lambda)$		Compute M_3
$\mathbf{v}_1 \leftarrow \sum_{i=1}^3 R_{\alpha_i}\widetilde{\mathbf{x}}_i + R_{\alpha_4}\widetilde{\mathbf{w}} + \widehat{g}_{\mid H_1'}$		
$\mathbf{v}_2 \leftarrow \sum_{i=1}^3 R_{\gamma_i}(\widetilde{A}_i\widetilde{\mathbf{w}} - \widetilde{I}_m\widetilde{\mathbf{x}}_i) + \widehat{g}_{\mid H_2'}$		
$\mathbf{v}_3 \leftarrow R_\delta\mathbf{t} - uR_\delta(\widetilde{\mathbf{x}}_3 + \widetilde{\mathbf{b}}) + \widehat{g}_{\mid H_3'}$		
$\mathbf{v}_0 \leftarrow \mathbf{v}_3 + u\mathbf{v}_1$	$\xrightarrow{\mathbf{v}_0, \mathbf{v}_2}$	
		If $\mathbf{v}_2 \notin (\operatorname{Ker} S \circ \psi)^{H_0'}$
		Return $\perp$
		$\mathbf{v}_1' \leftarrow \Phi(\Psi(\mathbf{v}_0))$
		$\mathbf{v}_3' \leftarrow \mathbf{v}_0 - u\mathbf{v}_1$
Run: $\mathsf{P}_{\mathsf{Lin}_5}(M_1, \mathbf{v}_1, (f_{\widetilde{\mathbf{x}}_i})_{i=1}^3, f_{\widetilde{\mathbf{w}}}, g)$		Run: $\mathsf{V}_{\mathsf{Lin}_5}^{(f_{\widetilde{\mathbf{x}}_i})_{i=1}^3, f_{\widetilde{\mathbf{w}}}, g}(M_1, \mathbf{v}_1')$
$\mathsf{P}_{\mathsf{Lin}_5}(M_2, \mathbf{v}_2, f_{\widetilde{\mathbf{w}}}, (f_{\widetilde{\mathbf{x}}_i})_{i=1}^3, g)$		$\mathsf{V}_{\mathsf{Lin}_5}^{f_{\widetilde{\mathbf{w}}}, (f_{\widetilde{\mathbf{x}}_i})_{i=1}^3, g}(M_2, \mathbf{v}_2)$
$\mathsf{P}_{\mathsf{Lin}_3}(M_3, uR_\delta\widetilde{\mathbf{b}} + \mathbf{v}_3, f_{\mathbf{t}}, f_{\widetilde{\mathbf{x}}_3}, g)$		$\mathsf{V}_{\mathsf{Lin}_3}^{f_{\mathbf{t}}, f_{\widetilde{\mathbf{x}}_3}, g}(M_3, uR_\delta\widetilde{\mathbf{b}} + \mathbf{v}_3')$
$\mathsf{P}_{\mathsf{Row}}(\mathbb{F}_q, L, H_1, \rho_1, f_{\widetilde{\mathbf{x}}_1}, f_{\widetilde{\mathbf{x}}_2}, f_{\mathbf{t}})$		$\mathsf{V}_{\mathsf{Row}}^{f_{\widetilde{\mathbf{x}}_1}, f_{\widetilde{\mathbf{x}}_2}, f_{\mathbf{t}}}(\mathbb{F}_q, L, H_1, \rho_1)$

Fig. 2. RS-encoded IOP for R1CS. We fix a linear order on H_0, H_1, H_2 and assume $\widetilde{A}_i \in \mathbb{F}_q^{H_0 \times H_2}$, $\widetilde{I}_m \in \mathbb{F}_q^{H_0 \times H_1}$. Note the first two steps can be precomputed knowing the input size, and that $\mathbf{v}_0, \mathbf{v}_2$ are sent directly, i.e. without providing oracle access

Observe this means can take $|L| \cdot \rho \approx \max(2m/k, 2n/k, 3\lambda) + 2\beta$ for a fixed rate $\rho \approx 1/8$.

5 Comparisons

In this section we compare our construction with [AHIV17, BCR$^+$19, GSV21, BFH$^+$20] when proving satisfiability of an R1CS over $\mathbb{F}_2$. In all cases we assume [BCS16] is used to compile IOP into NIZK. Our focus will be on the proof size, which we compute through a parameter optimiser, available at [Git21], based on [lib20], the open source implementation of Aurora and R1CS-Ligero. We also consider prover efficiency, which we only estimate theoretically. Regarding verifier time instead we do not expect significant improvements or overhead, as asymptotic costs are the same with roughly the same constants.

Aurora - proof size: Compiling Aurora [BCR$^+$19] to a NIZK, proof size is dominated by the replies to oracle queries. Calling $|L|$ the block length of the Reed Solomon code in use, each of these replies requires $O(\log^2 |L|)$ hash values. As we use Reed Solomon codewords that encode vectors k times smaller w.r.t. Aurora with naïve embedding, the block length in our work is roughly k times smaller. We therefore estimate the proof size to be reduced by a term $O(\log k \log |L|)$. Concrete proof sizes are shown in Fig. 3 where results on the left are obtained using proven soundness bounds, while on the right optimistic (but not proven) bounds are used, see the full version for more details. The improvement factor for 2^{20} constraints with a $(48, 192)_2$-RMFE and 128 security bits amounts in the first case to 1.65, in the second case to 1.31.
Aurora - prover time: Using again the fact that the block length is reduced by a factor of k with a $(k, e)_2$-RMFE observe that

– In the RS-encoded IOP, the cost is dominated by the $18 \cdot \mathrm{FFT}(\mathbb{F}_q, |L|)$. In our case we perform 35 fast Fourier transforms over a set k times smaller, leading to an improvement factor of $18k/35$.
– In the low degree test, prover complexity is upper bounded by $6|L|$ arithmetic operations [BBHR18a]. Hence our construction improves by a factor k.
– In the BCS transform, computing the Merkle tree from an oracle of size $|L|$ requires $2|L| - 1$ hashes. Using column hashing our construction requires the same amount of trees as in plain Aurora. Moreover, calling f_i FRI's i-th oracle, the length of f_i is $|L| \cdot 2^{-i\eta}$ for a constant η, i.e. it scales linearly in $|L|$. Therefore our protocol requires k times less hash function evaluations.

In conclusion, we estimate that deploying a $(48, 192)_2$-RMFE leads to a $18k/35 \approx 24.7\times$ speed up asymptotically.

Ligero and BooLigero - proof size: Applying our construction to R1CS-Ligero [BCR$^+$19], whose proof size is $\Theta(\sqrt{n})$, over a field $\mathbb{F}_{2^{160}}$ we can obtain shorter proof by a factor $\sqrt{k} \approx 6.9$ as we would invoke every sub-protocol on input k times shorter. However in [AHIV17] an optimisation through interactive

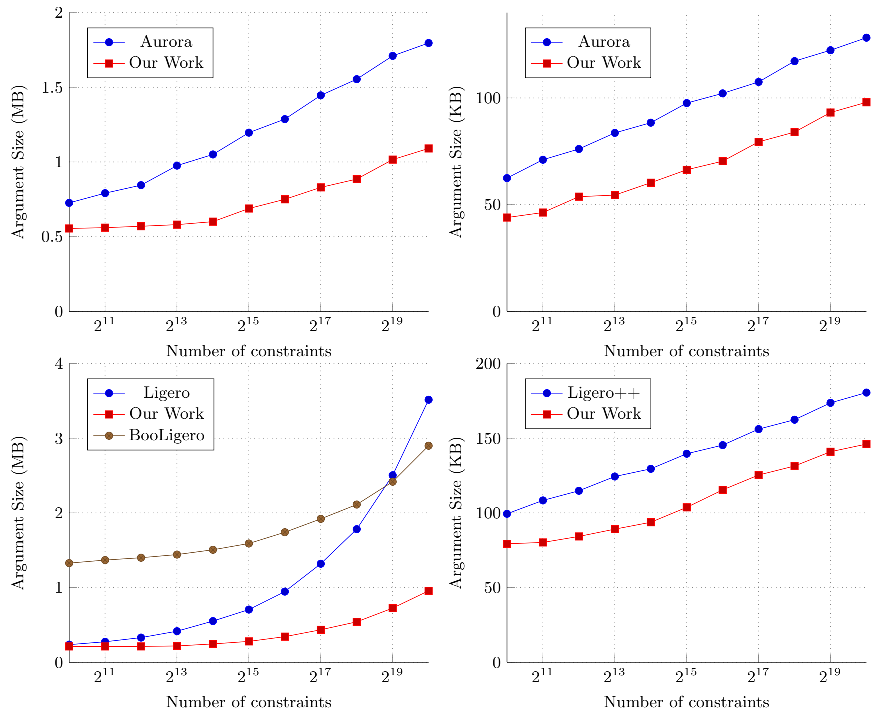

Fig. 3. Argument Size w.r.t. the number of constraints for 128 security bit for: Aurora with proven soundness bounds (up, left) and with optimistic bounds (up, right), Ligero/BooLigero *with interactive repetitions and smaller fields* (down, left) and Ligero++ (down, right). Our work uses a $(48, 192)_2$-RMFE in the first two cases, and a $(48, 160)_2$-RMFE for the others.

repetitions working over smaller fields is presented. As this version is harder to analyse asymptotically, we estimate its cost comparing it with BooLigero and our construction using a $(48, 160)_2$-RMFE (Fig. 3, down left).

Ligero and BooLigero - prover time: For simplicity we only compare our construction to Ligero without repetitions, as in this case operations are performed over the same extension of $\mathbb{F}_2$, for a R1CS over $\mathbb{F}_2$ with n variables and n constraints. Recall that $|L| = \Theta(\sqrt{n})$ and each vector is divided in m blocks of length ℓ, both growing asymptotically as $\sqrt{n}$. As in Aurora we split the prover time in three terms:

- In the IOP, costs are dominated asymptotically by $21m \cdot \mathrm{FFT}(\mathbb{F}_q, |L|)$. In our cases we would need $31m'$ fast Fourier transform but with $m' \sim m/\sqrt{k}$ and over a set $\sqrt{k}$ times smaller, leading to an improvement factor of $21k/31$
- As Ligero performs a direct low degree test no extra computation is performed for testing proximity

– In the BCS transform, using column hashing only one tree with $2|L|-1$ nodes has to be computed. Hence in our construction this step is performed $\sqrt{k}$ times faster.

In conclusion we expect an improvement factor between $6.9 - 32.5$ with a $(48, 160)_2$-RMFE. We leave prover time comparison with the more efficient version of Ligero that allows repetitions as future work.

Ligero++: As [BFH+20] combines Ligero with an inner product argument, which can be realised adapting Aurora's sumcheck to achieve poly-logarithmic argument size, we expect a prover time reduction comparable to those in plain Ligero and Aurora. The same applies to the proof size that, for completeness, we also estimate through our parameter optimiser, Fig. 3, achieving a median improvement factor of $1.26\times$.

References

ACE+21. Mark Abspoel, Ronald Cramer, Daniel Escudero, Ivan Damgård, and Chaoping Xing. Improved single-round secure multiplication using regenerating codes. *IACR Cryptol. ePrint Arch.*, 2021:253, 2021.

AHIV17. Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2087–2104. ACM Press, October / November 2017.

BBB+18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BBHR18a. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018.

BBHR18b. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. `https://eprint.iacr.org/2018/046`.

BCC+16. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 327–357. Springer, Heidelberg, May 2016.

BCR+19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.

BCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.

BFH$^+$20. Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2025–2038. ACM Press, November 2020.

BFS20. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 677–706. Springer, Heidelberg, May 2020.

BGKS20. Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: Sampling outside the box improves soundness. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 5:1–5:32. LIPIcs, January 2020.

BMN18. Alexander R. Block, Hemanta K. Maji, and Hai H. Nguyen. Secure computation with constant communication overhead using multiplication embeddings. In Debrup Chakraborty and Tetsu Iwata, editors, *INDOCRYPT 2018*, volume 11356 of *LNCS*, pages 375–398. Springer, Heidelberg, December 2018.

CCXY18. Ignacio Cascudo, Ronald Cramer, Chaoping Xing, and Chen Yuan. Amortized complexity of information-theoretically secure MPC revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 395–426. Springer, Heidelberg, August 2018.

CG20. Ignacio Cascudo and Jaron Skovsted Gundersen. A secret-sharing based MPC protocol for boolean circuits with good amortized complexity. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part II*, volume 12551 of *LNCS*, pages 652–682. Springer, Heidelberg, November 2020.

CG21. Ignacio Cascudo and Emanuele Giunta. On interactive oracle proofs for boolean r1cs statements. Cryptology ePrint Archive, Report 2021/694, 2021. `https://ia.cr/2021/694`.

COS20. Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Heidelberg, May 2020.

DGOT21. Cyprien Delpech, Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient zero-knowledge mpcith-based arguments. To appear in Proceedings of ACM CCS 21. Available at Cryptology ePrint Archive, Report 2021/215, 2021. `https://eprint.iacr.org/2021/215`.

DLN19. Ivan Damgård, Kasper Green Larsen, and Jesper Buus Nielsen. Communication lower bounds for statistically secure MPC, with or without preprocessing. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 61–84. Springer, Heidelberg, August 2019.

Git21. zk-SNARKs argument size comparison. `https://github.com/emanuelegiunta/snarks_comparison`, 2021.

GM10. Shuhong Gao and Todd Mateer. Additive fast fourier transforms over finite fields. *IEEE Transactions on Information Theory*, 56(12):6265–6272, 2010.

GSV21. Yaron Gvili, Sarah Scheffler, and Mayank Varia. Booligero: Improved sublinear zero knowledge proofs for boolean circuits. To appear in *Proceedings of Financial Crypto 2021*. Available at Cryptology ePrint Archive, https://eprint.iacr.org/2021/121.pdf, 2021.

IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

lib20. Libiop. `https://github.com/scipr-lab/libiop`, 2020.

Mer90. Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 218–238. Springer, Heidelberg, August 1990.

Mic94. Silvio Micali. CS proofs (extended abstracts). In *35th FOCS*, pages 436–453. IEEE Computer Society Press, November 1994.

PS21. Antigoni Polychroniadou and Yifan Song. Constant-overhead unconditionally secure multiparty computation over binary fields. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 812–841. Springer, Heidelberg, October 2021.

RRR16. Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 49–62. ACM Press, June 2016.

Set20. Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 704–737. Springer, Heidelberg, August 2020.

Wee05. Hoeteck Wee. On round-efficient argument systems. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 140–152. Springer, Heidelberg, July 2005.

WTS$^+$18. Riad S Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 926–943. IEEE, 2018.