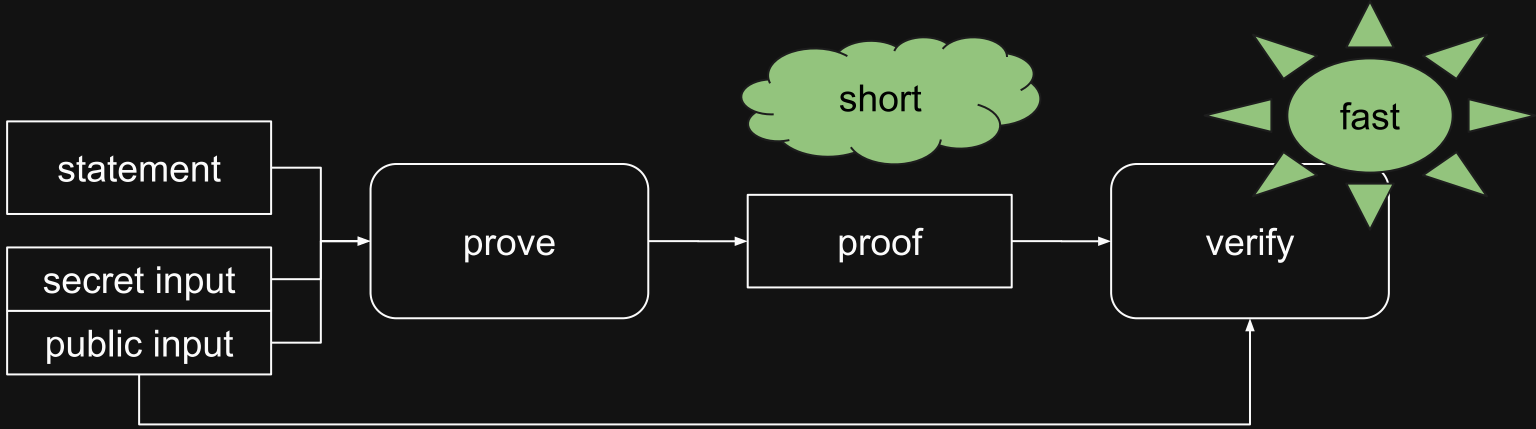
Efficient non-native SNARK recursion using bivariate polynomial testing

Ivo Kubjas

- cryptographer at Consensys
- gnark co-maintainer
- Linea co-developer

Short proof of statement



Optional: zero-knowledge

Many ways to compute a proof

- Sum-check - represent statement as a multilinear polynomial. Verifier needs to do final evaluation
- GKR - layered sum-check.
- STARKs - represent statement as constrained execution steps. Based on hash functions.

fast prover, **less assumptions**, **longer proof**, **slower verification**

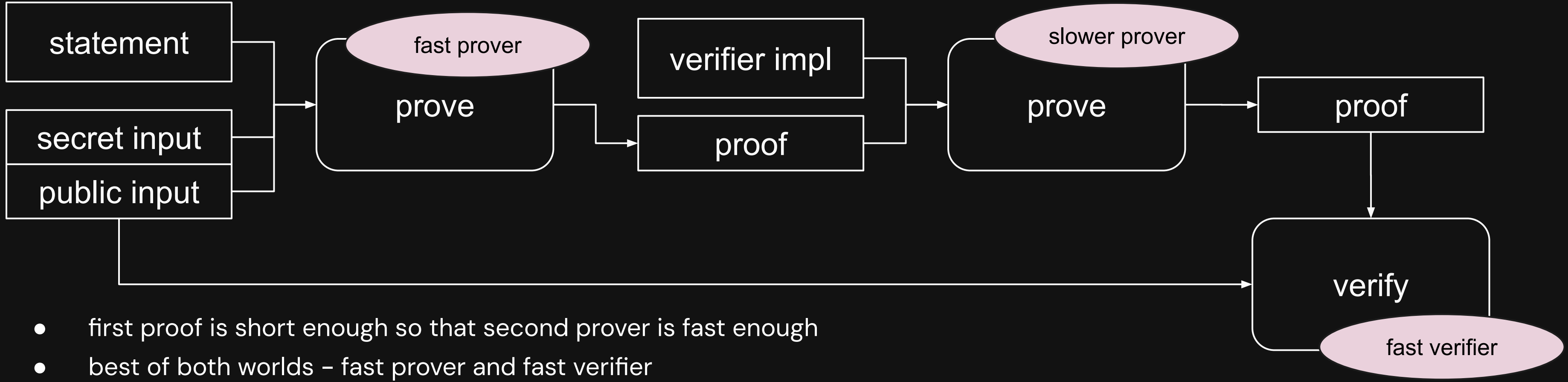
- Groth16 - based on pairings. Per-circuit (huge) keys. Constant proof and verification.
- PLONK - usually initialized using pairings. Universal (huge) keys. Constant proof and verification.

slover prover, **more assumptions**, **short proof**, **fast verification**

- etc.

Optimize for prover and verifier speed

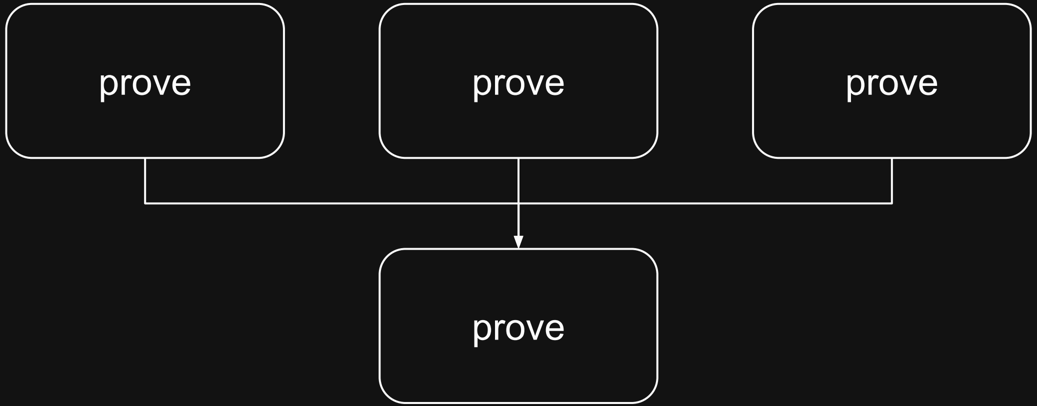
We can implement a proof verifier as a statement.



- first proof is short enough so that second prover is fast enough
- best of both worlds – fast prover and fast verifier

Proof independence

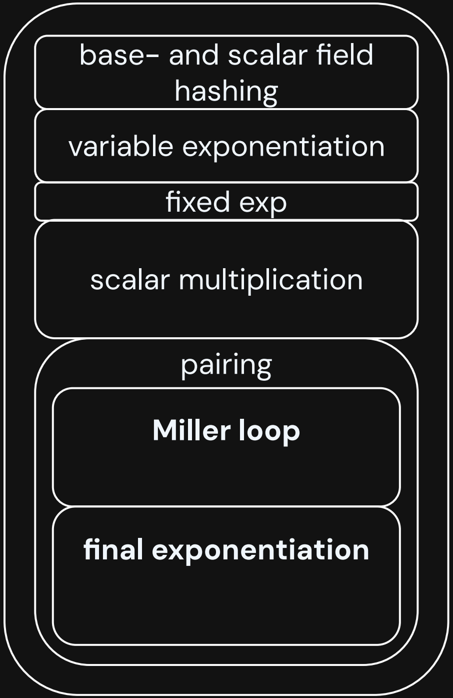
Frequently can prove independent statements in parallel



PLONK in PLONK

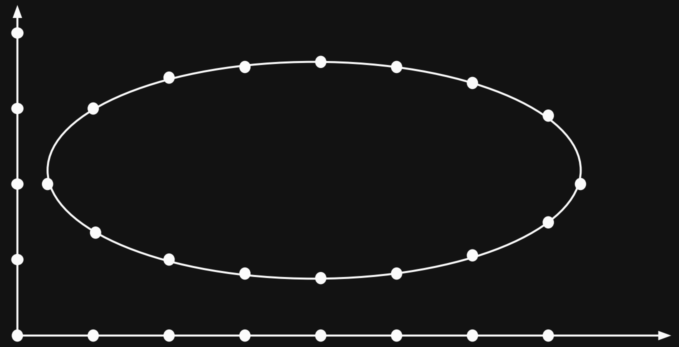
In Linea:

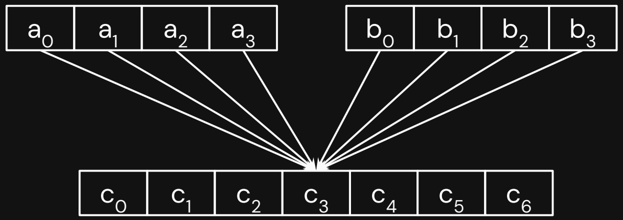
- BN254 PLONK in BLS12-377 PLONK
- BLS12-377 Vortex in BLS12-377 PLONK
- BLS12-377 PLONK in BW6-761 PLONK
- BW6-761 PLONK in BN254 PLONK



Field (mis)match

We need to compute target curve operations over $\mathbb{F}_p$ in the current curve scalar field $\mathbb{F}_q$





$c_k = \Sigma_{i+j=k} a_i * b_j$

We're lucky to have 2-chain BLS12-377/BW6-761 where there is match $\mathbb{F}_p = \mathbb{F}_q$

For other combinations we need to emulate multiprecision arithmetic:

- Non-native a as: $a = \Sigma_i \, a_i \, 2^{iB}$, where a_i are native field elements
- Representing $a(X) = \Sigma_i \, a_i \, X^i$, can verify computation $a(\tau) * b(\tau) = c(\tau) + n(\tau)p(\tau)$

Field extensions

Pairing friendly curves have a pairing function $e(P, Q) = T$, where $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, $T \in \mathbb{G}_T$

- $\mathbb{G}_1$ curve itself, $\mathbb{G}_2$ and $\mathbb{G}_T$ extensions field of the base field: $\mathbb{G}_2 \approx \mathbb{F}_p{}^m$, $\mathbb{G}_T \approx \mathbb{F}_p{}^k$

$$Q \in \mathbb{F}_p{}^m\colon Q(X) = Q_0 + Q_1 X + \dots + Q_{m-1} X^{m-1}$$

$$(Q_0, Q_1, \dots, Q_{m-1})$$

$$T \in (\mathbb{F}_p{}^3)^2\colon \quad ((T_0, T_1, T_2),(T_3, T_4, T_5))$$

$$((A_0, A_1, A_2),(A_3, A_4, A_5)) * ((B_0, B_1, B_2),(B_3, B_4, B_5)) =$$
$$((A_0, A_1, A_2)*(B_0, B_1, B_2), (A_0, A_1, A_2)*(B_3, B_4, B_5)+(A_3, A_4, A_5)*(B_0, B_1, B_2))$$

- For efficiency reasons usually are built using towers

BN254	BLS12-377	BW6-761
$\mathbb{F}p^2[u] = \mathbb{F}p/u^2+1$	$\mathbb{F}p^2[u] = \mathbb{F}p/u^2+5$	$\mathbb{F}p^3[u] = \mathbb{F}p/u^3+4$
$\mathbb{F}p^6[v] = \mathbb{F}p^2/v^3-9-u$	$\mathbb{F}p^6[v] = \mathbb{F}p^2/v^3-u$	$\mathbb{F}p^6[v] = \mathbb{F}p^3/v^2-u$
$\mathbb{F}p^{12}[w] = \mathbb{F}p^6/w^2-v$	$\mathbb{F}p^{12}[w] = \mathbb{F}p^6/w^2-v$	

feltroidprime · Follow
Last edited by feltroidprime on Nov 17, 2023
Contributed by

7 0

Faster Extension Field multiplications for Emulated Pairing Circuits

We can define $\boxed{P_{12}(x) = x^{12} - 18x^6 + 82}$ and $\boxed{P_6(x) = x^6 - 18x^3 + 82}$ as the irreducible polynomials generating the *direct* extensions $\mathbb{F}_{p^{12}} = \mathbb{F}_p[w]/P_{12}(w)$ and $\mathbb{F}_{p^6} = \mathbb{F}_p[v]/P_6(v)$.

Whereas, using the direct representation, one can write $X \in \mathbb{F}_{p^{12}}$ using $x_0, \ldots, x_{11} \in \mathbb{F}_p$:

$$X = x_0 + x_1 w + x_2 w^2 + x_3 w^3 + x_4 w^4 + x_5 w^5 + x_6 w^6 + x_7 w^7 + x_8 w^8 + x_9 w^9 + x_{10} w^{10} + x_{11} w^{11}$$

- Perform the Euclidean division of C by the irreducible polynomial P_{12} of degree 12 and obtain the quotient Q and the remainder R of degree (at most) 10 and (at most) 11 such that:

$$\boxed{A(x) * B(x) = Q(x) * P_{12}(x) + R(x)} \tag{1}$$

- Evaluate $a_i = A_i(z_i), b_i = B_i(z_i), q_i = Q_i(z_i), p_i = P_{12}(z_i), r_i = R_i(z_i)$
- Verify that $a_i * b_i = q_i * p_i + r_i$.
  According to the Schwartz-Zippel Lemma, if Q and R are incorrect, the evaluated polynomials will differ with a very high probability.

On Proving Pairings

Andrija Novakovic*[1] and Liam Eagen†[2]

[1]Geometry Research
[2]Alpen Labs, Zeta Function Technologies

April 24, 2024

BW6-761 and BLS12-377

Direct extensions $\mathbb{F}p^6[v] = \mathbb{F}p/v^6+4$ and $\mathbb{F}p^{12}[w] = \mathbb{F}p/w^{12}+5$

Need to show:

$$A(z)*B(z)=C(z) + P_6(z)*N(z)$$

$$(A_0 + A_1z + A_2z^2 + A_3z^3 + A_4z^4 + A_5z^5)*(B_0 + B_1z + B_2z^2 + B_3z^3 + B_4z^4 + B_5z^5) = \ldots$$

non-native!

Non-native multivariate evaluation

Actually, we can evaluate multivariate operations in non-native with same cost.

Instead of: $a * b = r + n * p$

We show: $\Sigma_i \Pi_j a_j^{e_{ij}} = r + n * p$

Using: $\Sigma_i \Pi_j a_j(\boldsymbol{\tau})^{e_{ij}} = r(\boldsymbol{\tau}) + n(\boldsymbol{\tau}) * p(\boldsymbol{\tau})$

Written out

$$(A_0+A_1z+A_2z^2+A_3z^3+A_4z^4+A_5z^5)*(B_0+B_1z+B_2z^2+B_3z^3+B_4z^4+B_5z^5) = (C_0+C_1z+C_2z^2+C_3z^3+C_4z^4+C_5z^5)+(N_0+N_1z+N_2z^2+N_3z^3+N_4z^4+N_5z^5)*(P_0+P_1z+P_2z^2+P_3z^3+P_4z^4+P_5z^5)$$

1. Evaluate $z, \ldots, z^5$
2. Evaluate $\pi = P(z)$ for all checks
3. Evaluate $\alpha = A(z)$ and $\beta = B(z)$ for every check
4. Check:

$$C_0+C_1z+C_2z^2+C_3z^3+C_4z^4+C_5z^5+N_0\pi+N_1z\pi+N_2z^2\pi+N_3z^3\pi+N_4z^4\pi+N_5z^5\pi-\alpha-\beta = 0$$

Reduction to 3 non-native operations

Main cost in range checking the hinted values

Benchmarks

Constraints in PLONKish (with commitment column)

Operation	Before	After	**Reduction**
Miller loop	6500708	3841000	**41%**
Miller loop fixed G2	5344302	2680076	**49%**
final exponentiation	5245872	3362746	**36%**
full pairing	11486969	6947630	**40%**
full pairing fixed G2	10440385	5888826	**44%**

Improvements

- We don't need to hint every quotient separately, but can combine
  - Both for non-native multivariate evaluations and extension multiplication
- Can apply non-native multivariate evaluations to scalar multiplications and exponentiations (fixed exponent)
- Currently implemented only for BW6-761, also for BN254, BLS12-377

$$A_1(z)*B_1(z)=C_1(z) + P_6(z)*N_1(z)$$

$$A_2(z)*B_2(z)=C_2(z) + P_6(z)*N_2(z)$$

$$A_3(z)*B_3(z)=C_3(z) + P_6(z)*N_3(z)$$

...

$$(\Sigma_i \lambda_i A_i(z)B_i(z)) = (\Sigma_i \lambda_i C_i(z)) + P_6(z)*N_\lambda(z)$$

Thank you

gnark.io
linea.build
gnark@consensys.net

TG: t.me/ivokub
Github: github.com/ivokub
X: x.com/ikubjas
ivo.kubjas@consensys.net