Stacking Sigmas

A. Goel[1], M. Green[1], M. Hall-Andersen[2], G. Kaptchuk[3]

1) Johns Hopkins University, Baltimore, USA.

2) Aarhus University, Aarhus, Denmark.

3) Boston University, Boston, USA.

- Introduction.
- Stackable Σ-protocols.
- Partially Binding Commitments and Stacking Compiler.
- Logarithmic Communication via Recursive Application.
- Wrapping up.

Introduction

Σ-Protocols.

A tuple of algorithms $\Pi = (A, Z, \phi)$

P

V

P		V
$a \leftarrow A(x, w; r)$	$\xrightarrow{\quad a \quad}$	
	$\xleftarrow{\quad c \quad}$	$c \xleftarrow{\$} \{0,1\}^{\lambda}$
$z \leftarrow Z(c, x, w; r)$	$\xrightarrow{\quad z \quad}$	
		$\phi(x, a, c, z) \stackrel{?}{=} \top$

Goal: Proving Disjunctions (of Σ-Protocols).

Zero-knowledge proofs for statements of the form:

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

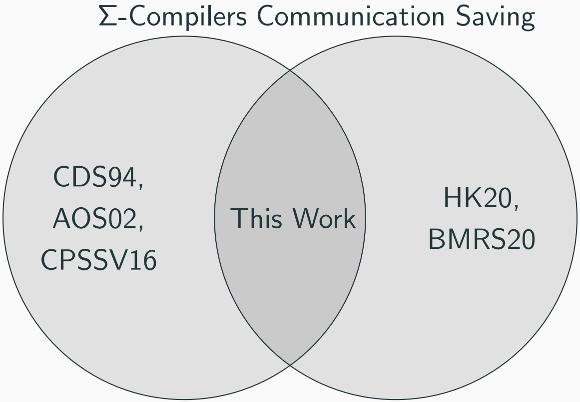
Goal: If proving $x_i \in \mathcal{L}_i$ using Π_i requires $\mathtt{CC}(\Pi_i)$ communication, derive Π' for $\mathcal{L}_{\mathrm{OR}}$ with $\mathtt{CC}(\Pi') \ll \sum_i \mathtt{CC}(\Pi_i)$.

Applications: ring signatures, branching computation, WI from HVZK.

Prior Works: Generic Compiler, or, Space Saving.

Choose one:

- Generic compiler for Σ-protocols.
- Communication saving for a particular protocol.

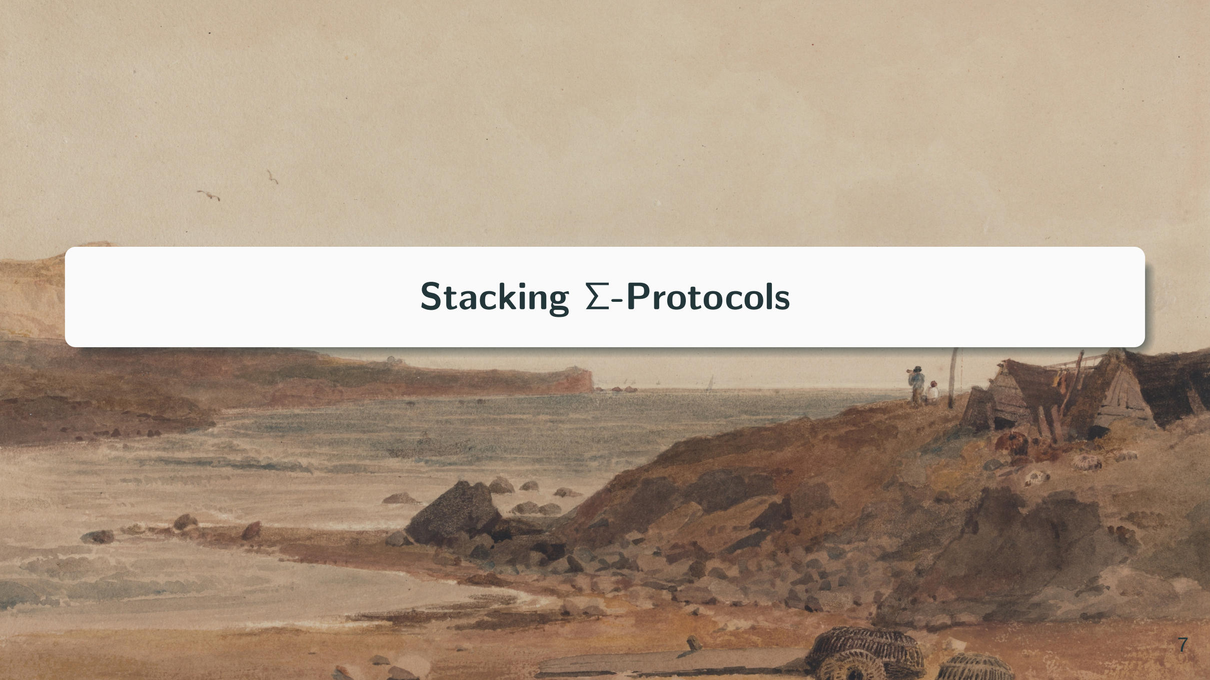


This Work: Space Saving for a Large Class of Protocols.

Comm. saving disjunctions for a large class of Σ-protocols: $O(\log(\ell) + \mathtt{CC}(\Pi))$ communication, concrete efficiency.

Not all Σ-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 Σ-Protocols

Intuition: Preimages of One-Way Homorphism (e.g. Schnorr).

$\Pi_\psi : \mathcal{R}_\psi(x, w) := x \stackrel{?}{=} \psi(w)$

P		V
		
$r \stackrel{\$}{\leftarrow} \texttt{dom}(\psi); a \leftarrow \psi(r)$	$\xrightarrow{a}$	
	$\xleftarrow{c}$	$c \stackrel{\$}{\leftarrow} C$
$z \leftarrow cw + r$	$\xrightarrow{z}$	$\psi(z) \stackrel{?}{=} x^c a$

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

Simulation

How to simulate Π_ψ:

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

Observation: simulation of many Σ-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.

Simulation: Extended Honest Verifier Zero-Knowledge.

Recyclable: The distribution of z is independent of x, i.e.

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

EHVZK: Given (1) a statement x, (2) a challenge c and (3) a last round message z.

Can find a st. $\phi(x, a, c, z) = 1$.

$$a \leftarrow \mathcal{S}^{\mathrm{EHVZK}}(x, c, z)$$

If both are statisified $\implies$ "Stackable"; our techniques apply.

Straw Man: Space Saving Disjunctions for Stackable Π.

P

has a witness w for x_1, prove $(w, x_1) \in \mathcal{R} \vee (w, x_2) \in \mathcal{R}$. **Idea:**

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

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

Does Not Work:

P

Cannot generate a_2 before seeing c (needs to simulate).

V

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.

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:** $h, g \in \mathbb{G}$.

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

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

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

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 $\implies$ computating $\mathsf{dlog}_g(h)$.

Idea: Space Saving Disjunctions for Stackable Π.

1.  P run $a_1 \leftarrow A(x_1, w_1; r)$. $(\mathsf{ck}, \mathsf{ek}) \leftarrow \mathsf{Gen}(i = 1)$
2.  P sends $\mathsf{ck}, a' = \mathsf{Com}(\mathsf{ck}, (a_1, \bot))$ to  V
3.  V sends c to  P
4.  P finishes the first transcript $z \leftarrow Z(x_1, w_1; r)$.
5.  P simulates a_2 using (c, z) and opens a' to (a_1, a_2).

Idea: Space Saving Disjunctions for Stackable Π.

P

V

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

$\mathsf{ck}, \mathsf{ek} \leftarrow \mathsf{Gen}(i = 1); a_1 \leftarrow A(x_1, w)$ $\xrightarrow{a' = \mathsf{Com}(\mathsf{ck}, (a_1, \perp); r)}$

$\xleftarrow{c'}$

$z \leftarrow Z(c, x_1, w)$

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

$r' \leftarrow \mathsf{Equiv}(\mathsf{ek}, (a_1, \perp), (a_1, a_2), r)$ $\xrightarrow{z' = (z, r')}$

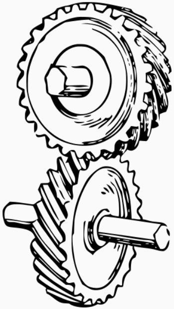
$a_1 \leftarrow \mathcal{S}^{\text{EHVZK}}(c', x_1, z)$

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

$a' \stackrel{?}{=} \mathsf{Com}((a_1, a_2), r')$

Recursive Application: Log. Communication.

Apply Compiler Again.

Π →  → Π'

Stackable

$\mathcal{L}$, $\mathsf{CC}(\Pi)$

Stackable

$\mathcal{L}' = x_1 \in \mathcal{L} \vee x_2 \in \mathcal{L}$

$\mathsf{CC}(\Pi') = \mathsf{CC}(\Pi) + O(\lambda)$

$$\mathcal{L}'' = (x_1 \in \mathcal{L} \vee x_2 \in \mathcal{L}) \vee (x_3 \in \mathcal{L} \vee x_4 \in \mathcal{L}) = (x_1, x_2) \in \mathcal{L}' \vee (x_3, x_4) \in \mathcal{L}'$$

Do it again! ($\log_2(\ell)$ times for ℓ clauses).

Total Communication: $\log_2(\ell) \cdot O(\lambda) + \mathsf{CC}(\Pi)$.

Slight Generalization: "Cross Stacking"

Generalization: Distinct protocols Π, Π' which share $\mathcal{D}^{(z)}$ (or some trivial "conversion" is possible, e.g. padding/packing).

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

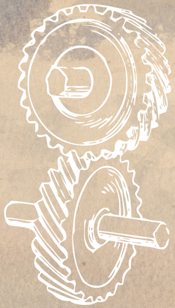
Example: KKW18[1] over $\mathbb{F}_2$ and KKW18 over $\mathbb{Z}_{2^{16}}$. In both cases "z" consists of uniformly random ring elements (bits).

[1] "Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures", Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang

Π

Stackable

$\mathcal{L}$, $\mathrm{CC}(\Pi)$



Π'

Stackable

$\mathcal{L}' = \mathbf{x_1} \in \mathcal{L} \vee \mathbf{x_2} \in \mathcal{L}$

$\mathrm{CC}(\mathbf{\Pi}') = \mathrm{CC}(\mathbf{\Pi}) + \mathbf{O}(\lambda)$

Thanks For Your Attention.

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

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