Stacking Sigmas

A. Goel¹, M. Green¹, M. Hall-Andersen², G. Kaptchuk³

1) Johns Hopkins University, Baltimore, USA.
2) Aarhus University, Aarhus, Denmark.
3) Boston University, Boston, USA.
- Introduction.
- Stackable $\Sigma$-protocols.
- Partially Binding Commitments and Stacking Compiler.
- Logarithmic Communication via Recursive Application.
- Wrapping up.
A tuple of algorithms $\Pi = (A, Z, \phi)$

\[
\begin{align*}
\text{P} & \leftarrow A(x, w; r) & \text{V} & \leftarrow \phi(x, a, c, z) \equiv \top \\
\text{c} & \leftarrow \{0, 1\}^\lambda
\end{align*}
\]
Goal: Proving Disjunctions (of $\Sigma$-Protocols).

Zero-knowledge proofs for statements of the form:

$$(x_1, \ldots, x_\ell) \in \mathcal{L}_{OR} \iff x_1 \in \mathcal{L}_1 \lor \ldots \lor x_\ell \in \mathcal{L}_\ell$$

**Goal:** If proving $x_i \in \mathcal{L}_i$ using $\Pi_i$ requires $\text{CC}(\Pi_i)$ communication, derive $\Pi'$ for $\mathcal{L}_{OR}$ with $\text{CC}(\Pi') \ll \sum_i \text{CC}(\Pi_i)$.

**Applications:** ring signatures, branching computation, WI from HVZK.
Prior Works: Generic Compiler, or, Space Saving.

Choose one:

- Generic compiler for $\Sigma$-protocols.
- Communication saving for a particular protocol.
This Work: Space Saving for a Large Class of Protocols.

Comm. saving disjunctions for a large class of $\Sigma$-protocols: $O(\log(\ell) + \text{cc}(\Pi))$ communication, *concrete efficiency*.

Not all $\Sigma$-protocols, but a wide class. e.g.

1. **Homorphism Preimage**: Schnorr, Chaum-Pedersen, etc.
2. **MPC-in-the-head**: KKW19 and Ligero (Circuit Sat.).
3. **Classics**: Blum87 (Hamiltonicity).
Stacking $\Sigma$-Protocols
Intuition: Preimages of One-Way Homorphism (e.g. Schnorr).

\[ \Pi_\psi : \mathcal{R}_\psi(x, w) := x \overset{?}{=} \psi(w) \]

\[ P \quad \forall \]

\[ r \leftarrow \text{dom}(\psi); a \leftarrow \psi(r) \quad \overset{a}{\rightarrow} \]

\[ c \quad \overset{c}{\leftarrow} \quad c \leftarrow C \]

\[ z \leftarrow cw + r \quad \overset{z}{\rightarrow} \quad \psi(z) \overset{?}{=} x^c a \]

Where \( C \subseteq \mathbb{Z} \), e.g. \( C = \mathbb{Z}_p \), \( \psi(w) = g^w \in \mathbb{G}_p \) (Schnorr).
How to simulate $\Pi_\psi$:

1. Sample 3rd round independently of $x$: $z \leftarrow \text{dom}(\psi)$
2. Compute accepting commitment: $a \leftarrow \psi(z) \cdot x^{-c}$

**Observation:** simulation of many $\Sigma$-protocols works similarly:

1. Sample 3rd round from a distribution (dependent on $c$)
2. Complete transcript using $x$

**Notable Example:** many MPC-in-the-head protocols: view of the opened parties often a string of uniformly random field elements.
Recyclable: The distribution of $z$ is independent of $x$, i.e.

$$z \leftarrow \mathcal{D}_c^{(z)}$$

EHVZK: Given (1) a statement $x$, (2) a challenge $c$ and (3) a last round message $z$. Can find a st. $\phi(x, a, c, z) = 1$.

$$a \leftarrow S_{EHVZK}^{EHVZK}(x, c, z)$$

If both are statisified $\implies$ “Stackable”; our techniques apply.
has a witness $w$ for $x_1$, prove $(w, x_1) \in R \lor (w, x_2) \in R$. **Idea:**

1. Prove the satisfied clause $(w, x_1) \in R$ obtain $\pi_1 = (a_1, c, z)$.
2. Apply “extended simulator” for the other clause:

$$a_2 \leftarrow S^{EHVZK}(x_2, c, z)$$

**Does Not Work:**

- Cannot generate $a_2$ before seeing $c$ (needs to simulate).
- Cannot send $c$ before receiving $a_1$ (for soundness).
Partially Binding Commitments
1-of-2 Partially Binding Commitments

A commitment scheme enabling “limited equivocation”:

\[ P \]

Commit to 2-tuples \((v_1, v_2)\) and index \(i \in \{1, 2\}\).

\[ P \]

Can later equivocate at position \(\bar{i}\), but not \(i\).

\[ V \]

Never learns the binding position \(i\).

\[ P \quad V \]

Enables \(P\) to “send” one of \(a_1/a_2\) to \(V\); without revealing which.

**Now**: a simple construction.
Simple Construction: 1-of-2 Example

From Pedersen commitments. **Setup:** $h, g \in \mathbb{G}$.

$(ck, ek) \leftarrow \text{Gen}(i)$, with binding position $i \in \{1, 2\}$.

1. Pick $ek \leftarrow \mathbb{Z}_{|\mathbb{G}|}$
2. Let $h^i \leftarrow g^{ek}$. Pick $h$ st. $h_2 \cdot h_1 = h$
3. Commitment key is $ck = h_1$.

$c \leftarrow \text{Com}(ck = (h_1), (v_1, v_2), (r_1, r_2))$:

1. Compute $h_2 = h \cdot h_1^{-1}$.
2. Output $c = (g^{r_1}h_1^{v_1}, g^{r_2}h_2^{v_2})$.

**Easy to see:** can easily equivocate in position $\bar{i}$ using $x$, but equivocating in both positions $\implies$ computing $\text{dlog}_g(h)$.
Idea: Space Saving Disjunctions for Stackable $\Pi$.

1. Run $a_1 \leftarrow A(x_1, w_1; r)$. $(ck, ek) \leftarrow \text{Gen}(i = 1)$

2. Sends $ck, a' = \text{Com}(ck, (a_1, \bot))$ to

3. Sends $c$ to

4. Finishes the first transcript $z \leftarrow Z(x_1, w_1; r)$.

5. Simulates $a_2$ using $(c, z)$ and opens $a'$ to $(a_1, a_2)$.
Idea: Space Saving Disjunctions for Stackable $\Pi$. 

P

$(x_1, w) \in \mathcal{R}, i = 1$

c_k, e_k \leftarrow \text{Gen}(i = 1); a_1 \leftarrow A(x_1, w) \quad a' = \text{Com}(c_k, (a_1, \perp); r) \\

c' \\

z \leftarrow Z(c, x_1, w)

a_2 \leftarrow S_{EHVZK}(x_2, c', z)

r' \leftarrow \text{Equiv}(e_k, (a_1, \perp), (a_1, a_2), r) \quad z' = (z, r')

V

a_1 \leftarrow S_{EHVZK}(c', x_1, z)

a_2 \leftarrow S_{EHVZK}(c', x_2, z)

a' \leftarrow \text{Com}((a_1, a_2), r')

Apply Compiler Again.

Stackable \( L, \text{cc}(\Pi) \) → Stackable \( L', \text{cc}(\Pi') \)

\[ L' = x_1 \in L \lor x_2 \in L \]

\[ \text{cc}(\Pi') = \text{cc}(\Pi) + O(\lambda) \]

\[ L'' = (x_1 \in L \lor x_2 \in L) \lor (x_3 \in L \lor x_4 \in L) = (x_1, x_2) \in L' \lor (x_3, x_4) \in L' \]

Do it again! (log\(_2\)(\(\ell\)) times for \(\ell\) clauses).

Total Communication: \(\log_2(\ell) \cdot O(\lambda) + \text{cc}(\Pi)\).
**Slight Generalization: “Cross Stacking”**

**Generalization:** Distinct protocols \( \Pi, \Pi' \) which share \( D(z) \)
(or some trivial “conversion” is possible, e.g. padding/packing).

**Informally:** Finish the transcript of \( \Pi \), obtain \((a, c, z)\), simulate \( \Pi' \) using \( z \) – as in the case of a single protocol.

**Example:** KKW18\(^1\) over \( \mathbb{F}_2 \) and KKW18 over \( \mathbb{Z}_{2^{16}} \). In both cases “\( z \)” consists of uniformly random ring elements (bits).

\(^1\)“Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures“, Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang
Π \quad \text{Stackable} \quad \mathcal{L}, \text{cc}(\Pi) \quad \rightsquigarrow \quad \Pi' \quad \text{Stackable} \quad \mathcal{L}' = x_1 \in \mathcal{L} \lor x_2 \in \mathcal{L} \quad \text{cc}(\Pi') = \text{cc}(\Pi) + O(\lambda)

Thanks For Your Attention.

See Full Paper: https://ia.cr/2021/422

Aarushi Goel, Matthew Green, Mathias Hall-Andersen, Gabriel Kaptchuk