Ternary Constraint System* † ‡ § ¶

Chance Hudson

2025-11-14

Rank 1 constraint systems (R1CS) drove innovation in programmable cryptography by cleanly expressing turing complete systems as mathematical structures. However, despite their linear algebra structure R1CS are not well suited for use with lattice arguments of knowledge. This note proposes a minimal constraint system for use with MLWE and MSIS lattice constructions.

Given circuit constants α, β and witness values $x, y \in \mathbb{Z}_q$ with prime q the following relations are valid constraints:

$$0 = \alpha x^2 - \beta y$$

$$0 = \alpha x - \beta y$$

$$0 = \alpha x + \beta y$$

Notably there is no general multiplication constraint. In non-binary fields the relation xy may be expressed as $2^{-1}((x+y)^2 - x^2 - y^2)$. Admitting only square multiplication makes it easier to reason about witness elements nested deeply in constraint systems. Using this technique we describe a SNARK based on standard lattice assumptions with groth16 style $\mathcal{O}(1)$ size. Naively $\mathcal{O}(3N)$ proving and verification. Three instantiations are described, one optimized for performance, one with conjectured statistical hiding and one with conjectured statistical binding. Finally, we differentiate statistical vs information theoretic security.

*Draft revision dba27cae

†this is a piece of shit only read it if you want to get more stupid

‡special thanks to srinath setty for blindfold/spartan2/spartan, blindfold description was instrumental

§special thanks to BDLOP authors for work on statistical hardness

¶special thanks to vadim lyubashevsky for a 60 page survey of lattice cryptography as of june 2025

1 Preliminaries

Some lattice arguments of knowledge have expressed constraints as polynomials [2]. This type of expression is mathematically convenient as embedding polynomials in matrices is standard practice. Unfortunately expressing program constraints as relations between polynomials is difficult and unintuitive. Further, typical lattice polynomial rings do not form fields. In this work we express constraints over scalars. Scalars are lifted into polynomials automatically during arithmetization. We can avoid committing polynomials with trailing zeroes by shaping the ring we operate within to optimize such that $m \mod N \approx 0$. In a system we refer to the number of constraints as $m \in \mathbb{Z}^+$.

This work is designed to be used with lattice arguments over cyclotomic polynomial rings with degree N such that $\log(N) \in \mathbb{Z}^+$. Cyclotomic polynomials are of the format $X^N - 1$ or $X^N + 1$ (cyclotomic polynomials must be divisors of $X^N - 1$). We define $R = \mathbb{Z}[X]/(X^N + 1)$ the negacyclic polynomial ring and $R_q = \mathbb{Z}_q[X](X^N + 1)$ with q prime as a ring over a finite field. Given $\mathbf{x} \in R_q$, $x_i \in \mathbb{Z}_q$ is a coefficient. Rings constructed over a fully splitting field have multiplication complexity $\mathcal{O}(n \log(n))$ using the number theoretic transform.

A field fully splits a polynomial ring if the ring modulus factors to linear components in the field. For example, the ring $\mathbb{Z}[X]/(X^4 + 1)$ in $\mathbb{Z}_5$ splits to $(x^2 + 2)(x^2 + 3)$, but cannot split further. However in $\mathbb{Z}_{17}$ the ring is split to $(x^2 - 4)(x^2 + 4)$ and then fully split to $(x - 2)(x + 2)(x - 8)(x + 8)$.

Fields with cardinality q such that $q-1$ is a power residue tend to split more (e.g. $\exists x \in \mathbb{Z}_q$ such that $x^i = q - 1$ with $i \geq 2$).

Of important note is examining polynomial rings as multiplicative combinations of residue spaces as in [3]. This technique permits a computational argument of invertibility for challenge elements in fully split rings. TODO: expand to section

Elements being lifted from a structure of lower cardinality to a structure of higher cardinality (e.g. $\mathbb{Z}_q \to R_q$) are indicated with a hat ($\hat{x}$). All division between integers is floored: $x, y \in \mathbb{Z}, x/y = \lfloor x/y \rfloor$. Scalar and polynomial vectors are indicated with arrows: (respectively) $\overrightarrow{x}$, $\overrightarrow{\mathbf{x}}$. Vectors are column vectors, matrices are uppercase.

Recall the set of whole numbers: $\mathbb{W} = \mathbb{Z}^+ \cup \{0\}$.

2 Displacement

Modern lattice constructions encode data symmetrically around the zero element, typically using gaussian and uniform distributions. Probability distributions centered at zero are mapped to displacements of field elements. We refer to displacement as a linear translation [is it linear?] of finite field elements to an integer range centered around zero. We take the norm of an element as the absolute value of displacement.

We can express the displacement function as in physics and geometry:

$$\Delta_q(x) : \mathbb{Z}_q \to \mathbb{Z}$$

$$\Delta_q(x) = \begin{cases} \hat{x} & \hat{x} < q/2 \\ \hat{x} - q & \hat{x} \geq q/2 \end{cases} \quad \text{where } \hat{x} \equiv x \pmod{q},\ 0 \leq \hat{x} < q$$

We have $\Delta_{101}(1) = 1$, $\Delta_{101}(100) = -1$, $\Delta_{101}(0) = 0$, $\Delta 101(55) = -46$. Observe that this expression of position minimizes the absolute value of representation and yields a totally ordered equality relation instead of a congruence. Note that the displacement function forms a linearly shifted homomorphism between a finite field and the integers (TODO: double check this):

$$\forall x, y, z \in \mathbb{Z}_q, \Delta(x) * \Delta(y) + \Delta(z) = \Delta(x * y + z)$$

Given a displacement function we can write a simple norm/distance statement, see appendex A.1 for a more verbose expression:

$$\forall x \in \mathbb{Z}_q, |x| = |\Delta(x)| \in \mathbb{Z} \cap [0, q/2 + 1]$$

The displacement function is used to map a discrete probability distribution centered at zero into a finite field. The norm statement is used to put symmetrical bounds on solution spaces in the field.

3 Polynomial and vector norms

In MSIS and MLWE instances we bound the norm of polynomials and vectors to ensure hardness. We specify the ℓ_1 norm of a polynomial with coefficients in $\mathbb{Z}_q$:

$$\forall \mathbf{x} \in R_q, |\mathbf{x}| = \sum_{i=1}^{N} |x_i| \in \mathbb{W}$$

And the norm of a vector of polynomials:

$$\forall k \in \mathbb{W}, \forall \overrightarrow{\mathbf{x}} \in R_q^k, |\overrightarrow{\mathbf{x}}| = \sum_{i=1}^{k} |\mathbf{x}_i| \in \mathbb{W}$$

Finally, the norm of a vector of scalar field elements:

$$\forall k \in \mathbb{W}, \forall \overrightarrow{x} \in \mathbb{Z}_q^k, |\overrightarrow{x}| = \sum_{i=1}^{k} |x_i| \in \mathbb{W}$$

Each of these can be characterized as distance from the zero element in the corresponding set expressed as a scalar integer.

4 Hardness assumptions

Here we recall details of relevant hardness assumptions.

4.1 MSIS

Given public parameter $\mathbf{A} \in R_q^{n \times m}$ find $\overrightarrow{\mathbf{x}} \in R_q^m$ such that $\mathbf{A}\overrightarrow{\mathbf{x}} = \overrightarrow{\mathbf{0}}$, $\mathbf{x} \neq \mathbf{0}$ and $|\overrightarrow{\mathbf{x}}| < \beta$ for $q > \beta \geq \sqrt{n \log q}$ and $m \geq n \log q$. We described the ℓ_1 norm above, here we describe the ℓ_2 and ℓ_∞ norms:

$$||\mathbf{x}||_2 = \left(\sum_{i=1}^{m} |x_i|^2 \right)^{1/2} \in \mathbb{Z}$$

$$||\mathbf{x}||_\infty = \max(x_i) \in \mathbb{Z}$$

5 Main contribution

A constraint system is used to express a program as a system of equations over a vector of variables. At proving time the program is evaluated and the value of each intermediate variable is stored to form the witness vector $\overrightarrow{w}^m \in \mathbb{Z}_q^m$. A constraint system has 3 vectors of tuple constraints of the form (α, β, i, j) such that $\alpha, \beta \in \mathbb{Z}_q$ and $i, j \in [0, m)$. Each vector corresponds to the constraints for sq, sub, add functions respectively:

$$sq(\alpha, \beta, i, j) = \alpha w_i^2 + \beta w_j$$

$$sub(\alpha, \beta, i, j) = \alpha w_i - \beta w_j$$

$$add(\alpha, \beta, i, j) = \alpha w_i + \beta w_j$$

The final relation checks that all constraint inputs to the above functions evaluate to zero.

TODO: actually write this section

5.1 Succinct argument of knowledge

Limiting the constraint system to square multiplication prevents convolution of inputs that are linear combinations. Using this property we can construct an argument of knowledge that is linear in the number of inputs and outputs only, similar to groth16. We do this by committing to the inputs, sampling a random mask for each input variable, and computing a final error term for each output variable. The verifier receives the masked inputs and evaluates the constraint system checking that the prover computed error values match the sampled error values. Finally, the prover makes a linearly homomorphic argument that the masks to the input variables match the mask agreed upon at the start.

5.2 Non-convolution

TODO: b term is not a valid constraint, separate

Let's consider an example constraint system. Say that we have inputs x and y and the following constraints. Here we express constraints as assignments by re-arranging the core relations:

$$a = x + y$$
$$b = a^2 + y$$
$$c = b + a$$

We'll say that c is a public variable we want to argue computation of. Coefficients are excluded for brevity without loss of generalization:

$$b = x^2 + 2xy + y^2 + y$$
$$c = x^2 + 2xy + y^2 + x + 2y$$

The prover will make an argument of knowledge given secret challenges $x', y' \xleftarrow{\$} \mathbb{Z}_q$ and setting $c_x \leftarrow x + x'$ and $c_y \leftarrow y + y'$. The prover then evaluates the c constraint using the masked values yielding:

$$c = (x + x')^2 + 2(x + x')(y + y') + (y + y')^2 + (x + x') + 2(y + y')$$
$$c = x^2 + 2xx' + x'^2 + 2xy + 4x'y + 2x'y' + y^2 + 2yy' + 2y'^2 + x + x' + 2y + y'$$

TODO: double check this, automated tests for inlined polynomials

Given this relation we can write the following:

$$e \leftarrow 2xx' + x'^2 + 4x'y + 2x'y' + 2yy' + 2y'^2 + x' + y'$$
$$c = x^2 + 2xy + y^2 + x + 2y + e$$

This final relation yields the expected output value of the secret inputs, and a linear error term. This approach nests indefinitely by accumulating the error term and argument term at each step of evaluation separately, for each input, then revealing the convoluted value as a single scalar.

TODO: test circuits with lots of input/output interaction, poseidonT16 poseidont256, figure out hashing in a 30 bit field, configure and argue statistical security

Argument:

Rewrite the above c constraint as $c = w + e$ where w is the un-mangled intermediate value. The only possible subsequent constraints are $0 = c + z$, $0 = c - z$, or $0 = c^2$. The first two yield trivial linear combinations, and the final one yields $0 = w^2 + we + e$, which combines two secret linear terms into one $0 = w^2 + e'$. The error term is a high dimension linear combination, but is information theoretically masked: verifier knows only two terms.

Also if it's not information theoretically masked we can toss it in a commitment and linear ZK arg while keeping $\mathcal{O}(1)$ size.

5.3 Lettuce

In this section we present the Lettuce SNARK, a three move IOP supporting fiat-shamir transformation with minimal entropy use. Assume we have a linearly homomorphic vector commitment scheme. This scheme must support ZK arguments of linear relation between committed values.

A prover wants to argue knowledge of a system $\mathcal{C}$ with inputs x, y and output z. The system may have any number of intermediate constraints and witness variables.

The prover opens a commitment to the vector x, y, z and sends it to the verifier. The verifier samples challenge elements $c_x, c_y \xleftarrow{\$} comm(x, y, z)$ and sends it to the prover. The prover and verifier both evaluate the system $\mathcal{C}$ using c_x and c_y as inputs, yielding output $z_m \leftarrow z + e$. e and z are information theoretically invisible to verifier.

The prover computes $x_m = c_x - x$ and $y_m = c_y - y$, the input mask values chosen by, and invisible to, verifier. Using the non-convolution property described above the prover computes $e = z + z_m$.

The prover makes a single, three term, linear ZK argument for each input and output:

$$c_x^\pi \leftarrow c_x = x_m + x$$

Order of events: x is selected, c_x is sampled from $comm(x)$, x_m was linearly homomorphically argued from masked evaluation of $\mathcal{C}$:

$$c_y^\pi \leftarrow c_y = y_m + y$$

The prover then argues output relations:

$$e^\pi \leftarrow e = z_m + e$$

The prover sends c_x^π, c_y^π, e^π, z, e to the verifier. The verifier checks c_x^π, c_y^π using the original commitments from the prover. Verifier then checks e^π using locally computed z_m. Finally the verifier computes the public output $z = z_m - e$.

The above system is not asymptotically bounded by number of intermediate steps. Therefore a program of infinite length can be compressed into an argument of finite size in a plausibly quantum resistant way.

6 Complexity analysis

The Lettuce SNARK with witness size m and constraints n has prover complexity $\mathcal{O}(n)$, verifier complexity $\mathcal{O}(n)$ and communication complexity $\mathcal{O}(1)$.

The shrewd reader might notice that the verifier in the previous section can begin verifying the argument before receiving it. The verifier samples challenge elements that become the masked input values, which are then evaluated through the circuit. The verifier can compute the masked output while the prover computes arguments of opening and error term.

Verification reduces to evaluating a multivariate polynomial with degree variable based on the size of the constraint system. Either the prover or the verifier may argue evaluation of the argument using a sumcheck style approach to achieve verification complexity $\mathcal{O}(m+log(n))$. TODO: double check this bound

7 Security analysis

The presented IOP is relatively simple. We'll argue security by showing that the argument is secure for the smallest circuit, then generalizing to all possible circuits.

Examining the base case of one input, one output we have input x and output z both in $\mathbb{Z}_q$ with constraint system $\mathcal{C}$. x is committed and a single challenge is produced c_x yielding input mask $x_m \leftarrow c_x - x$. The circuit is evaluated using the challenge value yielding the masked output z_m. The prover is bound to z and z_m and must provide an e fulfilling the relation $z = z_m - e$. This relation has only one possible solution.

Let's assume the circuit $\mathcal{C}$ takes input x and outputs $z = x^2$. The prover wants to argue $x' \neq x$ yields z_m. The final relation is as follows:

$$z = x^2 + 2xx_m + x_m^2 - e$$

The only term not bound by commitments or challenges is e, which has only one solution. Provided an information theoretically secure commitment scheme the Lettuce argument forms an information theoretically secure SNARK.

NOTE: is it information theoretically secure down to the commitment scheme? e.g. even breaking a system with a single additive constraint is impossible? FUTURENOTE: should analyze complexity of higher dimension systems, single additive constraint is easy to mask. Though it does seem... impossible to derive a relation in the system? Need to test with many convolutional inputs and outputs.

7.1 Small fields

Lattice commitment schemes often operate in fields with $q \approx 2^{30}$. The described system relies on a challenge element, however the hiding mechanism is information theoretically secure and the binding mechanism is statistically secure. Therefore a single interation in a small field should be sufficient for soundness. CITATION NEEDDEDDDDDDDEDEDEEDEDDEDEEDDEEDEDDEDE

7.2 Low entropy inputs

Circuit inputs are often low entropy, binary inputs are commonly used for decomposed operations. In this scheme the verifier samples a challenge, and the prover retroactively argues that the challenge is a sum of the input and a mask.

The mask being determined by the challenge without being revealed to the verifier. All arguments of linear relation happen behind homomorphic commitments so low entropy inputs should not impact soundness.

8 Discussion of principal relation

By masking inputs and forcing the verifier to evaluate the circuit itself we can exploit irreversibility in the programs being arithmetized. e.g. the verifier is *technically* computing poseidon of a masked input verifying a ZK argument of merkle inclusion.

Further optimizations are possible using circuit structure to perform decryption in circuit to allow unmasked inputs. e.g. to input 1GB of data as a witness I encrypt a single input that is a chacha private key. I pass the remaining 1GB encrypted as public circuit inputs. Then *in circuit* I decrypt, which logically encrypts for the verifier, because the encryption key is masked.

This strategy requires a commitment only for the private key input. The 1GB is loaded as free intermediate variables. Argument size would be $\mathcal{O}(privatekey)$.

9 Statistical vs information theoretic security

[1] argues the existence of statistical secure instantiations of MSIS and MLWE. We summarize the two approaches here.

9.1 Statistical MSIS

References

[1] Carsten Baum and Ivan Damgård and Vadim Lyubashevsky and Sabine Oechsner and Chris Peikert, More Efficient Commitments from Structured Lattice Assumptions, Cryptology ePrint Archive, Paper 2016/997, 2016, https://eprint.iacr.org/2016/997

[2] Ward Beullens and Gregor Seiler, LaBRADOR: Compact Proofs for R1CS from Module-SIS, Cryptology ePrint Archive, Paper 2022/1341, 2022, https://eprint.iacr.org/2022/1341

[3] Thomas Attema and Vadim Lyubashevsky and Gregor Seiler, Practical Product Proofs for Lattice Commitments, Cryptology ePrint Archive, Paper 2020/517 2020, https://eprint.iacr.org/2020/517

A Verbose expressions

A.1 Distance in a finite field

Given $x \in \mathbb{Z}_q$ we refer to $|x| : \mathbb{Z}_q \to [0, q/2+1] \cap \mathbb{Z}$ as the norm or distance from the zero element. We express this as the number of summations of either the one element, or its additive inverse (negative one); whichever is smaller. This forms a $\mathbb{Z}_q \to \mathbb{Z}$ surjection whereas displacement forms a bijection. Surjection is necessary to position elements symmetrically around zero according to a discrete probability distribution.

$$\forall x \in \mathbb{Z}_q, \exists! |x| \in \mathbb{Z} \cap [0, q/2+1], \left(x = \sum_{i=1}^{|x|} 1 \pmod q \right) \vee \left(x = \sum_{i=1}^{|x|} (q-1) \pmod q \right)$$