CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

Rate-1 Fully Local Somewhere Extractable Hashing from DDH

P. Branco, N. Döttling, A. Srinivasan, R. Zanotto

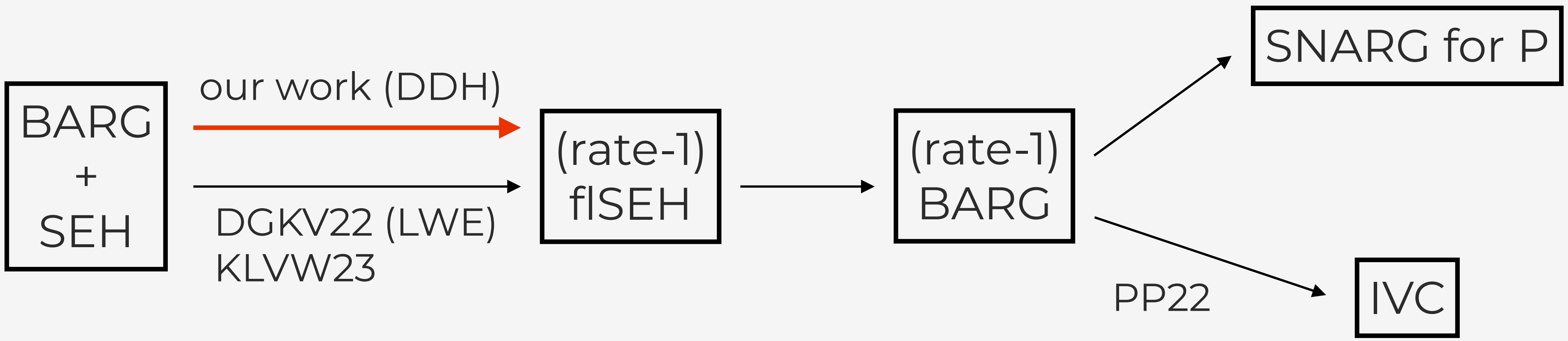
PKC 2024 | 16.04.2024

Contents

Background and applications

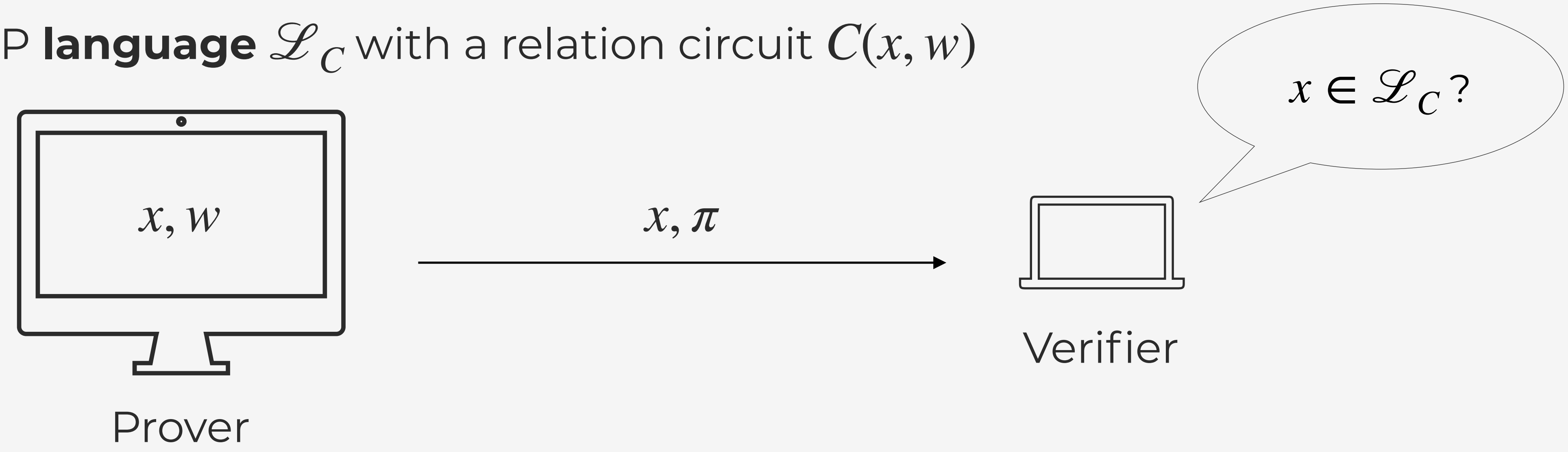
Outline of the result and background

- The main topic and goal of this line of research are **SNARGs**
- SNARGs for NP are known from ROM, impossible* in plain model [GW11]
- We can restrict to subclasses, in particular **P** and **batch-NP**
  - via correlation intractable hashing
  - via fully local extractable hashing



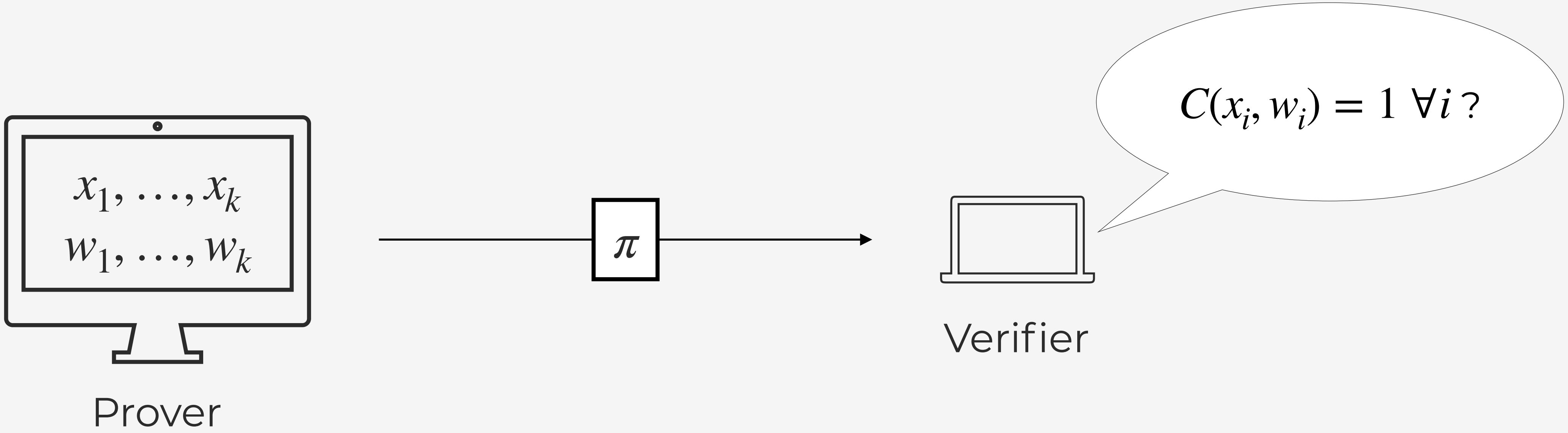
What are SNARGs?

- SNARG = **S**uccinct **N**on-interactive **ARG**ument
- Fix a NP **language** $\mathscr{L}_C$ with a relation circuit $C(x, w)$



- **Succinctness:** proof size and verification time are “small”, i.e. poly$(\lambda, \log |C|)$
- **Soundness:** if $x \notin \mathscr{L}_C$, the verifier rejects proofs from **PPT** adversaries
- No zero-knowledge (hiding) is required!

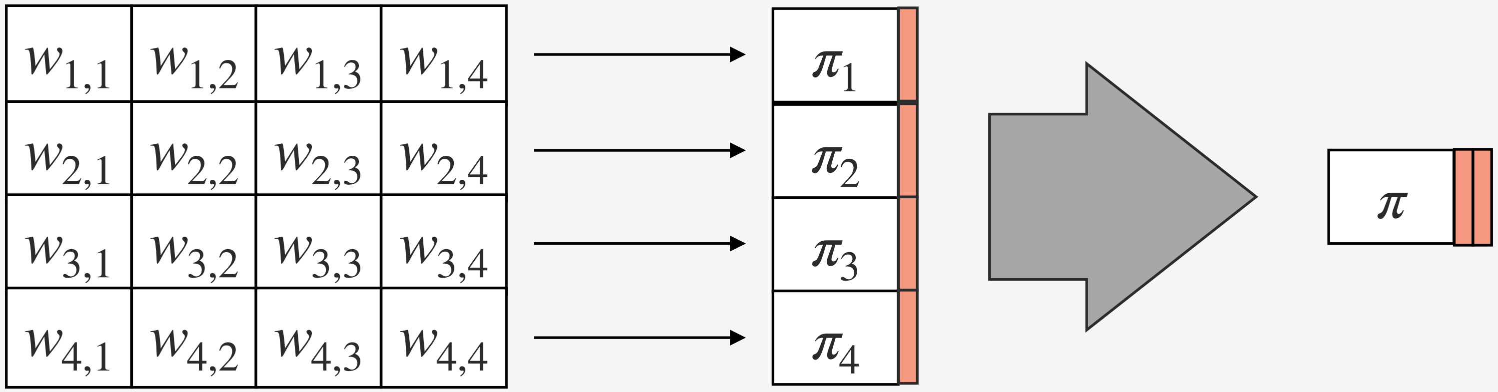
BARGs: SNARGs for batch-NP



- Naive solution: send all witnesses $w_1, \ldots, w_k$
- **Succinctness** requirement: $|\pi| < k \cdot |w_i|$, verifier is "fast". E.g. poly$(m, \log k)$
- **Soundness:** if even one statement is false, verifier rejects
- Remark: SNARGs for NP directly imply BARGs

Rate-1 BARGs

- A BARG is rate-1 if proof is as **small** as a single witness, i.e. $|\pi| = m + \text{poly}(\lambda)$
- We can do BARGs of BARGs of BARGs… for free!



Applications of rate-1 BARGs

- Multi-hop BARGs
- Incrementally Verifiable Computation
  - Strengthening of **delegation**/"SNARGs for P"
  - Compute and **update** proof of correctness of running computation
- Aggregate signatures
  - Generate a shorter **"digest" signature** from many different signatures
  - Also aggregation of aggregation

Constructions of rate-1 BARGs

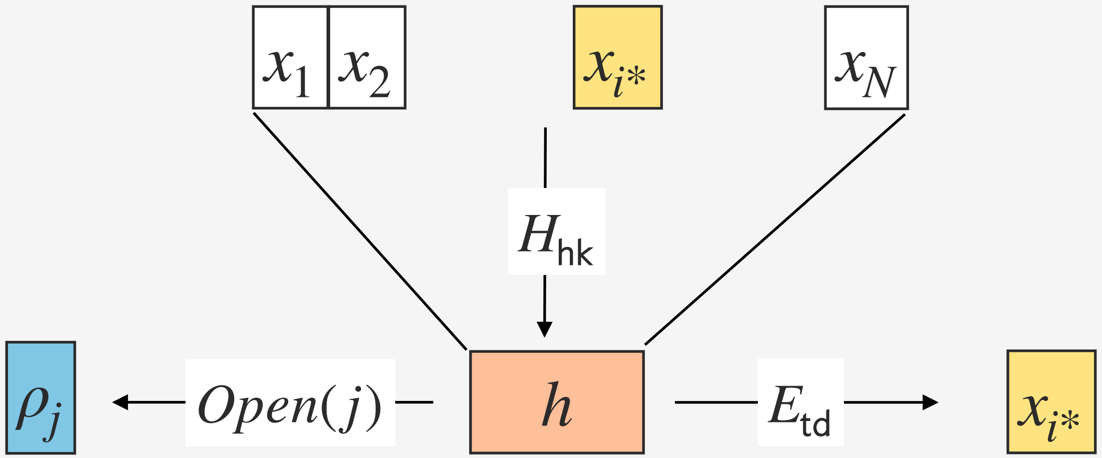
- PP22 constructs an almost rate-1 BARGs given **SEH** and **(index) BARGs**
  - Both can be instantiated from **LWE**
  - KLVW23+CGJ23 gives instantiation from DDH
- DGKV22 constructs rate-1 BARGs from **rate-1 flSEH**
  - They build their rate-1 flSEH from **LWE** (need FHE)

Fully local extractable hashing

Somewhere Extractable Hashing

- Strengthening of “statistically binding” notion by [HW15]
- We can **extract** bits of the input by **hiding trapdoors** in the hashing key
- They exist from “strong” primitives: FHE, rate-1 OT

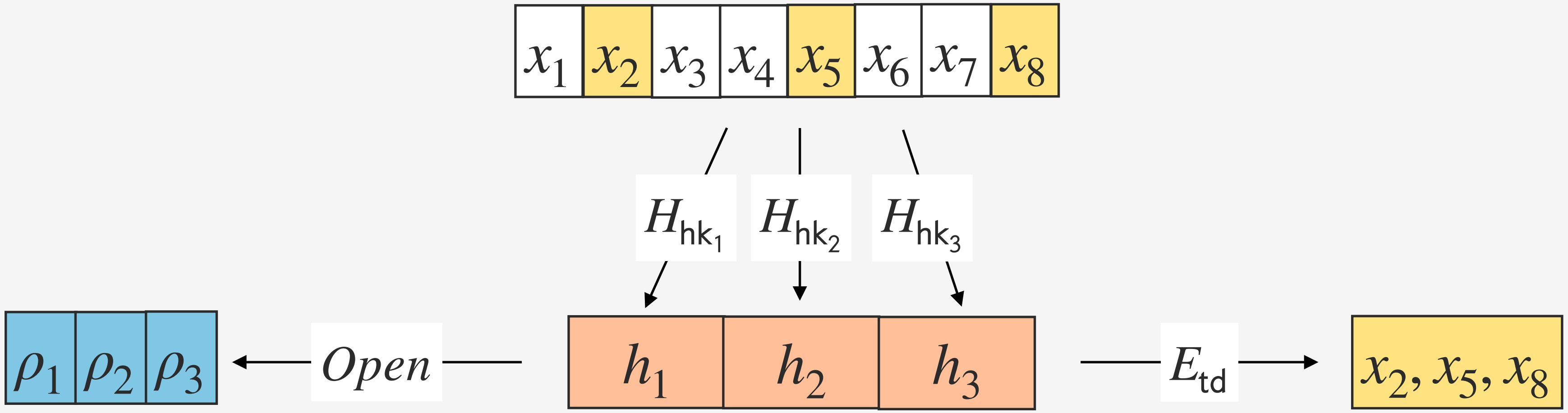
$$(\mathsf{hk}, \mathsf{td}) = Gen(i^*)$$



Multiple bits extraction

- We want to be extractable on a set of indices $I = \{i_1, \ldots, i_m\}$
- Remark: the total hash size has to be at least m
- In naive construction, also opening has size m

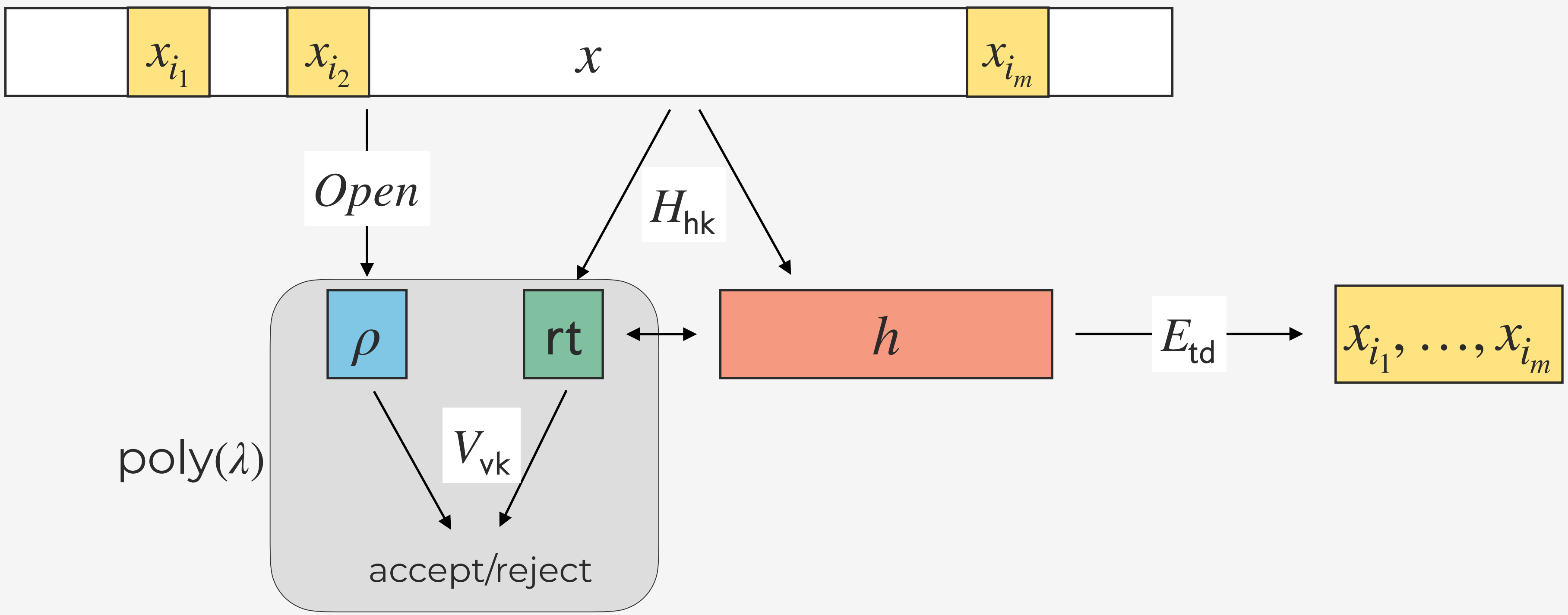
$$(\mathsf{hk}_1, \mathsf{td}_1) = Gen(i_1), \ldots, (\mathsf{hk}_m, \mathsf{td}_m) = Gen(i_m)$$



Why fully local?

- An opening must be **short**, and verifying it **fast**
- An additional "small digest" is needed

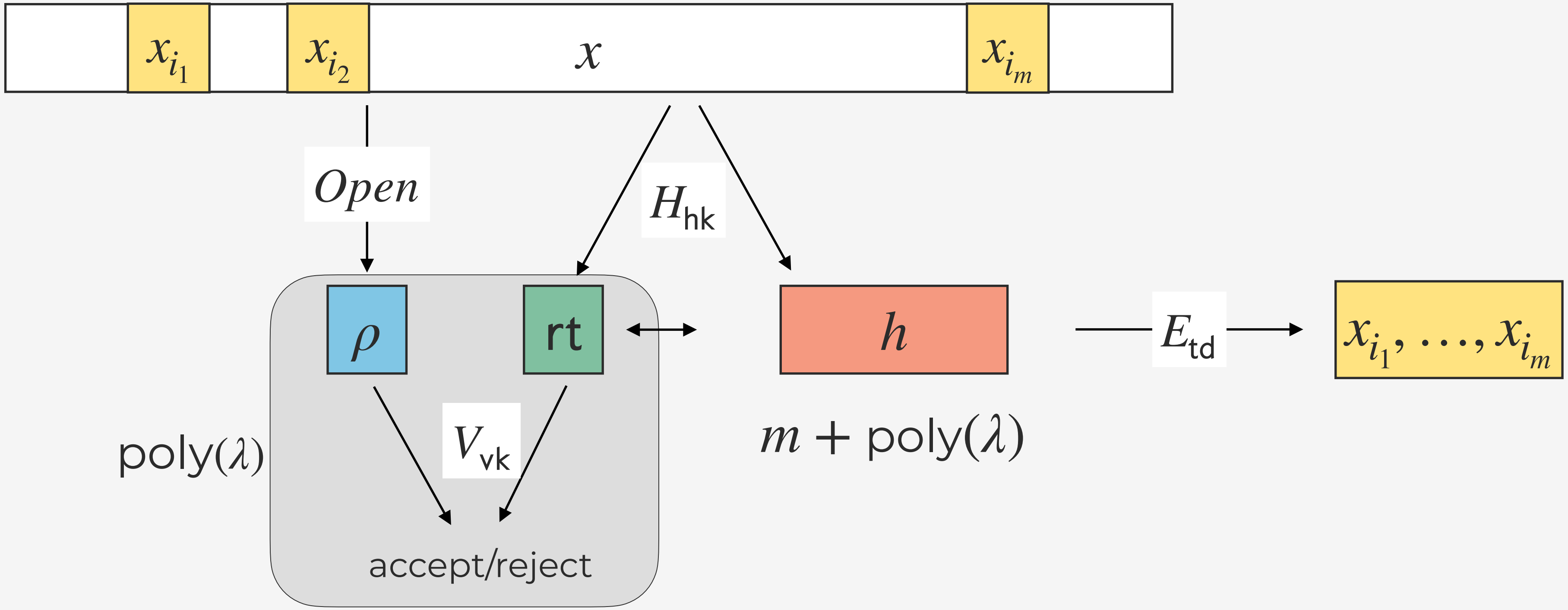
$$(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) = Gen(\{i_1, \ldots, i_m\})$$



Rate-1 flSEH

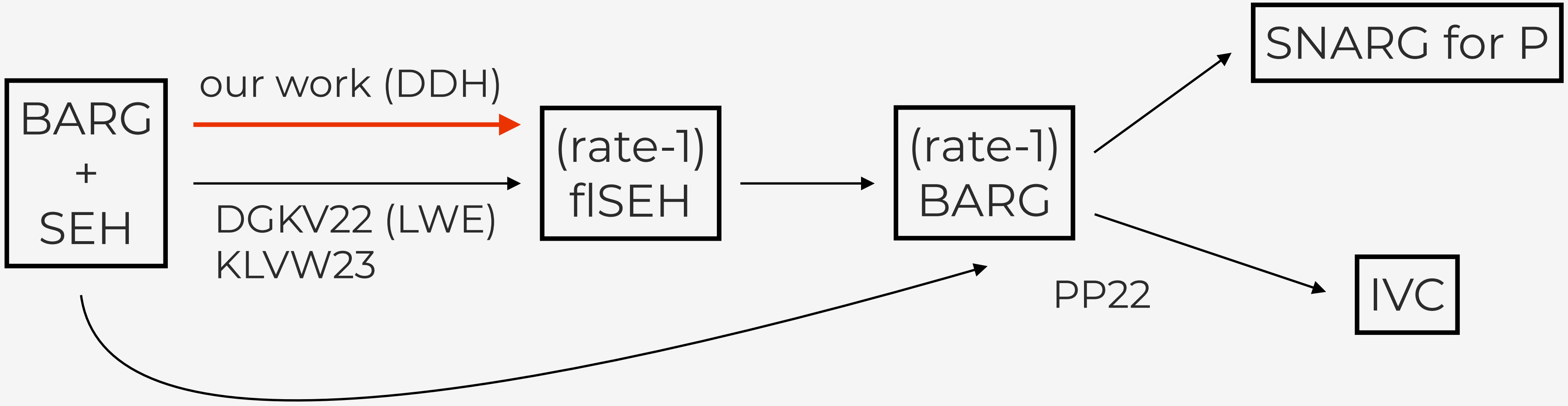
- We also want h to be as **short** as possible

$(\mathsf{hk}, \mathsf{vk}, \mathsf{td}) = Gen(\{i_1, \ldots, i_m\})$



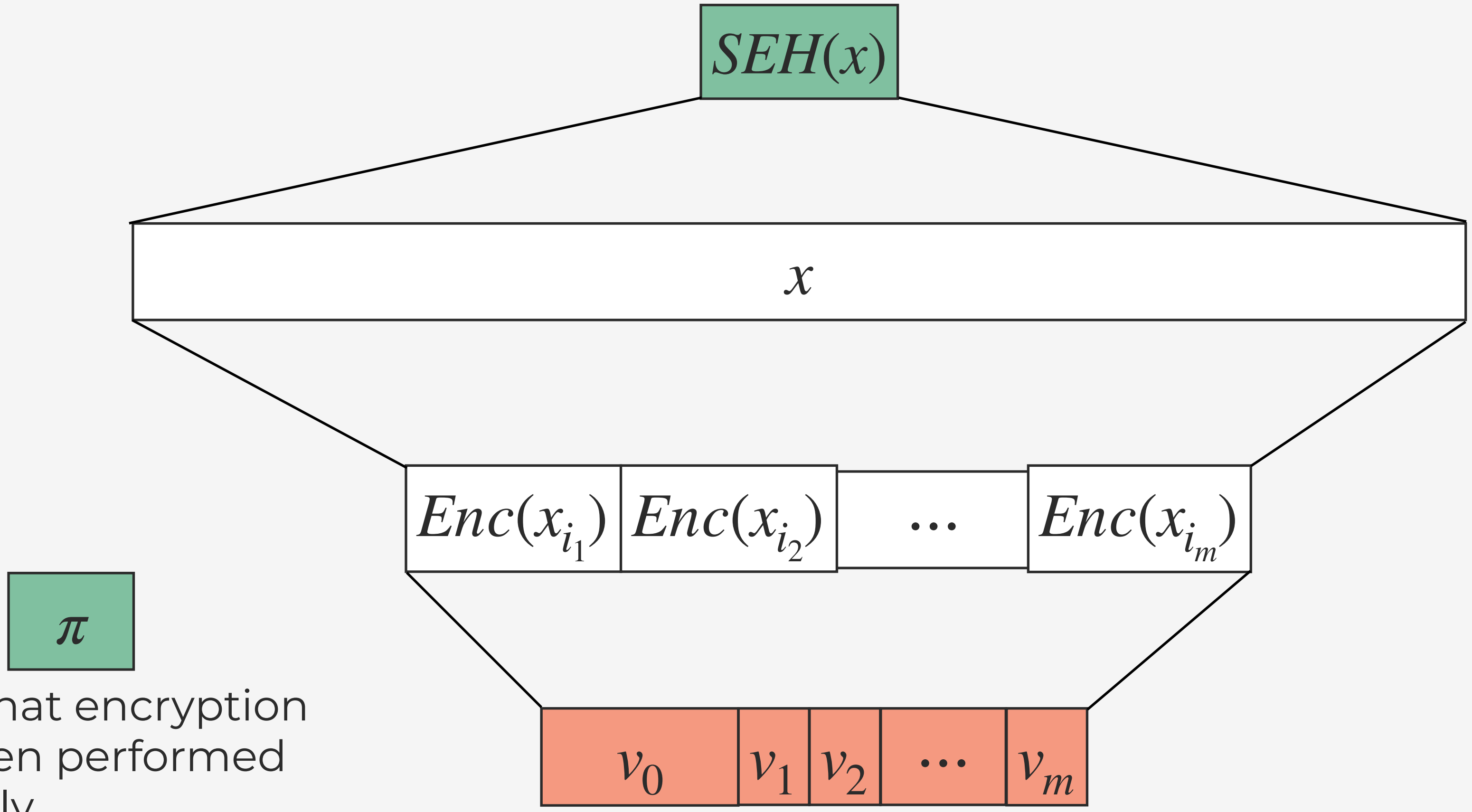
Our result: a rate-1 fully local SEH from DDH

- We get rate-1 flSEH from any BARG + SEH + DDH
  - From rate-1 flSEH we get rate-1 BARGs as in DGKV22
- PP22 can be instantiated with {DDH, LWE}, but has proof size $m + m/\lambda + poly(\lambda)$
- Caveat: our CRS is big, but rate-1 BARGs can **bootstrap** themselves



Our construction

Idea of construction



Proof that encryption has been performed correctly

ElGamal encryption

- Fix a prime order group G with generator g
- Public key is $h = g^a$ for some secret a
- To encrypt a **bit** x:
  - Compute random $c_0 = g^r$
  - Compute $c_1 = h^r \cdot g^x$
- This will allow for **compression** [BBDGM20], i.e. transmit (c_0, b) instead of (c_0, c_1), where b is a bit!

The hashing key

- Secret key/trapdoor is array of exponents $a = (a_1, \ldots, a_m)$
- Compute "public keys" $h_i = g^{a_i}$
- Sample random $r_1, \ldots, r_N$

$$M = \begin{pmatrix} g^{r_1} & g^{r_2} & \ldots & \ldots & g^{r_N} \\ h_1^{r_1} & \ldots & h_1^{r_{i_1}} g & \ldots & h_1^{r_N} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ h_m^{r_1} & \ldots & \ldots & h_m^{r_{im}} g & h_m^{r_N} \end{pmatrix}$$

Hashing

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix}$$

$$\begin{pmatrix} g^{r_1} & g^{r_2} & g^{r_3} & g^{r_4} \\ h_1^{r_1} & h_1^{r_2}g & h_1^{r_3} & h_1^{r_4} \\ h_2^{r_1} & h_2^{r_2} & h_2^{r_3}g & h_2^{r_4} \end{pmatrix} \longrightarrow \begin{pmatrix} g^{r_1x_1} & g^{r_2x_2} & g^{r_3x_3} & g^{r_4x_4} \\ h_1^{r_1x_1} & h_1^{r_2x_2}g^{x_2} & h_1^{r_3x_3} & h_1^{r_4x_4} \\ h_2^{r_1x_1} & h_2^{r_2x_2} & h_2^{r_3x_3}g^{x_3} & h_2^{r_4x_4} \end{pmatrix}$$

Hashing

$$\begin{pmatrix} g^{r_1 x_1} & g^{r_2 x_2} & g^{r_3 x_3} & g^{r_4 x_4} \\ h_1^{r_1 x_1} & h_1^{r_2 x_2} g^{x_2} & h_1^{r_3 x_3} & h_1^{r_4 x_4} \\ h_2^{r_1 x_1} & h_2^{r_2 x_2} & h_2^{r_3 x_3} g^{x_3} & h_2^{r_4 x_4} \end{pmatrix}$$

Hashing

$$\begin{pmatrix} g^{r_1x_1+r_2x_2} & g^{r_3x_3+r_4x_4} \\ h_1^{r_1x_1+r_2x_2}g^{x_2} & h_1^{r_3x_3+r_4x_4} \\ h_2^{r_1x_1+r_2x_2} & h_2^{r_3x_3+r_4x_4}g^{x_3} \end{pmatrix}$$

Hashing

$$\begin{pmatrix} g^{r_1x_1+r_2x_2+r_3x_3+r_4x_4} \\ h_1^{r_1x_1+r_2x_2+r_3x_3+r_4x_4} g^{x_2} \\ h_2^{r_1x_1+r_2x_2+r_3x_3+r_4x_4} g^{x_3} \end{pmatrix} = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix}$$

With trapdoor, we can **decrypt** $g^{x_{i_j}} = c_j \cdot c_0^{-a_j}$

Hashing

- We get batched ElGamal ciphertexts (at each step!)
- We then **compress** $(c_0, c_1, \ldots, c_m)$ into a rate-1 hash $v = (c_0, v_1, \ldots, v_m)$
- The **short digest** will be $h_x = SEH(x)$ and π
  - π is a **short** proof that v and h_x are **consistent**
  - It will be a BARG that the computation of v happened correctly

What do we BARG?

- We SE hash all the intermediate matrices $M_0, \ldots, M_T$ into $h_0, \ldots, h_T$
- For a given step t, the index BARG statement is

- The hash h_{t+1} opens to z at position (i, j)
- The hash h_t opens to z_1, z_2 at positions $(i, 2j), (i, 2j+1)$
- $z = z_1 \cdot z_2$

Analysis

- Short digest is **short**
  - Each hash (of x or matrices) needs to be binding on a **constant** number of positions. Their size is poly(λ).
  - All BARG proofs are also of size poly$(\lambda, \log N, \log m)$
- The extractable hash is **rate-1**
  - It's a compressed batch ElGamal ciphertext, of size $m +$ poly(λ)
- The CRS is a matrix of $(m + 1) \times N$ group elements, very **large**
  - We can **shorten** the CRS of resulting rate-1 BARG via recursion

Overview

- We build a **rate-1 fully local extractable hash** function from DDH
  - The extractable hash is built via series of **local computations**, allows us to BARG all parallel computation in single step
- We get truly **rate-1 BARGs**, again from DDH
  - We give a generic transformation from large CRS to short CRS

Thank you for the attention!

- Paper: https://ia.cr/2024/216
- Contact: riccardo.zanotto@cispa.de