

On the (in)security of ROS

Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, Mariana Raykova



On the (in)security of (some) blind signatures

Fabrice Benhamouda, Tancrede Lepoint, Julian Loss, Michele Orrù, Mariana Raykova



On the (in)security of (some) multi-signatures

Fabrice Benhamouda, Tancrede Lepoint, Julian Loss, Michele Orrù, Mariana Raykova



On the (in)security of (some) threshold signatures

Fabrice Benhamouda, Tancrede Lepoint, Julian Loss, Michele Orrù, Mariana Raykova



On the (in)security of (some) e-cash systems

Fabrice Benhamouda, Tancrede Lepoint, Julian Loss, Michele Orrù, Mariana Raykova



On the (in)security of (some) anonymous credentials

Fabrice Benhamouda, Tancrede Lepoint, Julian Loss, Michele Orrù, Mariana Raykova



ROS problem

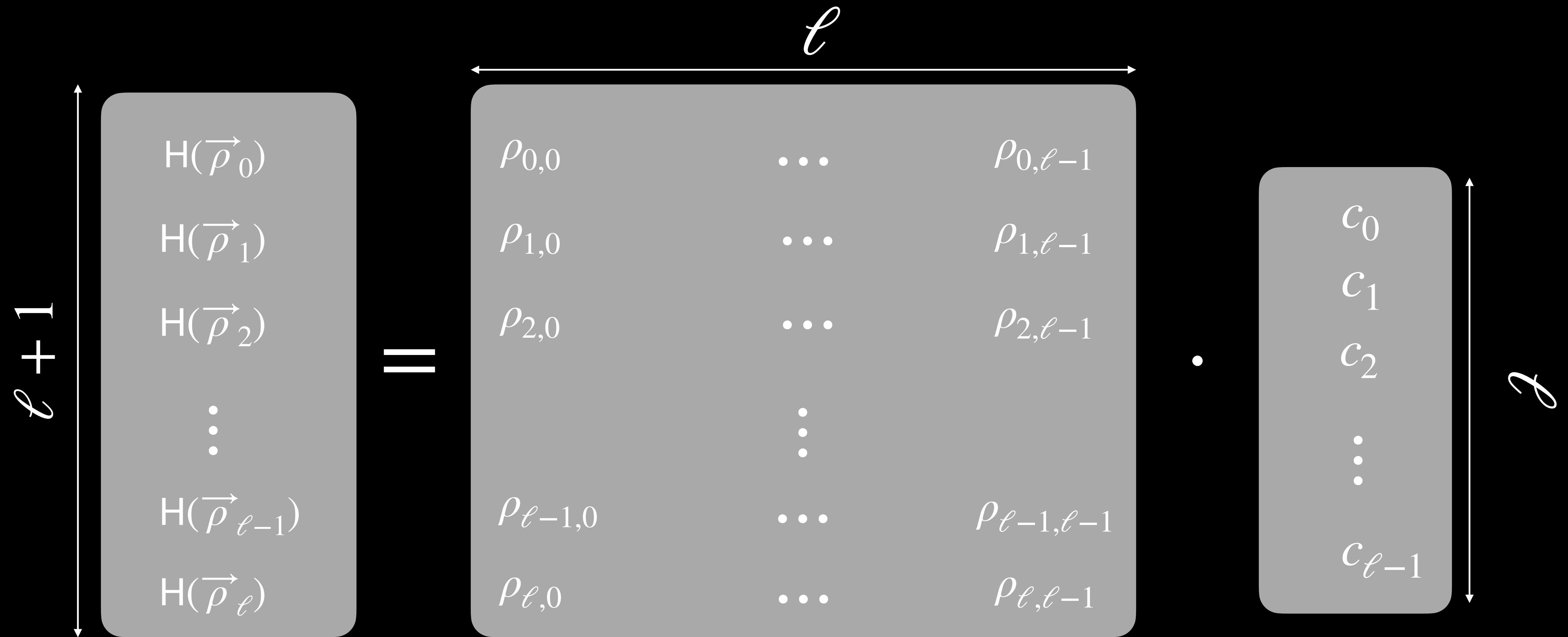
[Schnorr01]

Random inhomogeneities in a Overdetermined Solvable system of linear equations.

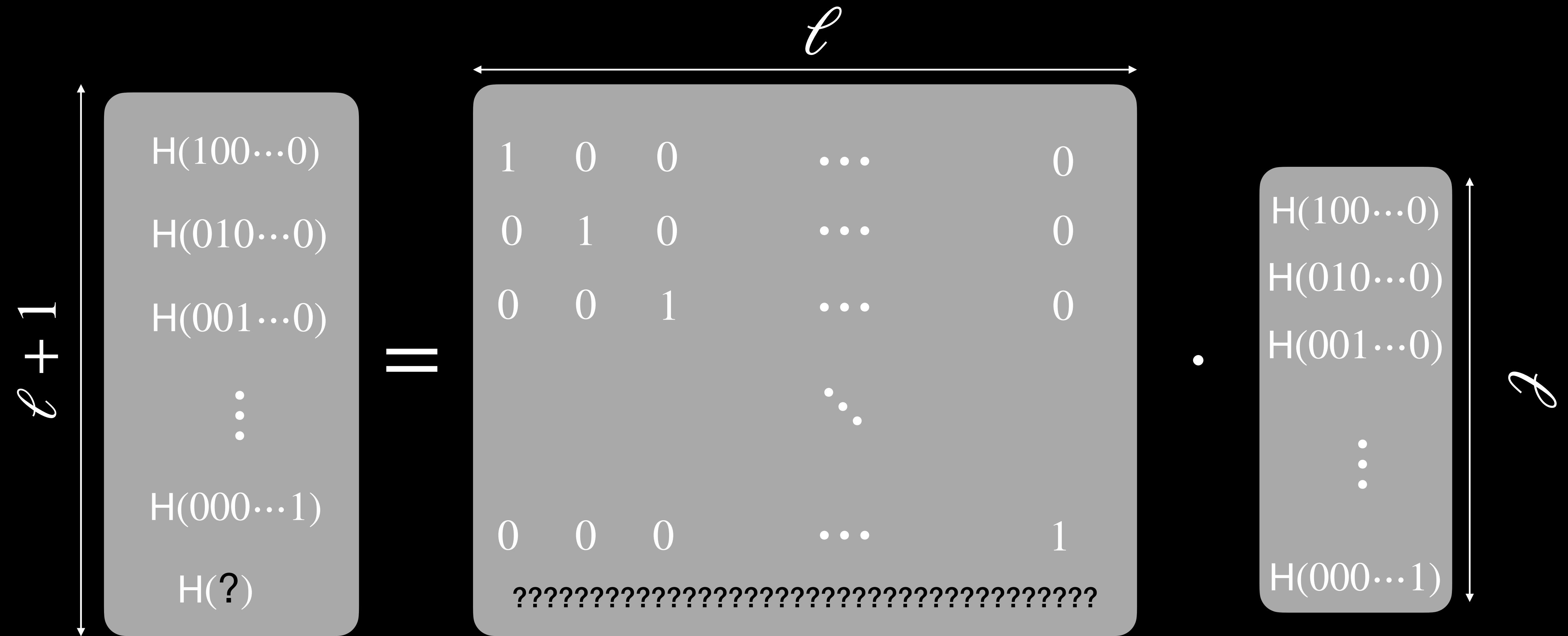
Find $\vec{c} \in \mathbb{F}_p^\ell$ and $\vec{\rho}_0, \dots, \vec{\rho}_\ell \in \mathbb{F}_p^\ell$ such that:

$$H(\vec{\rho}_j) = \langle \vec{\rho}_j, \vec{c} \rangle$$

ROS problem



ROS problem: partial solutions



ROS problem: partial solutions

$$\begin{array}{c|c|c} & \ell & \\ \ell+1 & \downarrow & \uparrow \\ \boxed{\begin{array}{l} \mathbb{H}(200\cdots 0) \\ \mathbb{H}(020\cdots 0) \\ \mathbb{H}(002\cdots 0) \\ \vdots \\ \mathbb{H}(000\cdots 2) \\ \mathbb{H}(?) \end{array}} & = & \boxed{\begin{array}{ccccc} 2 & 0 & 0 & \cdots & 0 \\ 0 & 2 & 0 & \cdots & 0 \\ 0 & 0 & 2 & \cdots & 0 \\ & & & \ddots & \\ 0 & 0 & 0 & \cdots & 2 \\ ????????????????????????????????????? \end{array}} & \cdot & \boxed{\begin{array}{l} ^{1/2}\mathbb{H}(200\cdots 0) \\ ^{1/2}\mathbb{H}(020\cdots 0) \\ ^{1/2}\mathbb{H}(002\cdots 0) \\ \vdots \\ ^{1/2}\mathbb{H}(000\cdots 2) \end{array}} \end{array}$$

ROS problem: previous results

[Wagner02]

- Generalised birthday attack:

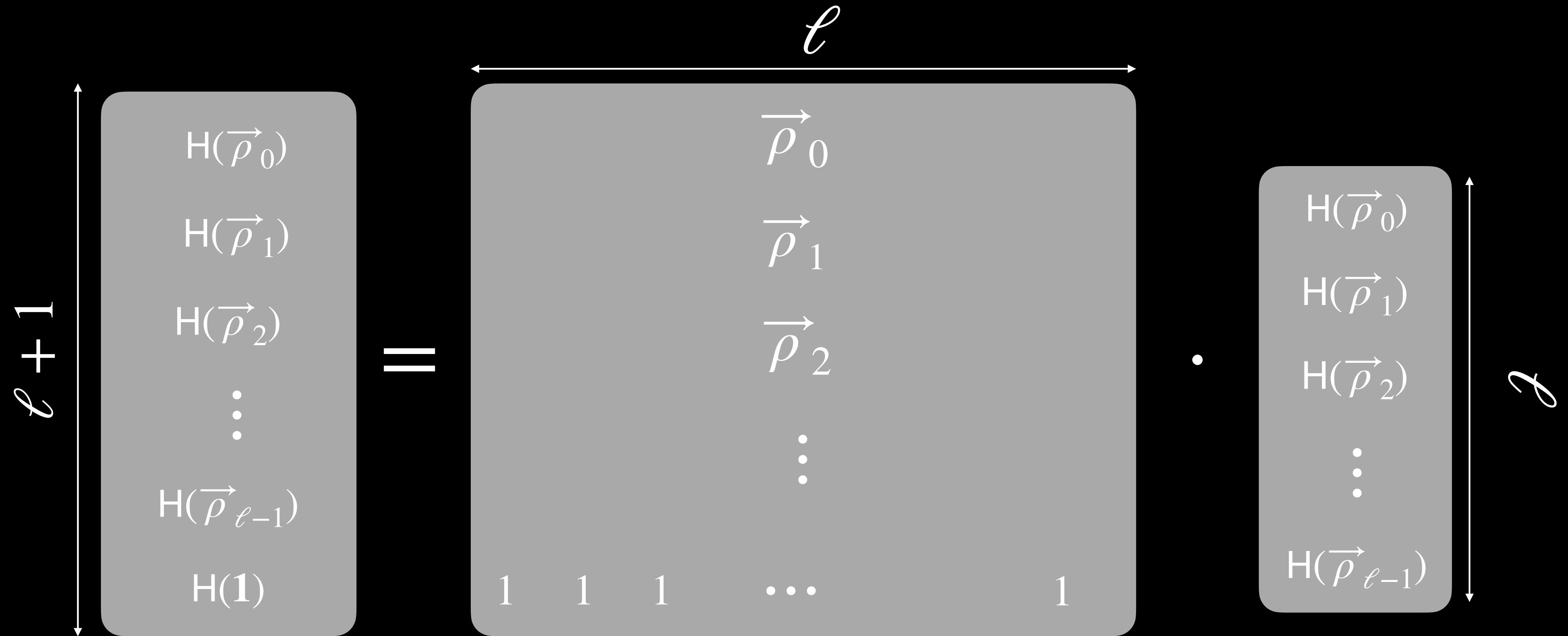
- Input lists L_0, \dots, L_ℓ with random elements
- Find elements in the list x_0, \dots, x_ℓ in the lists such that

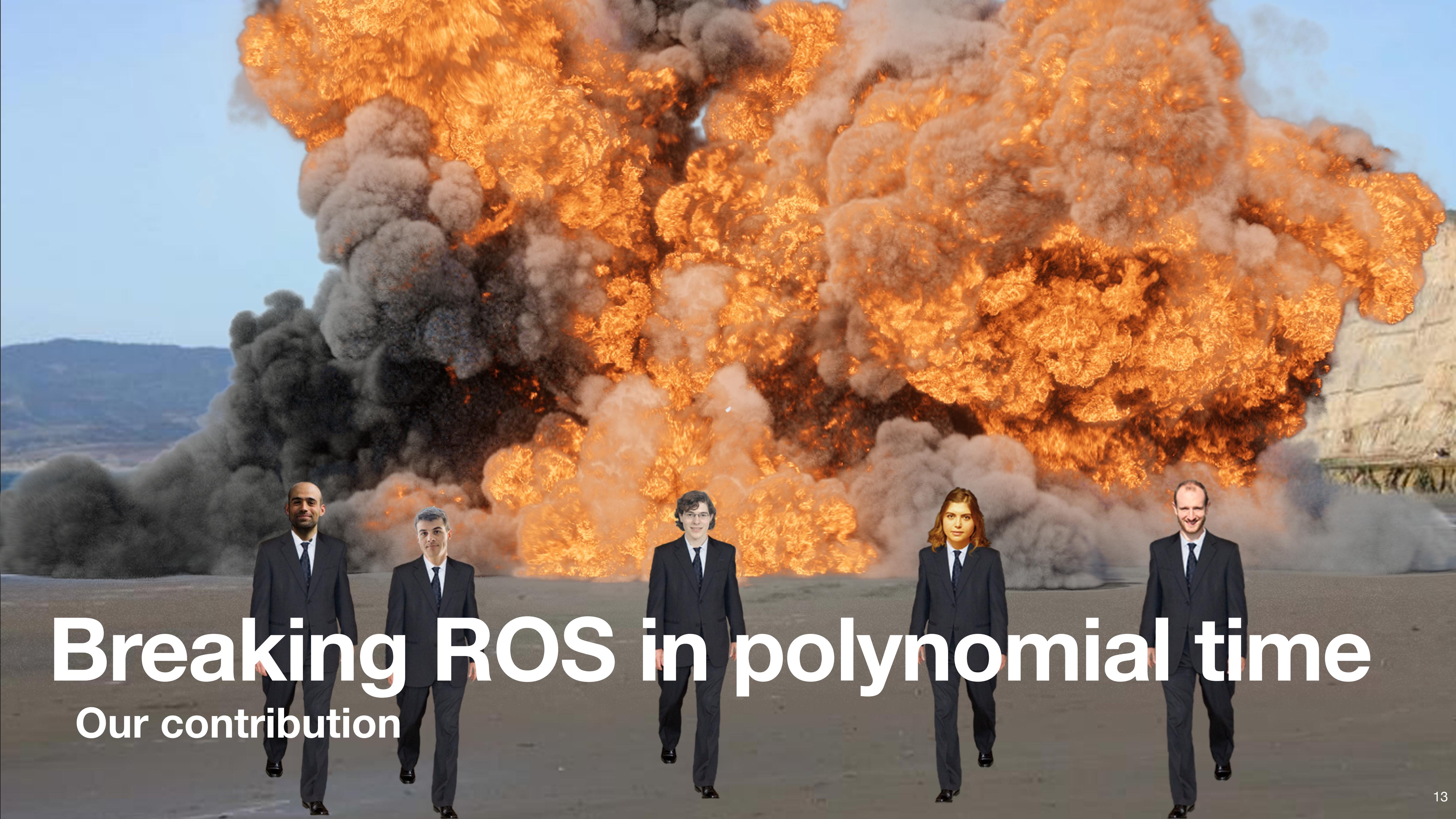
$$x_0 + \cdots + x_\ell = 0$$

→ For $\ell = 1$: birthday attack $O(\sqrt{p})$

→ Runs in time $O((\ell + 1) \cdot {}^{1 + \lfloor \log(\ell + 1) \rfloor} \sqrt{p})$.

Wagner's attack

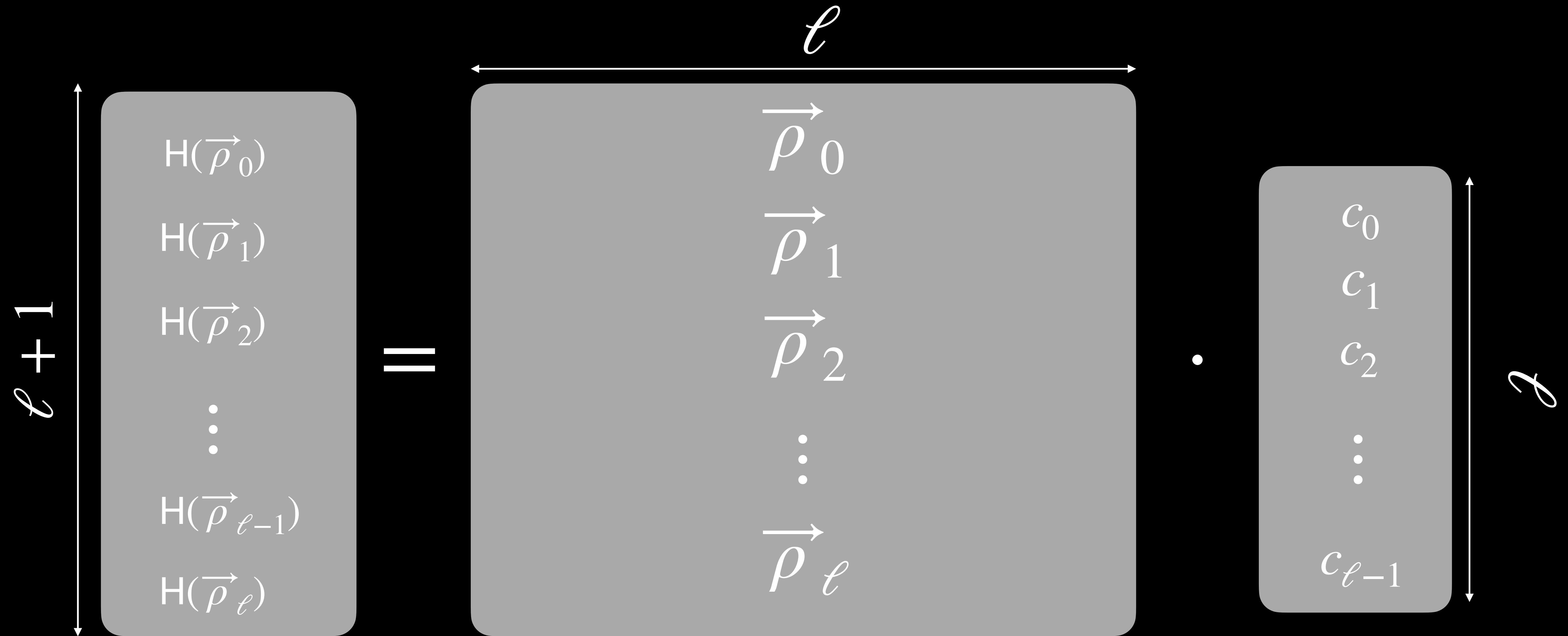




Breaking ROS in polynomial time

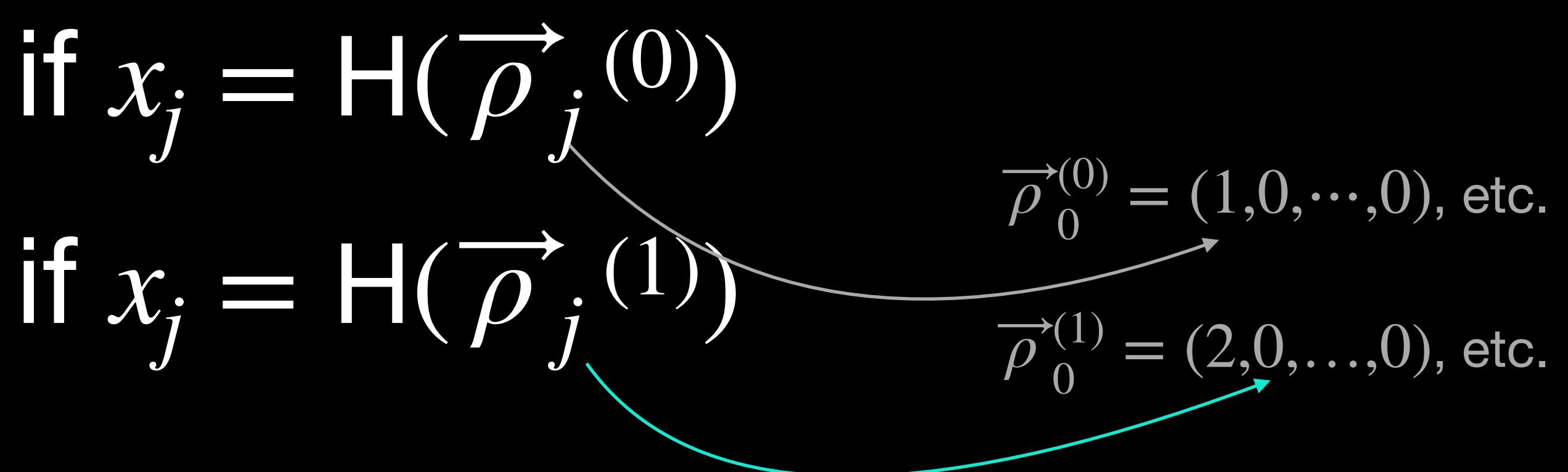
Our contribution

ROS Attack



ROS Attack: idea

For each $j \in \{0, \dots, \ell - 1\}$:

$$f_j(x_j) := \begin{cases} 0 & \text{if } x_j = H(\vec{\rho}_j^{(0)}) \\ 2^j & \text{if } x_j = H(\vec{\rho}_j^{(1)}) \end{cases}$$


$\vec{\rho}_0^{(0)} = (1,0,\dots,0)$, etc.

$\vec{\rho}_0^{(1)} = (2,0,\dots,0)$, etc.

** Skipping the constant factor in the coefficient for simplicity

ROS Attack: idea

For each $j \in \{0, \dots, \ell - 1\}$:

$$f_j(x_j) := 2^j \cdot \frac{x_j - H(\vec{\rho}_j^{(0)})}{H(\vec{\rho}_j^{(1)}) - H(\vec{\rho}_j^{(0)})}$$

ROS Attack: idea

For each $j \in \{0, \dots, \ell - 1\}$:

$$f_j(x_j) := 2^j \cdot \frac{x_j - c_j^{(1)}}{c_j^{(1)} - c_j^{(0)}}$$

ROS Attack: idea

$$\begin{aligned}\rho(x_0, \dots, x_{\ell-1}) &= f_0 + f_1 + \dots + f_{\ell-1} \\ &= \rho_0 x_0 + \dots + \rho_{\ell-1} x_{\ell-1} + \rho_\ell\end{aligned}$$

ROS Attack: idea

Let $\ell > \log p$. For any $n \in \mathbb{F}_p$:

$$\begin{aligned} n &= \sum_{j=0}^{\ell-1} 2^j \cdot b_j && [\text{Binary decomposition}] \\ &= \sum_{j=0}^{\ell-1} f_j(c_j^{(b_j)}) && [\text{By definition of } f_j] \\ &= \rho(c_0^{(b_0)}, \dots, c_{\ell-1}^{(b_{\ell-1})}) && [\text{Because } \rho(x_0, \dots, x_{\ell-1}) := f_0 + f_1 + \dots + f_{\ell-1}] \end{aligned}$$

ROS Attack: idea

Let $\ell > \log p$. For $n = H((\rho_0, \dots, \rho_{\ell-1})) + \rho_\ell$

$$n = \sum_{j=0}^{\ell-1} 2^j \cdot b_j \quad [\text{Binary decomposition}]$$

$$= \sum_{j=0}^{\ell-1} f_j(c_j^{(b_j)}) \quad [\text{By definition of } f_j]$$

$$= \rho(c_0^{(b_0)}, \dots, c_{\ell-1}^{(b_{\ell-1})}) \quad [\text{Because } \rho(x_0, \dots, x_{\ell-1}) := f_0 + f_1 + \dots + f_{\ell-1}]$$

$$\Rightarrow H((\rho_0, \dots, \rho_{\ell-1})) = \sum_{j=0}^{\ell-1} \rho_j c_j^{(b_j)}$$

ROS Attack: idea

Let $\ell > \log p$. For $n = H((\rho_0, \dots, \rho_{\ell-1})) + \rho_\ell$

$$n = \sum_{j=0}^{\ell-1} 2^j \cdot b_j \quad [\text{Binary decomposition}]$$

$$\begin{aligned} &= \sum_{j=0}^{\ell-1} f_j(c_j^{(b_j)}) \quad [\text{By definition of } f_j] \\ &= \rho(c_0^{(b_0)}, \dots, c_{\ell-1}^{(b_{\ell-1})}) \quad [\text{Because } \rho(x_0, \dots, x_{\ell-1}) := f_0 + f_1 + \dots + f_{\ell-1}] \end{aligned}$$

$$\Rightarrow H((\rho_0, \dots, \rho_{\ell-1})) = \sum_{j=0}^{\ell-1} \rho_j c_j^{(b_j)} = \langle (\rho_0, \dots, \rho_{\ell-1}), \vec{c} \rangle \blacksquare$$

Attack summary

- Pick basis. $\vec{\rho}_0^{(0)}, \dots, \vec{\rho}_{\ell-1}^{(0)}$ and $\vec{\rho}_0^{(1)}, \dots, \vec{\rho}_{\ell-1}^{(1)}$
- Hash'em. $c_j^{(b)} := H(\vec{\rho}_j^{(b)})$.
- Interpolate and compute $\rho(x_0, \dots, x_{\ell-1})$
- Decompose $H((\rho_0, \dots, \rho_{\ell-1})) + \rho_\ell = (b_0, \dots, b_{\ell-1})_2$.

Nontrivial solution: $(\rho_0, \dots, \rho_{\ell-1})$

Easy solutions: $\vec{\rho}_0^{(b_0)}, \dots, \vec{\rho}_{\ell-1}^{(b_{\ell-1})}$

Attack summary - generalised

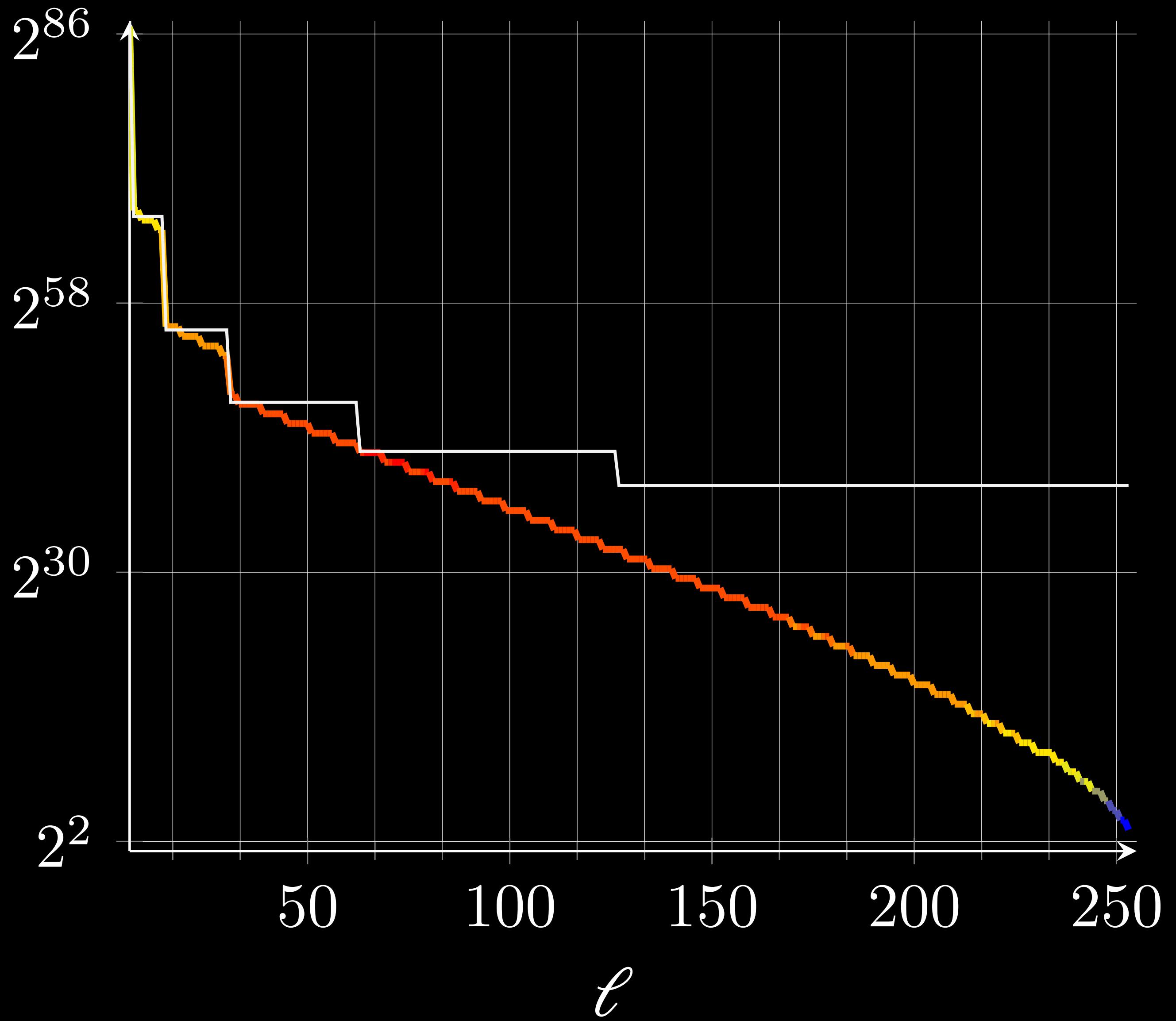
- Pick basis. $\vec{\rho}_0^{(0)}, \dots, \vec{\rho}_{\ell-1}^{(0)}$ and $\vec{\rho}_0^{(1)}, \dots, \vec{\rho}_{\ell-1}^{(1)}$
- Hash'em. $c_j^{(b)} := H(\vec{\rho}_j^{(b)})$.
- Interpolate and compute $\rho(x_0, \dots, x_{\ell-1})$
- Decompose $H((\rho_0, \dots, \rho_{\ell-1})) + \rho_\ell = (b_0, \dots, b_{\ell-4}, b_{\ell-3}, b_{\ell-2}, b_{\ell-1})_2$.

Attack summary - generalised

- Pick basis. $\vec{\rho}_0^{(0)}, \dots, \vec{\rho}_{\ell-1}^{(0)}$ and $\vec{\rho}_0^{(1)}, \dots, \vec{\rho}_{\ell-1}^{(1)}$
- Hash'em. $c_j^{(b)} := H(\vec{\rho}_j^{(b)})$.
- Interpolate and compute $\rho(x_0, \dots, x_{\ell-1})$
- Decompose $H((\rho_0, \dots, \rho_{\ell-1})) + \rho_\ell = (b_0, \dots, b_{\ell-4}, b_{\ell-3}, 0, 0)_2$.

Complexity

Attack cost

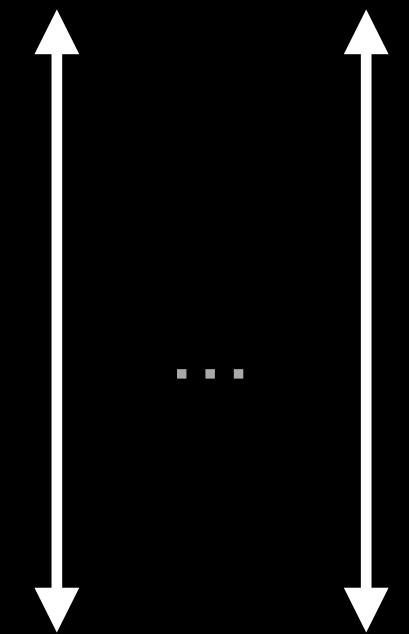


Attacking Blind signatures

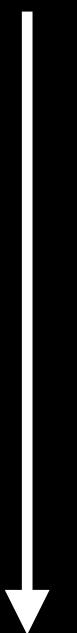
Blind signatures

Unforgeability

ℓ interactions



\mathcal{A}



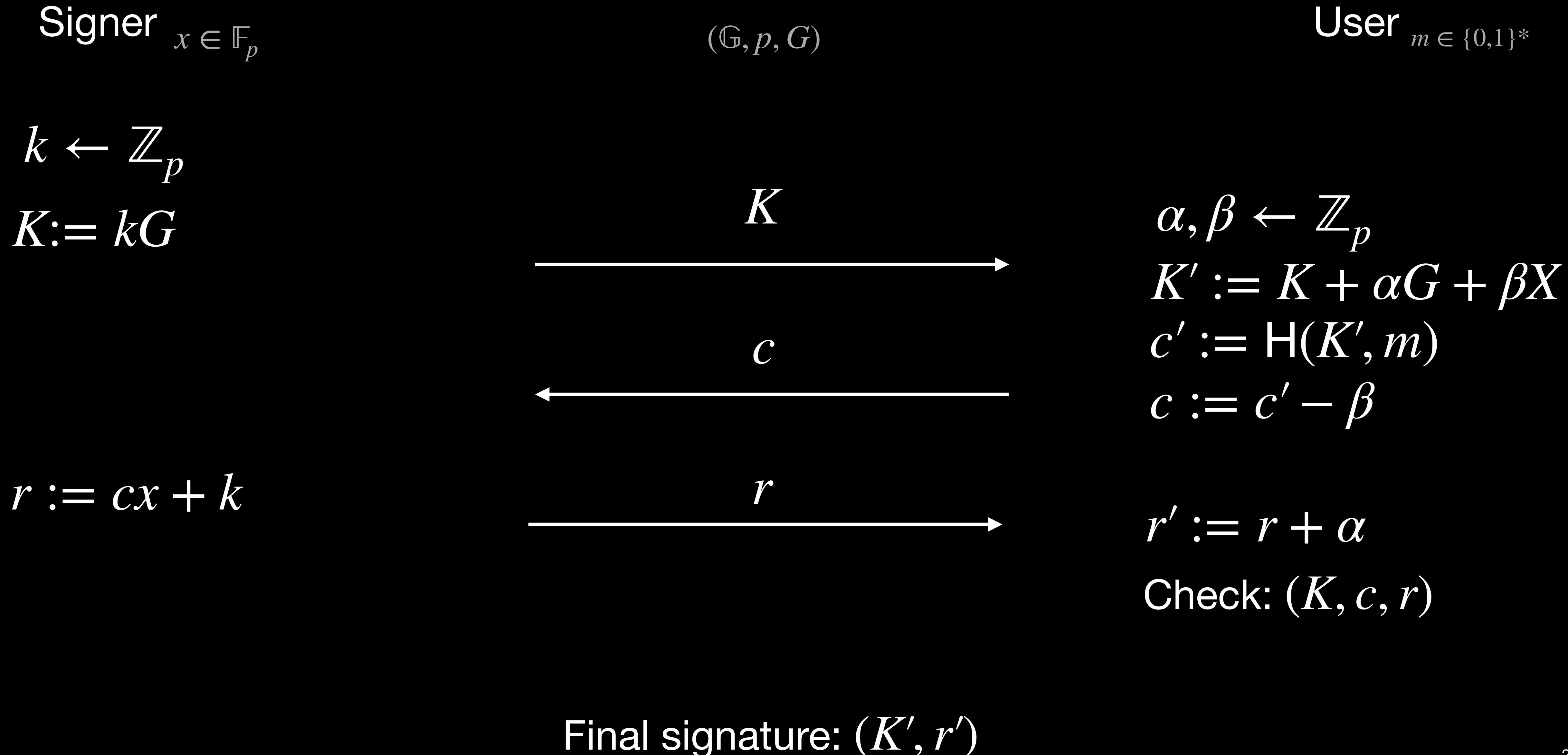
$\ell + 1$ signatures

Blind signatures

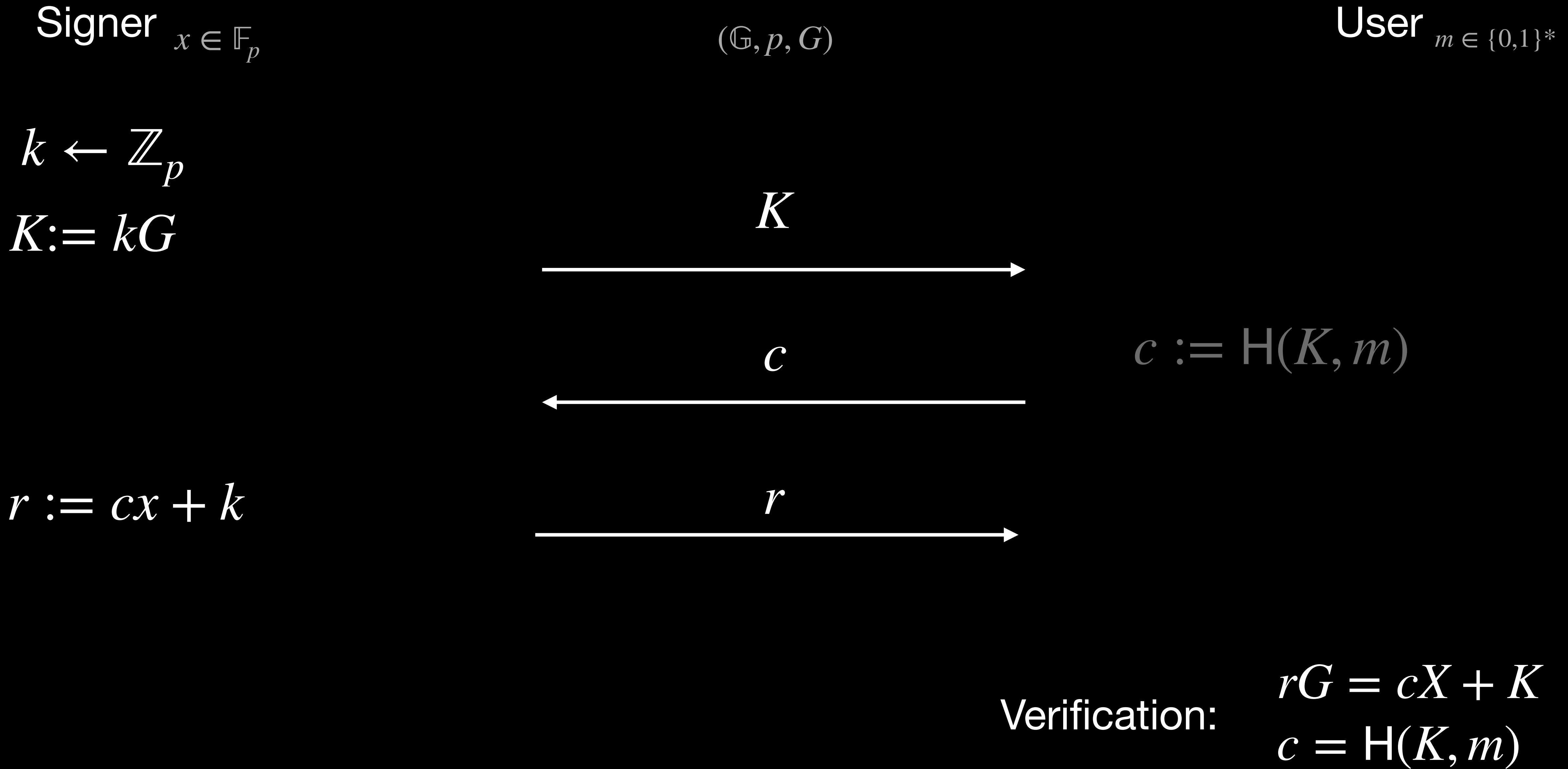
Blindness

[not covered]

Schnorr blind signature

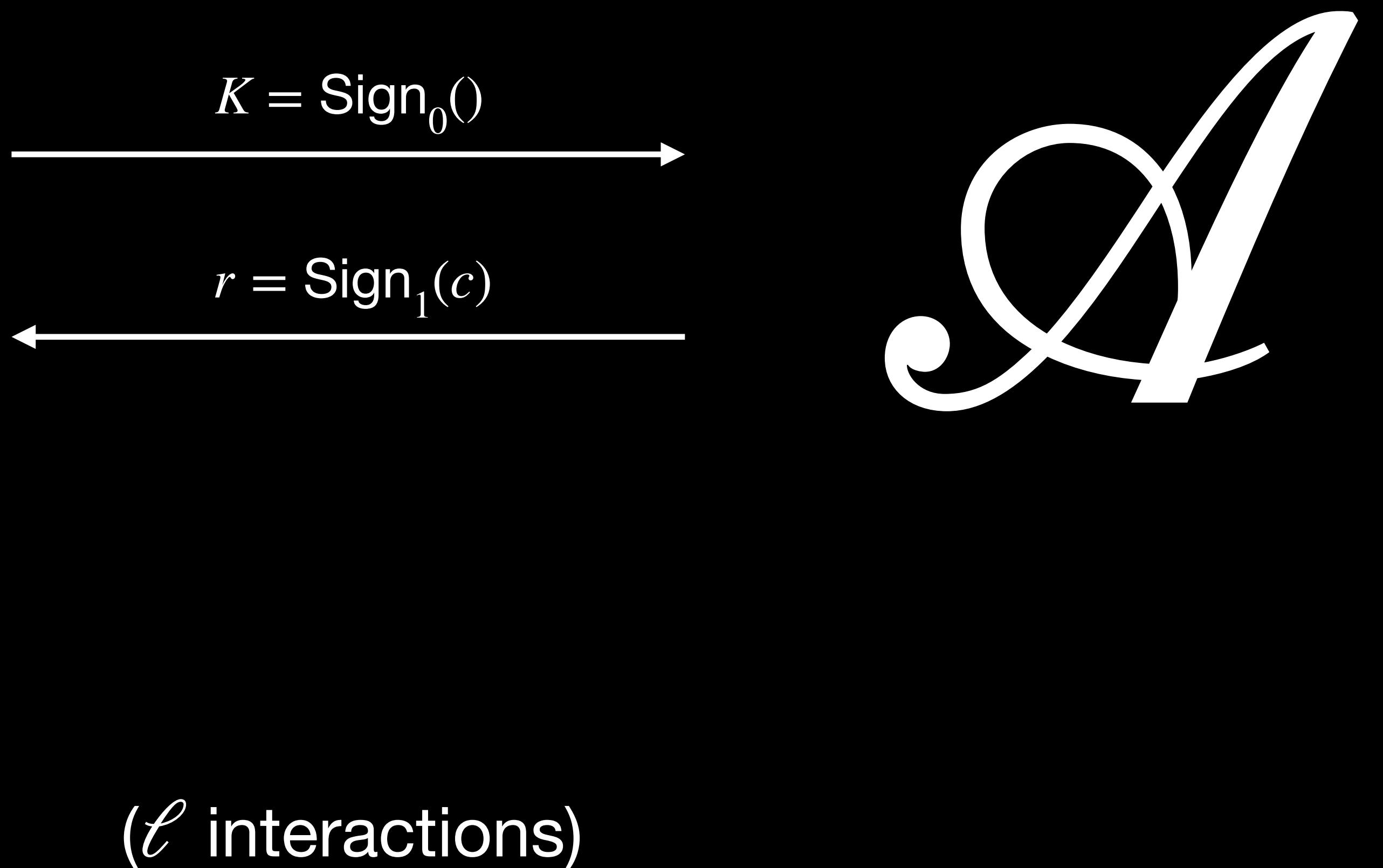


Schnorr blind signature



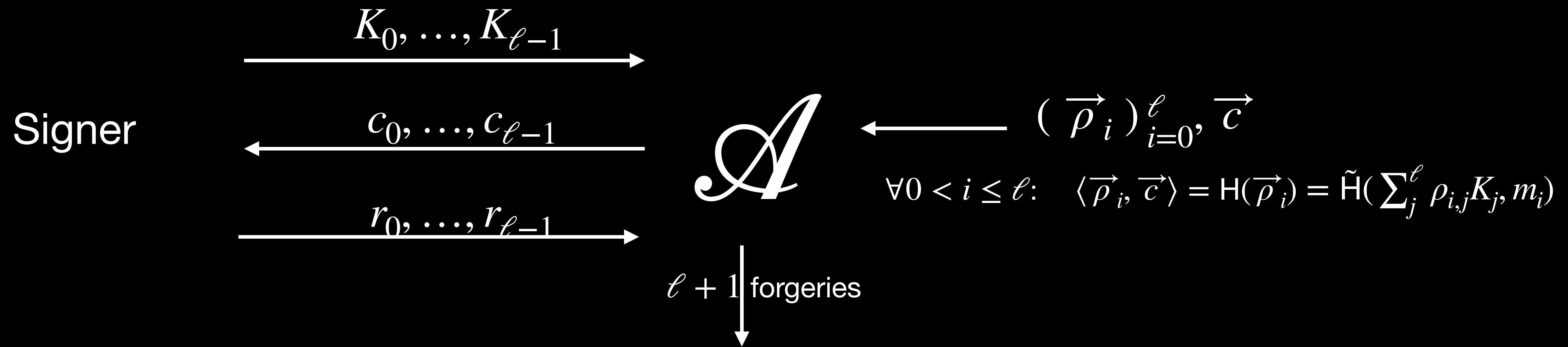
Schnorr blind signatures

One-more unforgeability



Attack on Schnorr blind signatures

[Schnorr01]



$$K_i^\star = \sum_{j=0}^{\ell-1} \rho_{i,j} \cdot K_j$$

$$c_i^\star = \sum_{j=0}^{\ell-1} \rho_{i,j} \cdot c_j$$

$$r_i^\star = \sum_{j=0}^{\ell-1} \rho_{i,j} \cdot r_j$$

$$r_i^\star G = \sum_j \rho_{i,j} \cdot r_j G = \sum_j \rho_{i,j} \cdot (c_j X + K_j) = c_i^\star X + K_i^\star$$

How practical? 55LOC.

```
1 # public parameters: secp256k1
2 Zq = GF(0xfffffffffffffffffffffffffffff...fffffefffffc2f)
3 E = EllipticCurve(Zq, [0, 7])
4 G = E.lift_x(0x79be667ef9dcbbac55a06295ce870b07029bfedb2dce28d959f2815b16f81798)
5 p = G.order()
6 Zp = GF(p)
7
8 def random_oracle(hinput, _table=dict()):
9     if hinput not in _table:
10         _table[hinput] = Zp.random_element()
11     return _table[hinput]
12
13 def verify(message, K, e, s):
14     assert random_oracle((K, message)) == e, "random oracle fails"
15     assert G * int(s) + X * int(e) == K, "verification equation fails"
16     return True
17
18 def inner_product(coefficients, values):
19     return sum(y*int(x) for x, y in zip(coefficients, values))
20
21 # server: generate public key
22 x = Zp.random_element()
23 X = G * int(x)
24
25 # adversary: open `ell` sessions
26 ell = 256
27
28 # server: generate commitments
29 k = [Zp.random_element() for i in range(ell)]
30 K = [G * int(k_i) for k_i in k]
31
32 # adversary: generate challenges
33 e = [[random_oracle((K_i, b)) for b in range(2)] for K_i in K]
34 P = ([-sum([Zp(2)^i * e[i][0]/(e[i][1] - e[i][0]) for i in range(ell)])] +
35      [Zp(2)^i / (e[i][1] - e[i][0]) for i in range(ell)])
36
37 forged_K = inner_product(P, [G+X] + K)
38 forged_message = "message"
39 forged_e = random_oracle((forged_K, forged_message))
40 bits = [int(b) for b in bin(forged_e)[2:].rjust(256, '0')][::-1]
41 chosen_e = [e[i][b] for (i, b) in enumerate(bits)]
42
43 # server: generate the responses
44 s = [k[i] - chosen_e[i]*x for i in range(ell)]
45
46 # attacker: generate the forged response
47 forged_s = inner_product(P, [1] + s)
48
49 ## check all previous signatures were valid
50 print(all(
51     # l signatures generated honestly
52     [verify(m_i, K_i, e_i, s_i) for (m_i, K_i, e_i, s_i) in zip(bits, K, chosen_e, s)] +
53     # final signature
54     [verify(forged_message, forged_K, forged_e, forged_s)])
55 ))
```

**Do not use blind
Schnorr signatures.**

Still in the game: blind RSA, Blind BLS, Abe blind signatures, clause Schnorr blind signatures.