ia.cr/2025/933

Efficient elliptic curve arithmetic in zero-knowledge circuits

Simon Masson

ZKNox

Joint work with	Y. El Housni	(Linea)
	T. Piellard	(Linea)
	L. Eagen	(Alpen Labs)

June 11th, 2025 – *Berlin, ZK Day*

ZKNOX team

Nicolas Bacca
20^+ years experience (10^+y web3)
Security and hardware specialist
Prev. Ledger cofounder/CTO

Renaud Dubois
20^+ years experience (3^+y web3)
Cryptographer
Prev. Ledger, Thales

Simon Masson
8^+ years experience (4^+y web3)
Cryptographer
Prev. Heliax, Thales

ZKNOX team

Nicolas Bacca
20^+ years experience (10^+y web3)
Security and hardware specialist
Prev. Ledger cofounder/CTO

Renaud Dubois
20^+ years experience (3^+y web3)
Cryptographer
Prev. Ledger, Thales

Simon Masson
8^+ years experience (4^+y web3)
Cryptographer
Prev. Heliax, Thales

Expertise and innovation to every challenge on the whole security chain:

- user end (secure enclaves, hardware wallets),
- back end (TEE, HSMs),
- on-chain (smart contracts).

ZKNOX team

Nicolas Bacca
20^+ years experience (10^+y web3)
Security and hardware specialist
Prev. Ledger cofounder/CTO

Renaud Dubois
20^+ years experience (3^+y web3)
Cryptographer
Prev. Ledger, Thales

Simon Masson
8^+ years experience (4^+y web3)
Cryptographer
Prev. Heliax, Thales

Expertise and innovation to every challenge on the whole security chain:

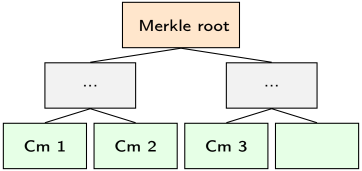
- user end (secure enclaves, hardware wallets),
- back end (TEE, HSMs),
- on-chain (smart contracts).

`https://zknox.eth.limo/`

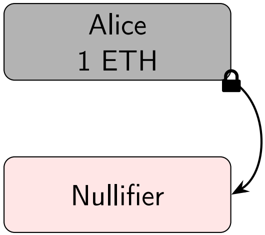
`https://github.com/zknoxhq/`

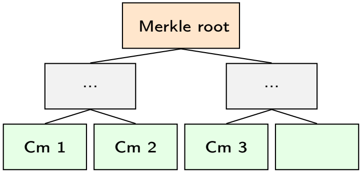
Introduction: privacy in Ethereum

Alice
1 ETH

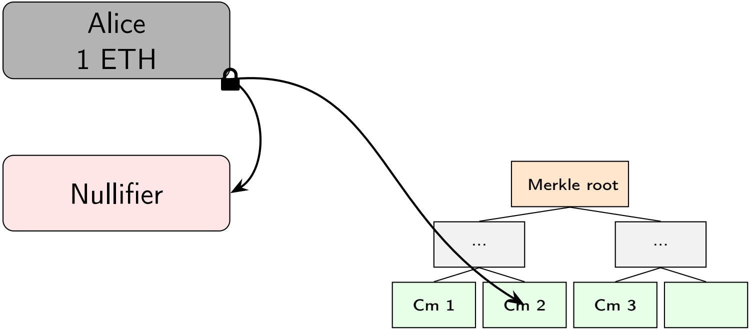


Introduction: privacy in Ethereum

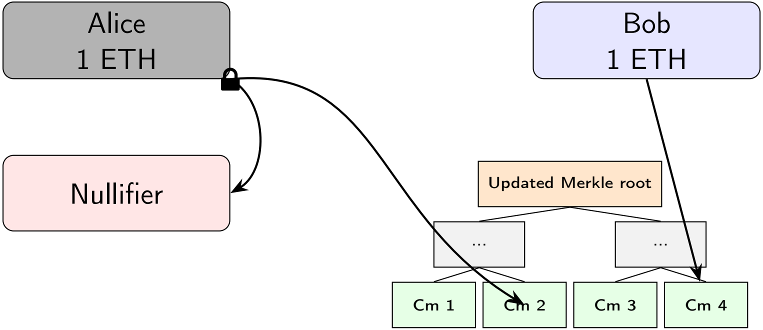




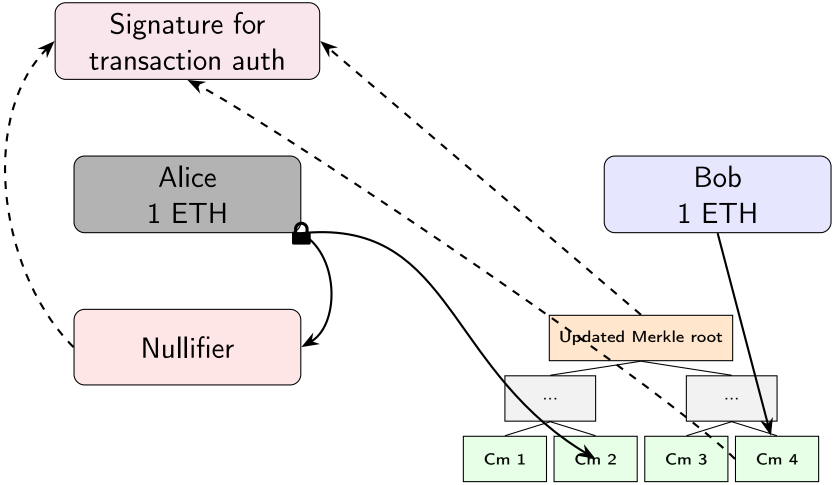
Introduction: privacy in Ethereum



Introduction: privacy in Ethereum



Introduction: privacy in Ethereum



Introduction: privacy in Ethereum

A zero-knowledge proof that:

- Binds the note owner to the created nullifier,
- Proves the ownership of the note,
- Authenticates the transaction from the owner to someone else.

Introduction: privacy in Ethereum

A zero-knowledge proof that: (in practice)

- Binds the note owner to the created nullifier,
  (hash commitment)
- Proves the ownership of the note,
  (several hashes in a Merkle tree)
- Authenticates the transaction from the owner to someone else.
  (signature verification)

Introduction: privacy in Ethereum

A zero-knowledge proof that: (in practice)

- Binds the note owner to the created nullifier,
  (hash commitment)
- Proves the ownership of the note,
  (several hashes in a Merkle tree)
- Authenticates the transaction from the owner to someone else.
  (signature verification)

The signature curve is chosen so that the authentication circuit is *compatible* with the nullifier and ownership circuits.
...but what is a circuit, exactly?

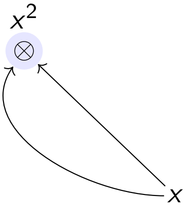
An example of circuit

Proving the knowledge of a solution ($x = 3$) of $x^3 + x + 5 = 35$:

x

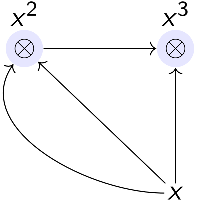
An example of circuit

Proving the knowledge of a solution ($x = 3$) of $x^3 + x + 5 = 35$:



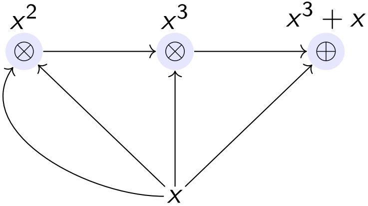
An example of circuit

Proving the knowledge of a solution ($x = 3$) of $x^3 + x + 5 = 35$:



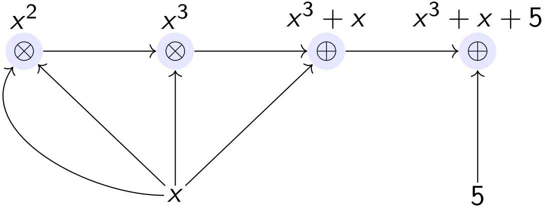
An example of circuit

Proving the knowledge of a solution ($x = 3$) of $x^3 + x + 5 = 35$:



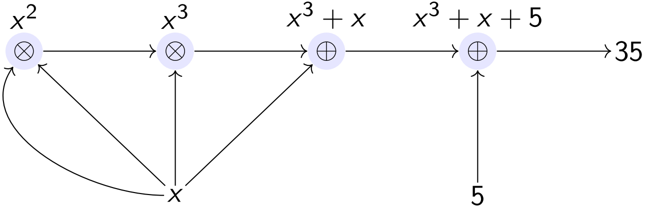
An example of circuit

Proving the knowledge of a solution ($x = 3$) of $x^3 + x + 5 = 35$:



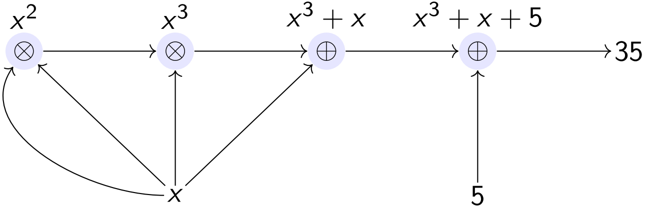
An example of circuit

Proving the knowledge of a solution ($x = 3$) of $x^3 + x + 5 = 35$:



An example of circuit

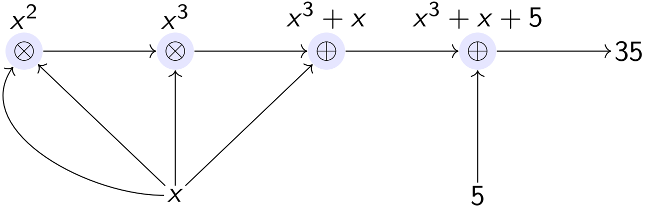
Proving the knowledge of a solution ($x = 3$) of $x^3 + x + 5 = 35$:



- (almost) Only additions and multiplications,
- Arithmetic is modulo a prime q (in $\mathbb{F}_q$).
  - Circom: BN254 scalar field,
  - Halo2: Pallas (or Vesta) scalar field,
  - Etc.

An example of circuit

Proving the knowledge of a solution ($x = 3$) of $x^3 + x + 5 = 35$:

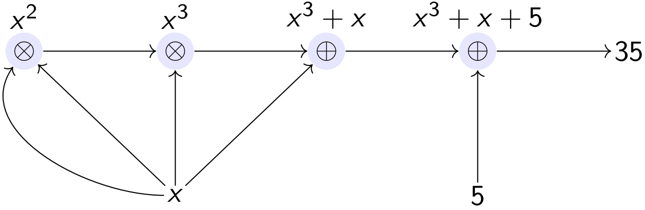


- (almost) Only additions and multiplications,
- Arithmetic is modulo a prime q (in $\mathbb{F}_q$).
  - Circom: BN254 scalar field,
  - Halo2: Pallas (or Vesta) scalar field,
  - Etc.

If the proof field is different, emulation is possible, but expensive!

An example of circuit

Proving the knowledge of a solution ($x = 3$) of $x^3 + x + 5 = 35$:

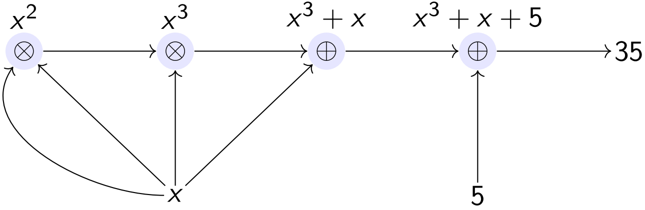


- (almost) Only additions and multiplications,
- Arithmetic is modulo a prime q (in $\mathbb{F}_q$).
  - Circom: BN254 scalar field,
  - Halo2: Pallas (or Vesta) scalar field,
  - Etc.

If the proof field is different, emulation is possible, but expensive!
The authentication circuit must be modulo q.
Select a curve whose base field is modulo q.

An example of circuit

Proving the knowledge of a solution ($x = 3$) of $x^3 + x + 5 = 35$:



- (almost) Only additions and multiplications,
- Arithmetic is modulo a prime q (in $\mathbb{F}_q$).
  - Circom: BN254 scalar field,
  - Halo2: Pallas (or Vesta) scalar field,
  - Etc.

If the proof field is different, emulation is possible, but expensive!
The authentication circuit must be modulo q.
Select a curve whose base field is modulo q.
(Railgun uses Circom; BabyJubjub defined over BN254 scalar field)

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = P$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2]P$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]P$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^3]P$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^4]P$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^4]P + P$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2]([2^4]P + P)$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2]([2^4]P + P) + P$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[1000110 1_2]P$?
   $Q = [2]([2]([2^4]P + P) + P)$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P)$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions $(\log(k) = 256)$.
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

$$\begin{aligned} & [10001101_2]P_1 \\ & +[10111000_2]P_2 \\ \text{Precomputed table}: \quad & \{0, P_1, P_2, P_1 + P_2\} \end{aligned}$$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

$$\begin{aligned} & [10001101_2]P_1 \\ & +[10111000_2]P_2 \\ \text{Precomputed table :} \quad & \{0, P_1, P_2, P_1 + P_2\} \end{aligned}$$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

$$\begin{aligned} & [10\textcolor{red}{0}01101_2]P_1 \\ & +[10\textcolor{red}{1}11000_2]P_2 \\ \text{Precomputed table :} \quad & \{0, P_1, \textcolor{red}{P_2}, P_1 + P_2\} \end{aligned}$$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

$$\begin{aligned} & [10001101_2]P_1 \\ & +[10111000_2]P_2 \\ \text{Precomputed table}: \quad & \{0, P_1, P_2, P_1 + P_2\} \end{aligned}$$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

$$\begin{aligned} & [10001101_2]P_1 \\ & +[10111000_2]P_2 \\ \text{Precomputed table}: \quad & \{0, P_1, P_2, P_1 + P_2\} \end{aligned}$$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

$$\begin{aligned} &[10001101_2]P_1 \\ &+[10111000_2]P_2 \\ \text{Precomputed table}: \quad &\{0, P_1, P_2, P_1 + P_2\} \end{aligned}$$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

$$\begin{aligned} & [10001101_2]P_1 \\ & +[10111000_2]P_2 \\ \text{Precomputed table}: \quad & \{0, P_1, P_2, P_1 + P_2\} \end{aligned}$$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

$$\begin{aligned} & [10001101_2]P_1 \\ & +[10111000_2]P_2 \\ \text{Precomputed table}: \quad & \{0, P_1, P_2, P_1 + P_2\} \end{aligned}$$

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

$$\begin{aligned} &[10001101_2]P_1 \\ &+[10111000_2]P_2 \\ \text{Precomputed table}: \quad &\{0, P_1, P_2, P_1 + P_2\} \end{aligned}$$

Cost: $o(\log(k))$ additions + precomputation table.

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

$$\begin{aligned} & [10001101_2]P_1 \\ & +[10111000_2]P_2 \\ \text{Precomputed table}: \quad & \{0, P_1, P_2, P_1 + P_2\} \end{aligned}$$

   Cost: $o(\log(k))$ additions + precomputation table.

In practice:

- Off-chain: very efficient (a few milliseconds),
- On-chain: non-native scalar multiplication ≈ 4 seconds.

Elliptic curve in circuits

Signature verification: Elliptic curve scalar multiplications.

EdDSA in Wikipedia: $2^c SB \stackrel{?}{=} 2^c R + 2^c H(R||A||M)A$.

1. How to compute $[10001101_2]P$?
   $Q = [2^2]([2]([2^4]P + P) + P) + P = [141]P$ ✓
   Cost: $o(\log(k))$ additions ($\log(k) = 256$).
2. How to compute $[10001101_2]P_1 + [10111000_2]P_2$?

$$\begin{aligned} & [10001101_2]P_1 \\ & +[10111000_2]P_2 \\ \text{Precomputed table}: \quad & \{0, P_1, P_2, P_1 + P_2\} \end{aligned}$$

   Cost: $o(\log(k))$ additions + precomputation table.

In practice:

- Off-chain: very efficient (a few milliseconds),
- On-chain: non-native scalar multiplication ≈ 4 seconds.

This work: reduce scalars using circuit hints:

- Smaller circuits for signature verification,
- Improved proof computation for circuits of elliptic curves.

Hinted scalar multiplication

Consider the equation $[k]P = Q$.

Hinted scalar multiplication

Consider the equation $[k]P = Q$.
The scalar k can be decomposed as $k = x/z \bmod r$.

Hinted scalar multiplication

Consider the equation $[k]P = Q$.
The scalar k can be decomposed as $k = x/z \bmod r$.
$\{x - kz = 0 \bmod r\}$ is a lattice of dimension 2:

$$\begin{pmatrix} r & 0 \\ k & 1 \end{pmatrix} = \begin{pmatrix} \square\square\square\square\square\square & 0 \\ \square\square\square\square\square\square & 1 \end{pmatrix}$$

Hinted scalar multiplication

Consider the equation $[k]P = Q$.
The scalar k can be decomposed as $k = x/z \bmod r$.
$\{x - kz = 0 \bmod r\}$ is a lattice of dimension 2:

$$\begin{pmatrix} r & 0 \\ k & 1 \end{pmatrix} = \begin{pmatrix} \square\square\square\square\square\square & 0 \\ \square\square\square\square\square\square & 1 \end{pmatrix} \sim \begin{pmatrix} \square\square\square & \square\square\square \\ \square\square\square & \square\square\square \end{pmatrix}$$

Apply lattice reduction (like LLL) to find a short vectors.
Expected size $\sqrt{r}$.

Hinted scalar multiplication

Consider the equation $[k]P = Q$.
The scalar k can be decomposed as $k = x/z \bmod r$.
$\{x - kz = 0 \bmod r\}$ is a lattice of dimension 2:

$$\begin{pmatrix} r & 0 \\ k & 1 \end{pmatrix} = \begin{pmatrix} \square\square\square\square\square\square & 0 \\ \square\square\square\square\square\square & 1 \end{pmatrix} \sim \begin{pmatrix} \square\square\square & \square\square\square \\ \square\square\square & \square\square\square \end{pmatrix}$$

Apply lattice reduction (like LLL) to find a short vectors.
Expected size $\sqrt{r}$.

$$[k]P = Q \iff [x]P - [z]Q = 0$$

Hinted scalar multiplication

Consider the equation $[k]P = Q$.
The scalar k can be decomposed as $k = x/z \bmod r$.
$\{x - kz = 0 \bmod r\}$ is a lattice of dimension 2:

$$\begin{pmatrix} r & 0 \\ k & 1 \end{pmatrix} = \begin{pmatrix} \square\square\square\square\square\square & 0 \\ \square\square\square\square\square\square & 1 \end{pmatrix} \sim \begin{pmatrix} \square\square\square & \square\square\square \\ \square\square\square & \square\square\square \end{pmatrix}$$

Apply lattice reduction (like LLL) to find a short vectors.
Expected size $\sqrt{r}$.

$$[k]P = Q \iff [x]P - [z]Q = 0$$

- $[k]P = Q$: scalar of size 256,
- $[x]P - [z]Q = 0$: scalars of size 128. ✓

Hinted double scalar multiplication

Consider the equation $[k_1]P_1 + [k_2]P_2 = Q$.

Hinted double scalar multiplication

Consider the equation $[k_1]P_1 + [k_2]P_2 = Q$.
The scalars k_1, k_2 can be simultaneously decomposed as

$$k_1 = \frac{x_1}{z} \bmod r, \qquad k_2 = \frac{x_2}{z} \bmod r.$$

Hinted double scalar multiplication

Consider the equation $[k_1]P_1 + [k_2]P_2 = Q$.
The scalars k_1, k_2 can be simultaneously decomposed as

$$k_1 = \frac{x_1}{z} \bmod r, \qquad k_2 = \frac{x_2}{z} \bmod r.$$

$\{x_1 - k_1 z = 0 \bmod r \text{ and } x_2 - k_2 z = 0 \bmod r\}$ form a lattice of dimension 3:

$$\begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ k_1 & k_2 & 1 \end{pmatrix} = \begin{pmatrix} \square\square\square\square\square\square & 0 & 0 \\ 0 & \square\square\square\square\square\square & 0 \\ \square\square\square\square\square\square & \square\square\square\square\square\square & 1 \end{pmatrix}$$

Hinted double scalar multiplication

Consider the equation $[k_1]P_1 + [k_2]P_2 = Q$.
The scalars k_1, k_2 can be simultaneously decomposed as

$$k_1 = \frac{x_1}{z} \bmod r, \qquad k_2 = \frac{x_2}{z} \bmod r.$$

$\{x_1 - k_1 z = 0 \bmod r \text{ and } x_2 - k_2 z = 0 \bmod r\}$ form a lattice of dimension 3:

$$\begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ k_1 & k_2 & 1 \end{pmatrix} = \begin{pmatrix} \square\square\square\square\square\square & 0 & 0 \\ 0 & \square\square\square\square\square\square & 0 \\ \square\square\square\square\square\square & \square\square\square\square\square\square & 1 \end{pmatrix} \sim \begin{pmatrix} \square\square\square\square & \square\square\square\square & \square\square\square\square \\ \square\square\square\square & \square\square\square\square & \square\square\square\square \\ \square\square\square\square & \square\square\square\square & \square\square\square\square \end{pmatrix}$$

Apply lattice reduction (like LLL) to find a short vectors.
Expected size $\sqrt[3]{r^2}$.

Hinted double scalar multiplication

Consider the equation $[k_1]P_1 + [k_2]P_2 = Q$.
The scalars k_1, k_2 can be simultaneously decomposed as

$$k_1 = \frac{x_1}{z} \bmod r, \qquad k_2 = \frac{x_2}{z} \bmod r.$$

$\{x_1 - k_1 z = 0 \bmod r \text{ and } x_2 - k_2 z = 0 \bmod r\}$ form a lattice of dimension 3:

$$\begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ k_1 & k_2 & 1 \end{pmatrix} = \begin{pmatrix} \square\square\square\square\square\square & 0 & 0 \\ 0 & \square\square\square\square\square\square & 0 \\ \square\square\square\square\square\square & \square\square\square\square\square\square & 1 \end{pmatrix} \sim \begin{pmatrix} \square\square\square\square & \square\square\square\square & \square\square\square\square \\ \square\square\square\square & \square\square\square\square & \square\square\square\square \\ \square\square\square\square & \square\square\square\square & \square\square\square\square \end{pmatrix}$$

Apply lattice reduction (like LLL) to find a short vectors.
Expected size $\sqrt[3]{r^2}$.

$$[k_1]P_1 + [k_2]P_2 = Q \iff [x_1]P_1 + [x_2]P_2 - [z]Q = 0$$

Triple scalar multiplication with scalars of 171 bits. ✓

GLV hinted scalar multiplication

GLV: a technique to faster scalar multiplication for specific curves:
$[k]P = [k_1]P + [k_2]\psi(P)$ where $\psi(P)$ is easy to compute, and k_1, k_2 halved size.

GLV hinted scalar multiplication

GLV: a technique to faster scalar multiplication for specific curves:
$[k]P = [k_1]P + [k_2]\psi(P)$ where $\psi(P)$ is easy to compute, and k_1, k_2 halved size.
GLV with hints: a fraction decompositions in $\mathbb{Z}[\lambda]$ where λ is an eigenvalue of ψ.

GLV hinted scalar multiplication

GLV: a technique to faster scalar multiplication for specific curves:
$[k]P = [k_1]P + [k_2]\psi(P)$ where $\psi(P)$ is easy to compute, and k_1, k_2 halved size.
GLV with hints: a fraction decompositions in $\mathbb{Z}[\lambda]$ where λ is an eigenvalue of ψ.

Single scalar multiplication with GLV and hint

$$\begin{pmatrix} r & 0 & 0 & 0 \\ -\lambda & 1 & 0 & 0 \\ k & 0 & 1 & 0 \\ 0 & 0 & -\lambda & 1 \end{pmatrix}$$

$$[k]P = Q$$
$$\Updownarrow$$
$$[x]P + [y]\psi(P) - [z]Q - [t]\psi(Q) = 0$$

Quadruple 64-bit scalar multiplication.

GLV hinted scalar multiplication

GLV: a technique to faster scalar multiplication for specific curves:
$[k]P = [k_1]P + [k_2]\psi(P)$ where $\psi(P)$ is easy to compute, and k_1, k_2 halved size.
GLV with hints: a fraction decompositions in $\mathbb{Z}[\lambda]$ where λ is an eigenvalue of ψ.

Single scalar multiplication with GLV and hint

$$\begin{pmatrix} r & 0 & 0 & 0 \\ -\lambda & 1 & 0 & 0 \\ k & 0 & 1 & 0 \\ 0 & 0 & -\lambda & 1 \end{pmatrix}$$

$$[k]P = Q$$
$$\Updownarrow$$
$$[x]P + [y]\psi(P) - [z]Q - [t]\psi(Q) = 0$$

Quadruple 64-bit scalar multiplication.

Double scalar multiplication with GLV and hint

$$\begin{pmatrix} r & 0 & 0 & 0 & 0 & 0 \\ -\lambda & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & r & 0 & 0 & 0 \\ 0 & 0 & -\lambda & 1 & 0 & 0 \\ k_1 & 0 & k_2 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -\lambda & 1 \end{pmatrix}$$

$$[k_1]P_1 + [k_2]P_2$$
$$\Updownarrow$$
$$[x_1]P_1 + [y_1]\psi(P_1) + [x_2]P_2 + [y_2]\psi(P_2)$$
$$-[z]Q - [t]\psi(Q) = 0$$

Sextuple 86-bit scalar multiplication.

Practical results

- Implementation in GNARK with two proof systems: R1CS and SCS.
- Native lookups are expensive $\Longrightarrow$ GLV with hints become expensive.

Practical results

- Implementation in GNARK with two proof systems: R1CS and SCS.
- Native lookups are expensive $\Longrightarrow$ GLV with hints become expensive.

Circuit	R1CS			SCS		
	Before	This work		Before	This work	
Non-native (P256)	157 685	78 940	50%	612 759	294 128	52%
Non-native GLV (secP256k1)	78 940	60 089	24%	385 461	223 188	42%
Native (Jubjub)	3 314	2 401	28%	5 863	4 549	22%
Native GLV (Bandersnatch)	2 621	4 038		4 712	8 519	

Practical results

- Implementation in GNARK with two proof systems: R1CS and SCS.
- Native lookups are expensive $\Longrightarrow$ GLV with hints become expensive.

Circuit	R1CS			SCS		
	Before	This work		Before	This work	
Non-native (P256)	157 685	78 940	50%	612 759	294 128	52%
Non-native GLV (secP256k1)	78 940	60 089	24%	385 461	223 188	42%
Native (Jubjub)	3 314	2 401	28%	5 863	4 549	22%
Native GLV (Bandersnatch)	2 621	4 038		4 712	8 519	

- The scalar decomposition is not optimal yet (`xgcd` vs `lll`),
- The cost is implementation-dependent,
- Out-of-circuit considerations are also important,
- Double scalar multiplication not implemented yet.

Conclusion and perspectives

- Reduction of elliptic curve circuits

Conclusion and perspectives

- Reduction of elliptic curve circuits
- Significant for non-native circuits (for example, ZK passports)
  - Current implementations take few seconds,
  - Expected reduction of roughly 40%.

Conclusion and perspectives

- Reduction of elliptic curve circuits
- Significant for non-native circuits (for example, ZK passports)
  - Current implementations take few seconds,
  - Expected reduction of roughly 40%.
- Reduction for native circuits (for example, Railgun)
  - Moving from BN254 to BLS12-381 for a higher security,
  - Reducing the circuit size compared to current Circom implementations.

Conclusion and perspectives

- Reduction of elliptic curve circuits
- Significant for non-native circuits (for example, ZK passports)
  - Current implementations take few seconds,
  - Expected reduction of roughly 40%.
- Reduction for native circuits (for example, Railgun)
  - Moving from BN254 to BLS12-381 for a higher security,
  - Reducing the circuit size compared to current Circom implementations.
- Implementation work remaining:

Conclusion and perspectives

- Reduction of elliptic curve circuits
- Significant for non-native circuits (for example, ZK passports)
  - Current implementations take few seconds,
  - Expected reduction of roughly 40%.
- Reduction for native circuits (for example, Railgun)
  - Moving from BN254 to BLS12-381 for a higher security,
  - Reducing the circuit size compared to current Circom implementations.
- Implementation work remaining:
  - ☐ Double scalar multiplications,

Conclusion and perspectives

- Reduction of elliptic curve circuits
- Significant for non-native circuits (for example, ZK passports)
  - Current implementations take few seconds,
  - Expected reduction of roughly 40%.
- Reduction for native circuits (for example, Railgun)
  - Moving from BN254 to BLS12-381 for a higher security,
  - Reducing the circuit size compared to current Circom implementations.
- Implementation work remaining:
  - [ ] Double scalar multiplications,
  - [ ] Optimal scalar decomposition (`xgcd` vs lattice reduction),

Conclusion and perspectives

- Reduction of elliptic curve circuits
- Significant for non-native circuits (for example, ZK passports)
  - Current implementations take few seconds,
  - Expected reduction of roughly 40%.
- Reduction for native circuits (for example, Railgun)
  - Moving from BN254 to BLS12-381 for a higher security,
  - Reducing the circuit size compared to current Circom implementations.
- Implementation work remaining:
  - ☐ Double scalar multiplications,
  - ☐ Optimal scalar decomposition (`xgcd` vs lattice reduction),
  - ☐ STARK implementation (non-native circuits $\Longrightarrow$ expected speed-up),

Conclusion and perspectives

- Reduction of elliptic curve circuits
- Significant for non-native circuits (for example, ZK passports)
  - Current implementations take few seconds,
  - Expected reduction of roughly 40%.
- Reduction for native circuits (for example, Railgun)
  - Moving from BN254 to BLS12-381 for a higher security,
  - Reducing the circuit size compared to current Circom implementations.
- Implementation work remaining:
  - [ ] Double scalar multiplications,
  - [ ] Optimal scalar decomposition (`xgcd` vs lattice reduction),
  - [ ] STARK implementation (non-native circuits $\implies$ expected speed-up),
  - [ ] Lookup optimizations for native circuits.

Conclusion and perspectives

- Reduction of elliptic curve circuits
- Significant for non-native circuits (for example, ZK passports)
  - Current implementations take few seconds,
  - Expected reduction of roughly 40%.
- Reduction for native circuits (for example, Railgun)
  - Moving from BN254 to BLS12-381 for a higher security,
  - Reducing the circuit size compared to current Circom implementations.
- Implementation work remaining:
  - [ ] Double scalar multiplications,
  - [ ] Optimal scalar decomposition (`xgcd` vs lattice reduction),
  - [ ] STARK implementation (non-native circuits $\Longrightarrow$ expected speed-up),
  - [ ] Lookup optimizations for native circuits.

Thank you for your attention.