plookup: speeding up SNARKs on non-friendly functions with lookup tables

Ariel Gabizon Zachary J. Williamson
Aztec

SNARKs are easy prey in a world full of nasty binary functions

$\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{F}$

Want to show $\mathbf{c} = \mathbf{a} \quad \oplus \quad \mathbf{b}$ as 8-bit strings

SNARKs are easy prey in a world full of nasty binary functions

$\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{F}$

Want to show $\mathbf{c} = \mathbf{a} \quad \oplus \quad \mathbf{b}$ as 8-bit strings

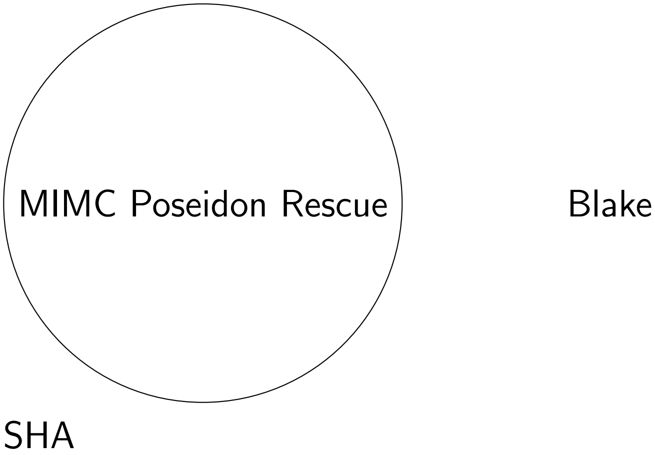
Standard way requires 25-32 constraints: Give bit decomposition of $\mathbf{a}, \mathbf{b}, \mathbf{c}$, check bitwise xor.

Want to show $\mathbf{c} = \mathbf{a} \oplus \mathbf{b}$ as 8-bit strings.

Standard way requires 25-32 constraints: Give bit decomposition of $\mathbf{a}, \mathbf{b}, \mathbf{c}$, check bitwise xor.

This is a **multiplicative** *factor you pay on each small operation while computing SHA/BLAKE*

Approach 1: Keep SNARKs in friendly neighborhoods



Our Approach: lookup tables (see also: Arya[Bootle, Cerulli, Groth, Jakobsen, Maller])

Precompute table $\mathbf{T}$ of all triplets $(\mathbf{a}, \mathbf{b}, \mathbf{c})$
s.t. $\mathbf{c} = \mathbf{a} \quad \oplus \quad \mathbf{b}$.

Our Approach: lookup tables (see also: Arya[Bootle, Cerulli, Groth, Jakobsen, Maller])

Precompute table $\mathbf{T}$ of all triplets $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ s.t. $\mathbf{c} = \mathbf{a} \oplus \mathbf{b}$.

Instead of representing $\oplus$ logic, check that $(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in \mathbf{T}$

Our Approach: lookup tables (see also: Arya[Bootle, Cerulli, Groth, Jakobsen, Maller])

Precompute table T of all triplets $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ s.t. $\mathbf{c} = \mathbf{a} \oplus \mathbf{b}$.

Instead of representing $\oplus$ logic, check that $(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in \mathsf{T}$

After enough lookups, has amortized cost of ~ 1 constraint per $\oplus$.

The plookup protocol in a nutshell

(a simpler protocol we came up with while preparing the slides)

Basic tool: The multiset check

example: Given $\mathbf{a}, \mathbf{b} \in \mathbb{F}^3$, want to check $\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\} \stackrel{?}{=} \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$

Basic tool: The multiset check

example: Given $\mathbf{a}, \mathbf{b} \in \mathbb{F}^3$, want to check $\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\} \stackrel{?}{=} \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$

Choose random $\boldsymbol{\gamma} \in \mathbb{F}$. Check

$$(\mathbf{a}_1+\boldsymbol{\gamma})(\mathbf{a}_2+\boldsymbol{\gamma})(\mathbf{a}_3+\boldsymbol{\gamma}) \stackrel{?}{=} (\mathbf{b}_1+\boldsymbol{\gamma})(\mathbf{b}_2+\boldsymbol{\gamma})(\mathbf{b}_3+\boldsymbol{\gamma})$$

Basic tool: The multiset check

example: Given $\mathbf{a}, \mathbf{b} \in \mathbb{F}^3$, want to check
$\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\} \stackrel{?}{=} \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$

Choose random $\boldsymbol{\gamma} \in \mathbb{F}$. Check

$$(\mathbf{a}_1+\boldsymbol{\gamma})(\mathbf{a}_2+\boldsymbol{\gamma})(\mathbf{a}_3+\boldsymbol{\gamma}) \stackrel{?}{=} (\mathbf{b}_1+\boldsymbol{\gamma})(\mathbf{b}_2+\boldsymbol{\gamma})(\mathbf{b}_3+\boldsymbol{\gamma})$$

If $\mathbf{a}, \mathbf{b}$ different as sets then w.h.p products different.

Basic tool: The multiset check

example: Given $\mathbf{a}, \mathbf{b} \in \mathbb{F}^3$, want to check
$\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\} \stackrel{?}{=} \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}$

Choose random $\boldsymbol{\gamma} \in \mathbb{F}$. Check

$$(\mathbf{a}_1+\boldsymbol{\gamma})(\mathbf{a}_2+\boldsymbol{\gamma})(\mathbf{a}_3+\boldsymbol{\gamma}) \stackrel{?}{=} (\mathbf{b}_1+\boldsymbol{\gamma})(\mathbf{b}_2+\boldsymbol{\gamma})(\mathbf{b}_3+\boldsymbol{\gamma})$$

If $\mathbf{a}, \mathbf{b}$ different as sets then w.h.p products different.

$\mathfrak{PlonK}$'s grand product implements this super efficiently

Witness $\mathbf{f} = \{\mathbf{f_i}\}_{\mathbf{i} \in [\mathbf{n}]}$ Table $\mathbf{t} = \{\mathbf{t_i}\}_{\mathbf{i} \in [\mathbf{d}]}$
Want to prove $\mathbf{f} \subset \mathbf{t}$. (using randomness we can reduce tuples to single elements).

Witness $\mathbf{f} = \{3, 1, 1\}$ Table $\mathbf{t} = \{1, 3, 4\}$

1. Prover commits to $\mathbf{s} :=$ *sorted* version of $\mathbf{f} \cup \mathbf{t}$.
   $\mathbf{s} := (1, 1, 1, 3, 3, 4)$

Witness $\mathbf{f} = \{3, 1, 1\}$ Table $\mathbf{t} = \{1, 3, 4\}$

1. Prover commits to $\boldsymbol{s} :=$ *sorted* version of $\mathbf{f} \cup \mathbf{t}$.
   $\boldsymbol{s} := (1, 1, 1, 3, 3, 4)$
2. Prover shows $\boldsymbol{s} = \mathbf{f} \cup \mathbf{t}$.

Witness $\mathbf{f} = \{3, 1, 1\}$ Table $\mathbf{t} = \{1, 3, 4\}$

1. Prover commits to $\boldsymbol{s} :=$ *sorted* version of $\mathbf{f} \cup \mathbf{t}$.
   $\boldsymbol{s} := (1, 1, 1, 3, 3, 4)$
2. Prover shows $\boldsymbol{s} = \mathbf{f} \cup \mathbf{t}$.
3. Look at difference multiset of $\boldsymbol{s}$
   $\mathbf{s}' := \{0, 0, 2, 0, 1\}$, and difference multiset of $\mathbf{t}$
   $\mathbf{t}' := \{2, 1\}$

Witness $\mathbf{f} = \{3, 1, 1\}$ Table $\mathbf{t} = \{1, 3, 4\}$

1. Prover commits to $\mathbf{s} :=$ *sorted* version of $\mathbf{f} \cup \mathbf{t}$.
   $\mathbf{s} := (1, 1, 1, 3, 3, 4)$
2. Prover shows $\mathbf{s} = \mathbf{f} \cup \mathbf{t}$.
3. Look at difference multiset of $\mathbf{s}$
   $\mathbf{s}' := \{0, 0, 2, 0, 1\}$, and difference multiset of $\mathbf{t}$
   $\mathbf{t}' := \{2, 1\}$
4. Prover shows $\mathbf{s}' = \mathbf{t}' \cup \{0, 0, 0\}$.