Zero Knowledge Salon

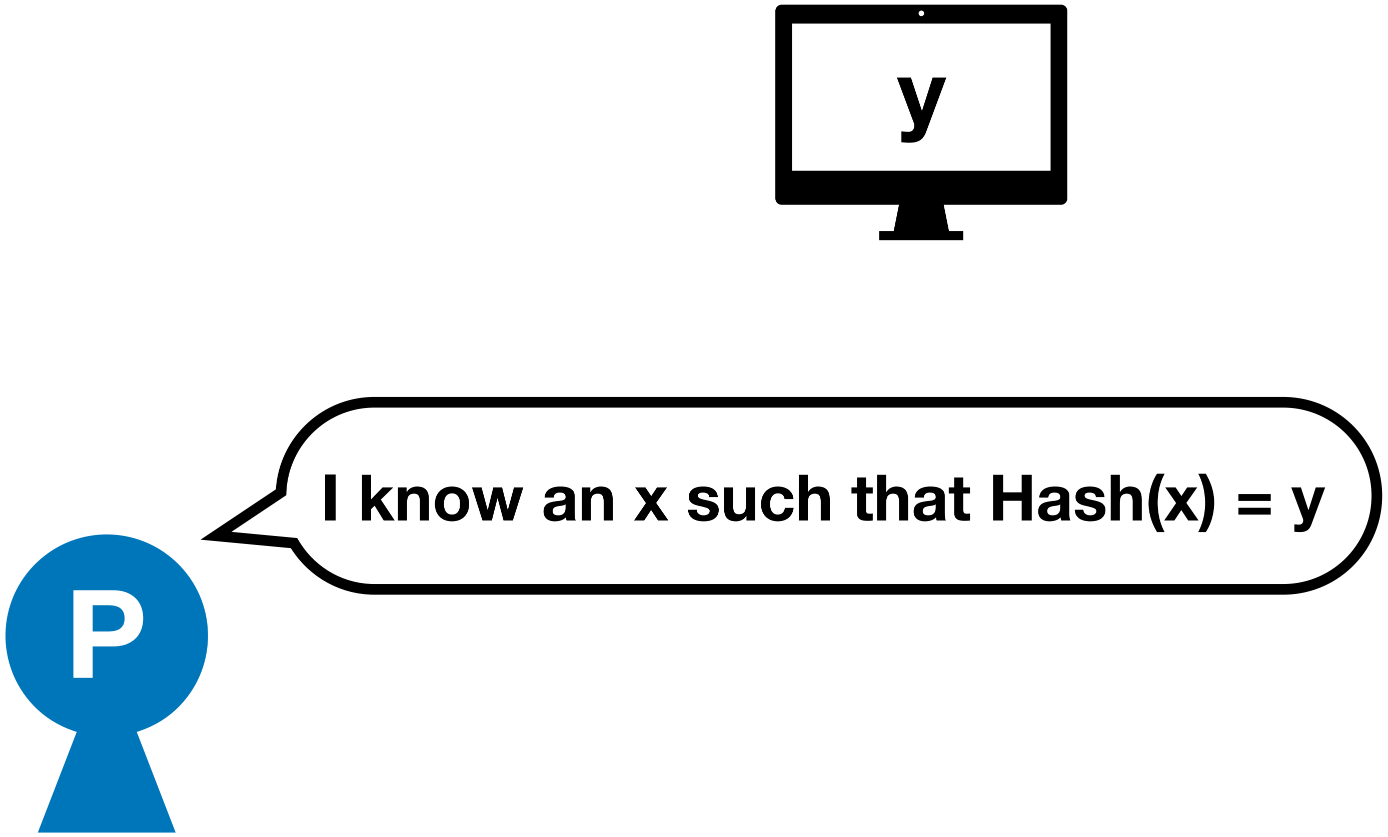
Po-Chun Kuo and Hao Chung
Oct 31, 2018

Outline

1. Intro to zero knowledge
2. Zcash
3. Zero set
4. Bullet proof
5. zk-SNARK

Zero Knowledge

Prover wants to convince Verifier that he/she knows a statement *x* without revealing further information.

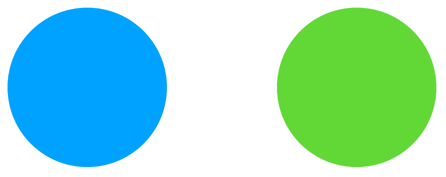


Example

Problem setup:

- Two characters: Prover and Verifier (verifier is blind)
- They have two balls: the one is blue, the other is green
    - The two balls are all identical except the colors
- Prover wants to show that “I know the fact that they are in different color.”

Proof system:



1. Verifier places two balls behind his back.
2. Verifier takes one of the balls and displays it.
3. Verifier places the ball behind his back.
4. Verifier switch the ball with probability 50% and displays the ball again. Ask “Do I switch the ball?”

Zero Knowledge Proof

A zero knowledge proof should satisfy three conditions:

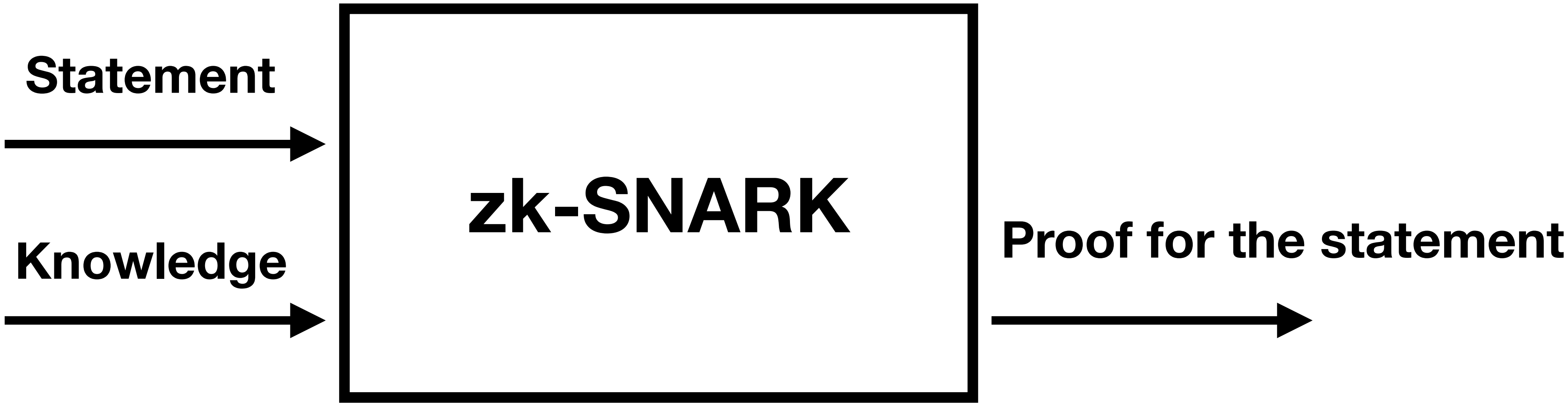
- **Completeness**: if the statement is true, verifier should accept the proof
- **Soundness**: if the statement is false, it should be infeasible that verifier accept the proof
- **Zero knowledge**: Verifier should not learn any information beyond that the statement is true.

The Privacy in the cryptocurrency

In cryptocurrency system, there are two properties that can be protected:

1. The identity of the sender and the receiver
2. The amount of the transaction

zk-SNARK

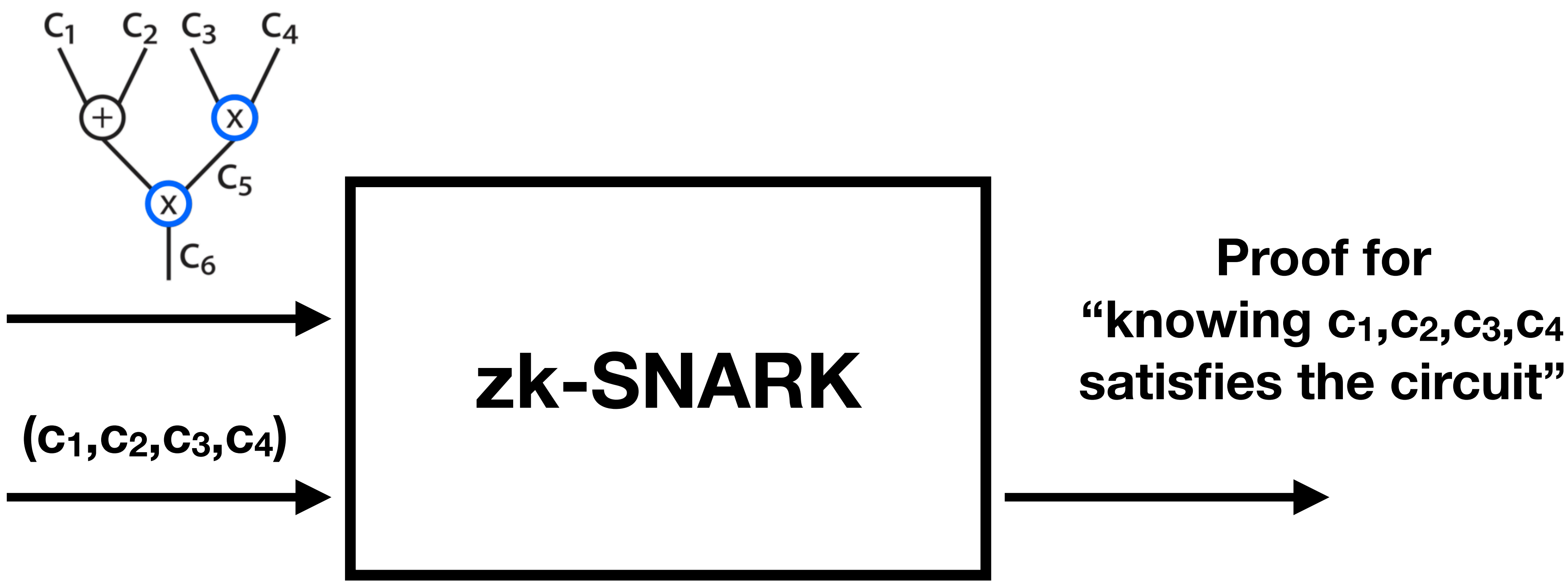


zk-SNARK

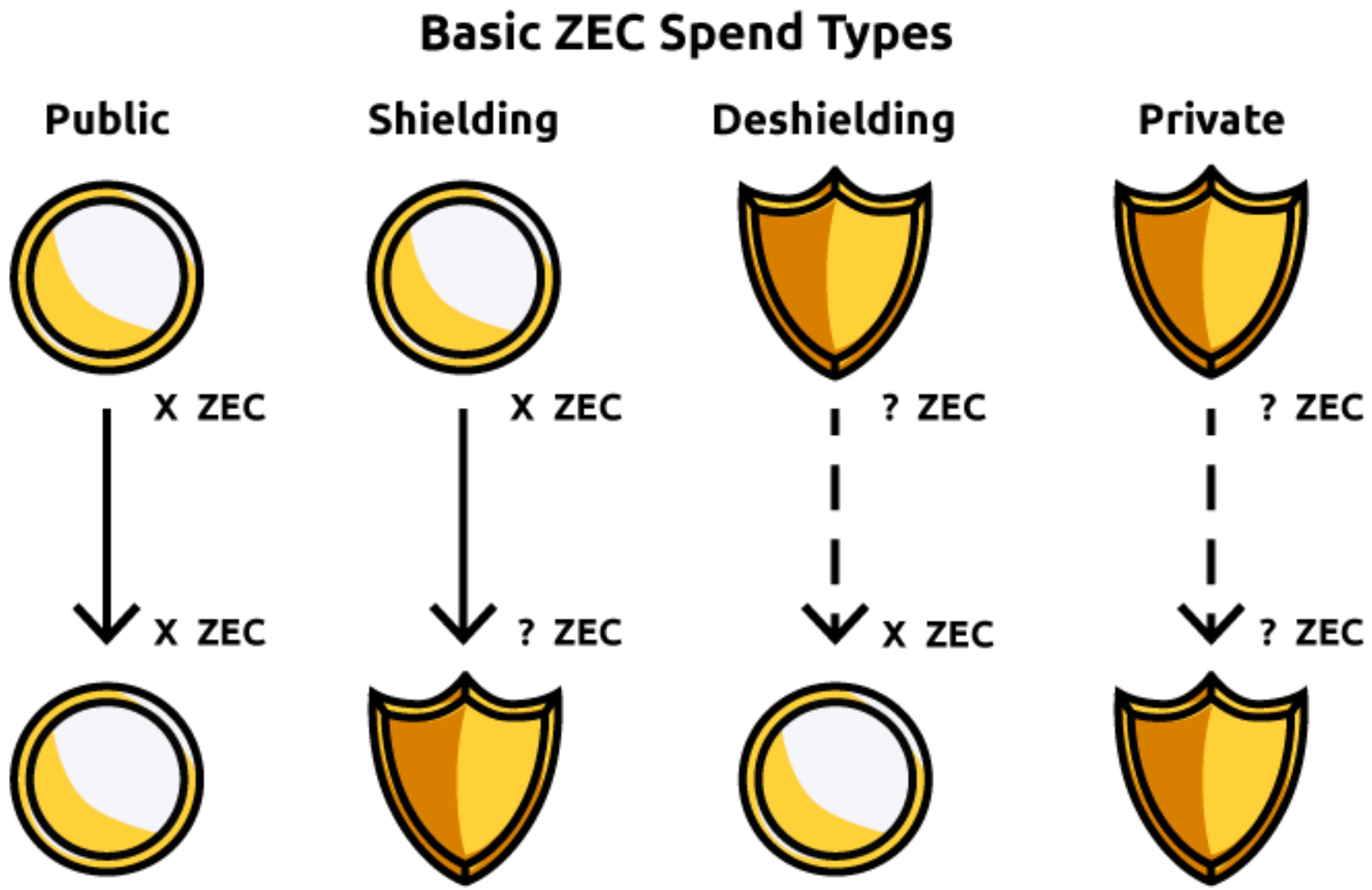
For example, Alice wants to show that she knows x such that y = Hash(x).

zk-SNARK

For example, Alice wants to show that she knows an assignment $(c_1,c_2,\ldots,c_n)$ satisfies the circuit.



Four types of modes



Zcash

- In Bitcoin system, the inputs of a transaction are many UTXOs (unspent transaction outputs).
- In Zcash, an UTXO can be thought as an unspent note:

$$Note_1 = (pk_1, v_1, r_1)$$

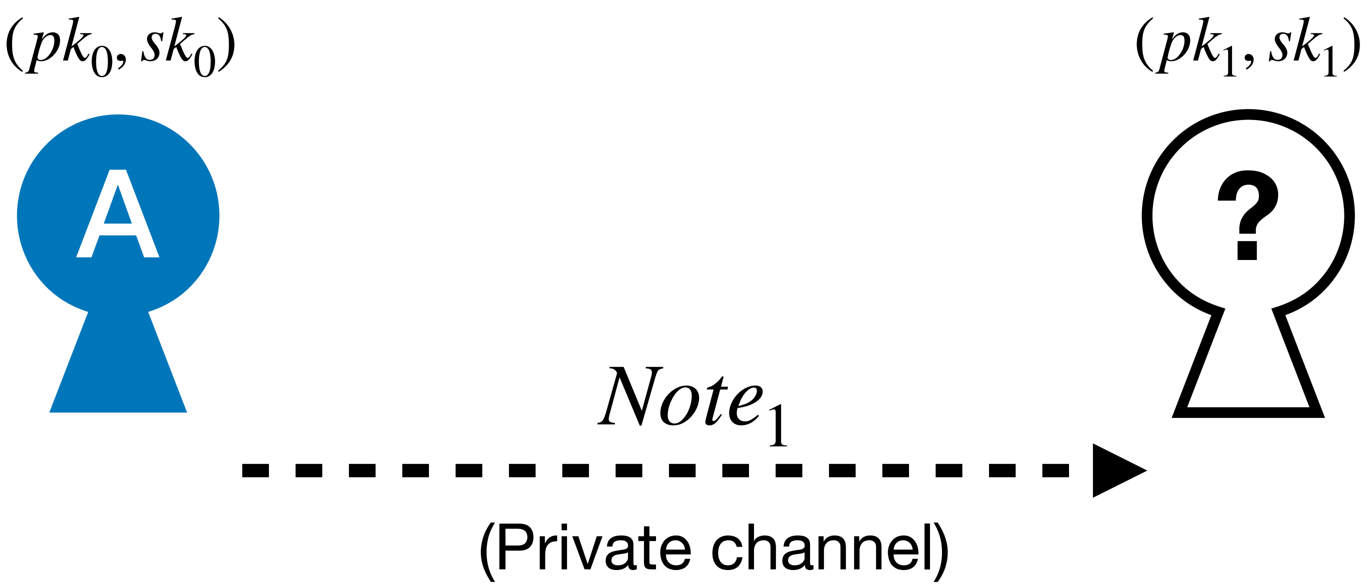
v: amount of cash
r: randomness of the note

- In order to protect the privacy, the note is published in “hash” format

$$C_1 = Hash(Note_1)$$

Shielding

Suppose Alice has a set of UTXOs whose balance sum to v
Now Alice wants to “mint” a black coin.



1. Alice creates a new note $Note_1 = (pk_1, v_1, r_1)$.
2. Alice announces the commitment of the note $C_1 = Hash(Note_1)$

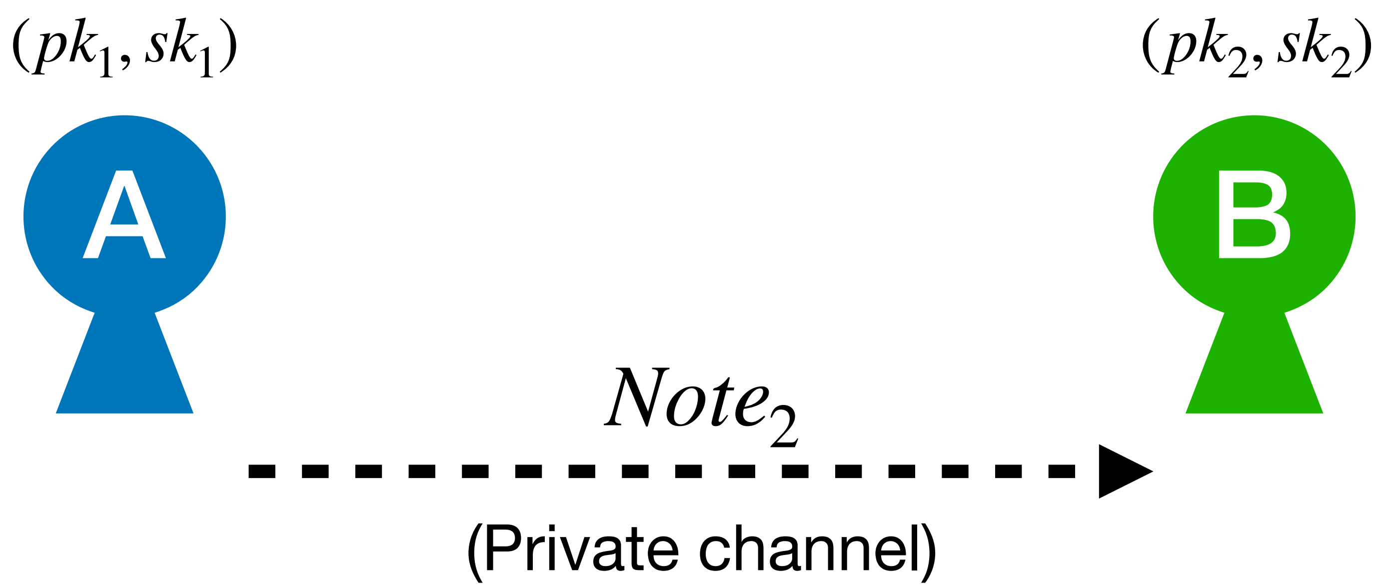
Anonymity pool

- To enhance privacy, Alice can mint many "coins" in a shielding transaction.
- For verifying the transaction, Alice gives a proof for the sum of all the coins is v
- All the miners maintain a table of "commitments" and "nullifiers," which form an anonymity pool.

Commitment	Nullifier
$C_1 = Hash(Note_1)$	
$\vdots$	

Private

What if Alice wants to send her note to Bob?



1. Alice creates a new note $Note_2 = (pk_2, v_2, r_2)$ and sends it to Bob.
2. Alice announces the commitment of the note $C_2 = Hash(Note_2)$
3. In order to make sure $Note_1$ will not be spent again, Alice need to "nullify" $Note_1$. Alice announces

$$N_1 = Hash(r_1)$$

Zcash

- As the blockchain grows, miners and users maintain a table of “commitments” and “nullifiers."

Commitment	Nullifier
$C_1 = Hash(Note_1)$	$N_1 = Hash(r_3)$
$C_2 = Hash(Note_2)$	$N_2 = Hash(r_2)$
$C_3 = Hash(Note_3)$	$\vdots$
$C_4 = Hash(Note_4)$	
$\vdots$	

How do miners verify the transaction?

- How do miners know $Note_1$ is a valid note?
  (miners only see the commitments)
- How do miners know $Note_1$ belongs to Alice?

To make the transaction valid, Alice gives a proof for the following statements:

1. Alice knows pk_1, r_1, v_1 such that Hash(pk_1, r_1, v_1) exists in the table.
2. Alice knows sk_1 corresponding to pk_1.
3. The hash of r_1 is N_1.
4. The input value v_1 equals to output value v_2.

Note that miners cannot know which coin is nullified!!!

Deshielding

Deshielding is very similar

Suppose Alice wants to transfer Note_1 to her transparent address

(pk_0, sk_0)

(pk_3, sk_3)

1. Alice creates a new transparent address (pk_3, sk_3)
2. To “nullify” Note_1, Alice announces

$$N_1 = Hash(r_1)$$

The verification of the deshielding transaction is the same as private transaction.

Construct Zero-Knowledge

- ZK Set
- Bulletproof
- ZK-SNARK

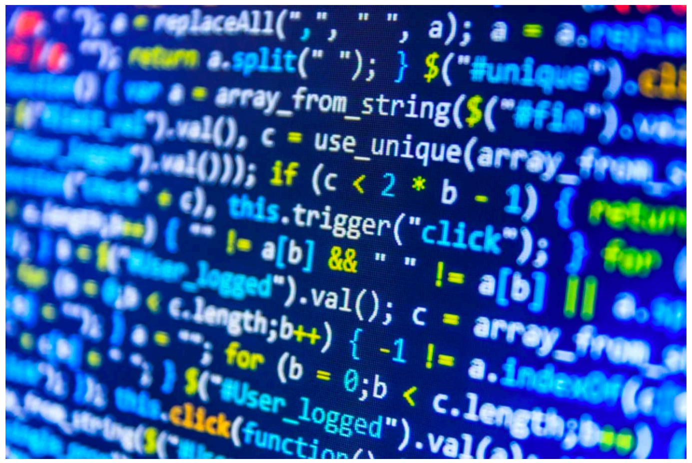
- ZK-START

Zero-Knowledge

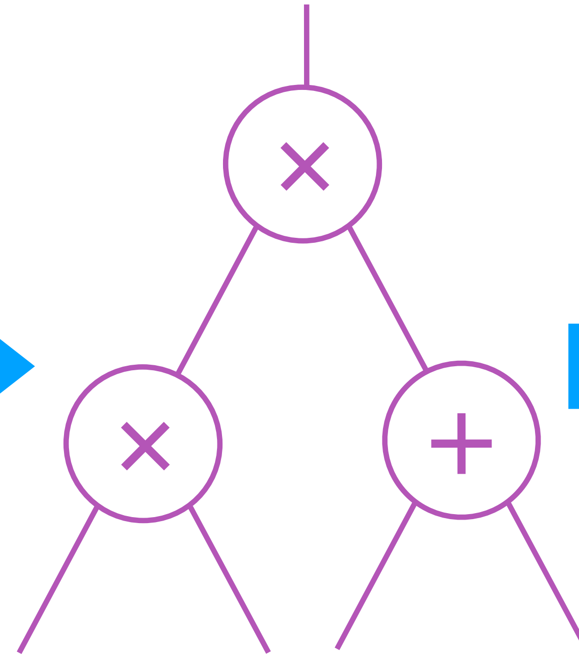
- Completeness
- Soundness
- Zero-knowledge

- Properties to be practical
  - Succinct: it is efficient to verify a proof
  - Public verifiable: anyone can verify the proof
  - non-interactive: no interaction when reading the proof

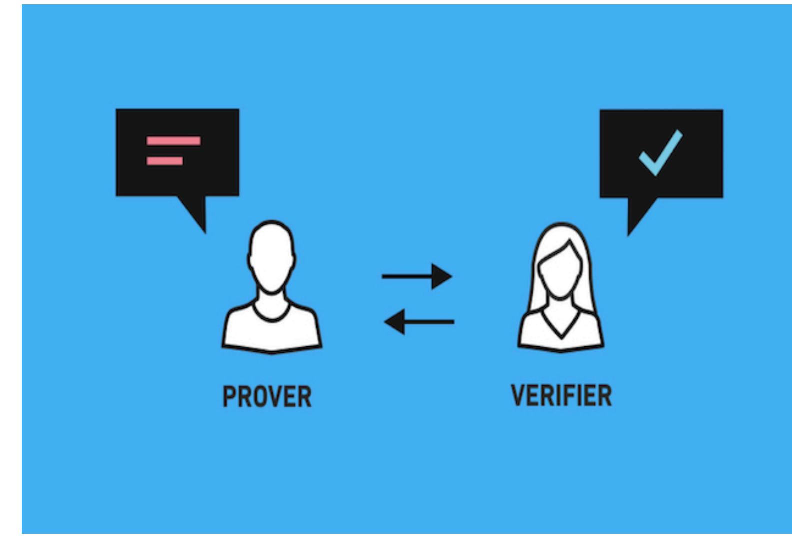
DEXON



Code



Arithmetic Circuit



Zero Knowledge Proof

Proof that you execute the program without leaking any information about input

instruction mnemonic	operands	effects	flag	notes
`and`	ri rj A	compute bitwise AND of $[rj]$ and $[A]$ and store result in ri	result is 0^W	
`or`	ri rj A	compute bitwise OR of $[rj]$ and $[A]$ and store result in ri	result is 0^W	
`xor`	ri rj A	compute bitwise XOR of $[rj]$ and $[A]$ and store result in ri	result is 0^W	
`not`	ri A	compute bitwise NOT of $[A]$ and store result in ri	result is 0^W	
`add`	ri rj A	compute $[rj]_u + [A]_u$ and store result in ri	overflow	
`sub`	ri rj A	compute $[rj]_u - [A]_u$ and store result in ri	borrow	
`mull`	ri rj A	compute $[rj]_u \times [A]_u$ and store least significant bits of result in ri	overflow	
`umulh`	ri rj A	compute $[rj]_u \times [A]_u$ and store most significant bits of result in ri	overflow	
`smulh`	ri rj A	compute $[rj]_s \times [A]_s$ and store most significant bits of result in ri	over/underflow	
`udiv`	ri rj A	compute quotient of $[rj]_u/[A]_u$ and store result in ri	$[A]_u = 0$	
`umod`	ri rj A	compute remainder of $[rj]_u/[A]_u$ and store result in ri	$[A]_u = 0$	
`shl`	ri rj A	shift $[rj]$ by $[A]_u$ bits to the left and store result in ri	MSB of $[rj]$	
`shr`	ri rj A	shift $[rj]$ by $[A]_u$ bits to the right and store result in ri	LSB of $[rj]$	
`cmpe`	ri A	none ("compare equal")	$[ri] = [A]$	
`cmpa`	ri A	none ("compare above", unsigned)	$[ri]_u > [A]_u$	
`cmpae`	ri A	none ("compare above or equal", unsigned)	$[ri]_u \geq [A]_u$	
`cmpg`	ri A	none ("compare greater", signed)	$[ri]_s > [A]_s$	
`cmpge`	ri A	none ("compare greater or equal", signed)	$[ri]_s \geq [A]_s$	
`mov`	ri A	store $[A]$ in ri		
`cmov`	ri A	if flag $= 1$, store $[A]$ in ri		
`jmp`	A	set pc to $[A]$		
`cjmp`	A	if flag $= 1$, set pc to $[A]$ (else increment pc as usual)		
`cnjmp`	A	if flag $= 0$, set pc to $[A]$ (else increment pc as usual)		
`store`	A ri	store $[ri]$ at memory address $[A]_u$		
`load`	ri A	store the content of memory address $[A]_u$ into ri		
`read`	ri A	if the $[A]_u$-th tape has remaining words then consume the next word, store it in ri, and set flag $= 0$; otherwise store 0^W in ri and set flag $= 1$	$\longleftarrow$	(1)
`answer`	A	stall or halt (and the return value is $[A]_u$)		(2)

(1) All but the first two tapes are empty: if $[A]_u \notin \{0, 1\}$ then store 0^W in ri and set flag $= 1$.
(2) `answer` causes a stall (i.e., not increment pc) or a halt (i.e., the computation stops); the choice between the two is undefined.

e.g. compile C code into TinyRAM code by GCC complier
and generate the circuit by the reduction in zk-SNARK [BCGTV'13]
e.g. https://github.com/akosba/jsnark

Zero-knowledge Set

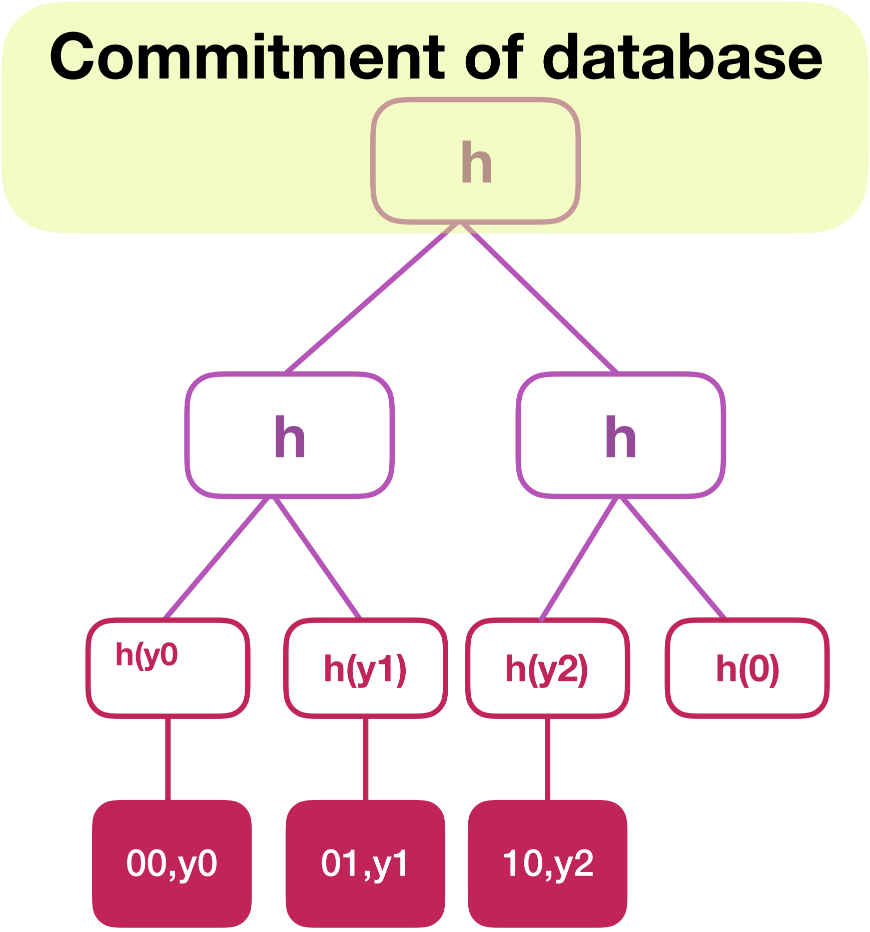
- Problem
  - Give a database $S = \{(x_i,D(x_i))\}_i$, when querying x
    - If x in S, return (D(x), proof)
    - If x not in S, return (no, proof)
  - Each proof leaks no information about S except x in S or not
  - Prover cannot lie (both false positive/negative)
  - Prover convinces the Verifier by the proof

  - Why is this problem hard / non-trivial ?

Zero-knowledge Set

- Can Merkle tree directly adapted on?

- We introduce the method from
  - S. Micali, M. Rabin, J. Kilian
  - “Zero-knowledge sets.”
  - IEEE FOCS 2003



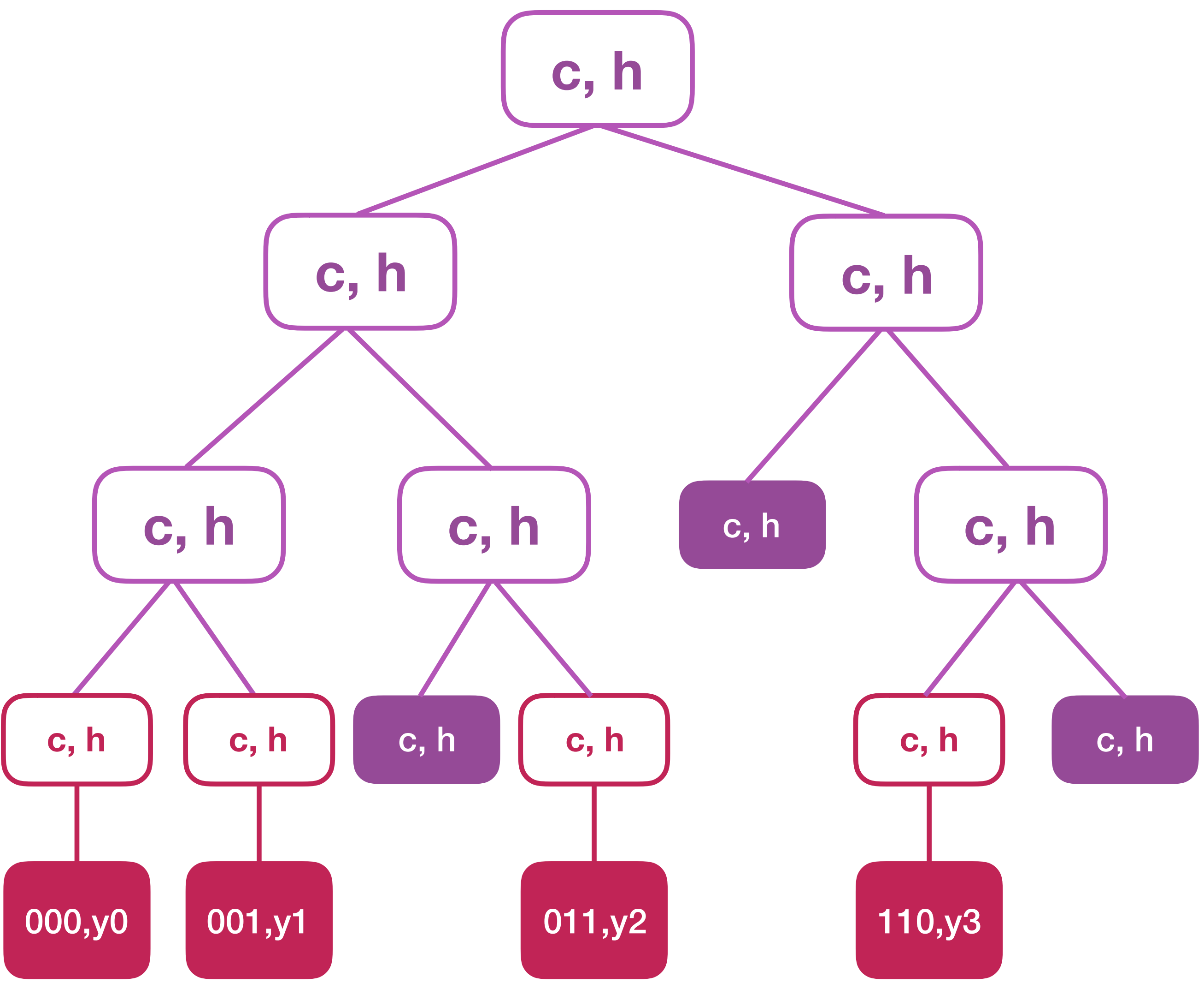
Discrete logarithm Problem

- Given a group $G = <g>$ and $y = g^x$
  - Hard to compute x
  - $\log_g y$ is hard to compute in discrete space

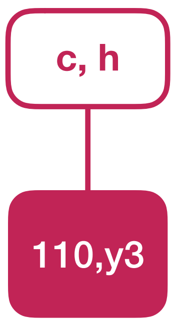
- e.g. $G=(mul, Z_{101})$, $g = 2$, $y=36$, $x=?$

Pedersen Commitment

- Given two independent generators g, h of a group G
  - Pedersen Commitment $H(x, r) = g^x h^r$
  - e.g. $G=(\text{mul}, Z_{11})$, $g = 2$, $h = 6$, $H(3,4) = 6$
- Perfect hiding / computational blinding



Data



$x = 110$

$D(x) = y$

$c = g^{y}(h_{110})^{r_{110}}$

$h_{110} = h^{e_{110}}$

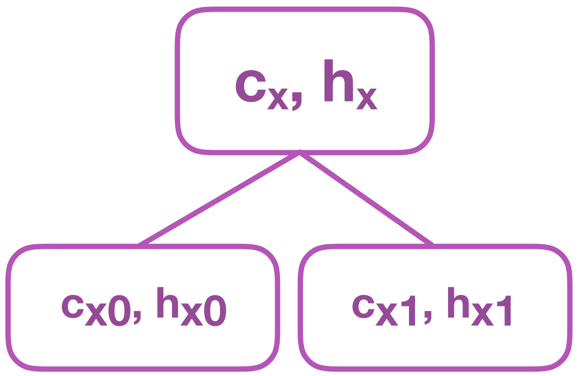
Frontier

$x = 10$

$c = g^{0}(h_{10})^{r_{10}}$

$h_{10} = g^{e_{10}}$

Node

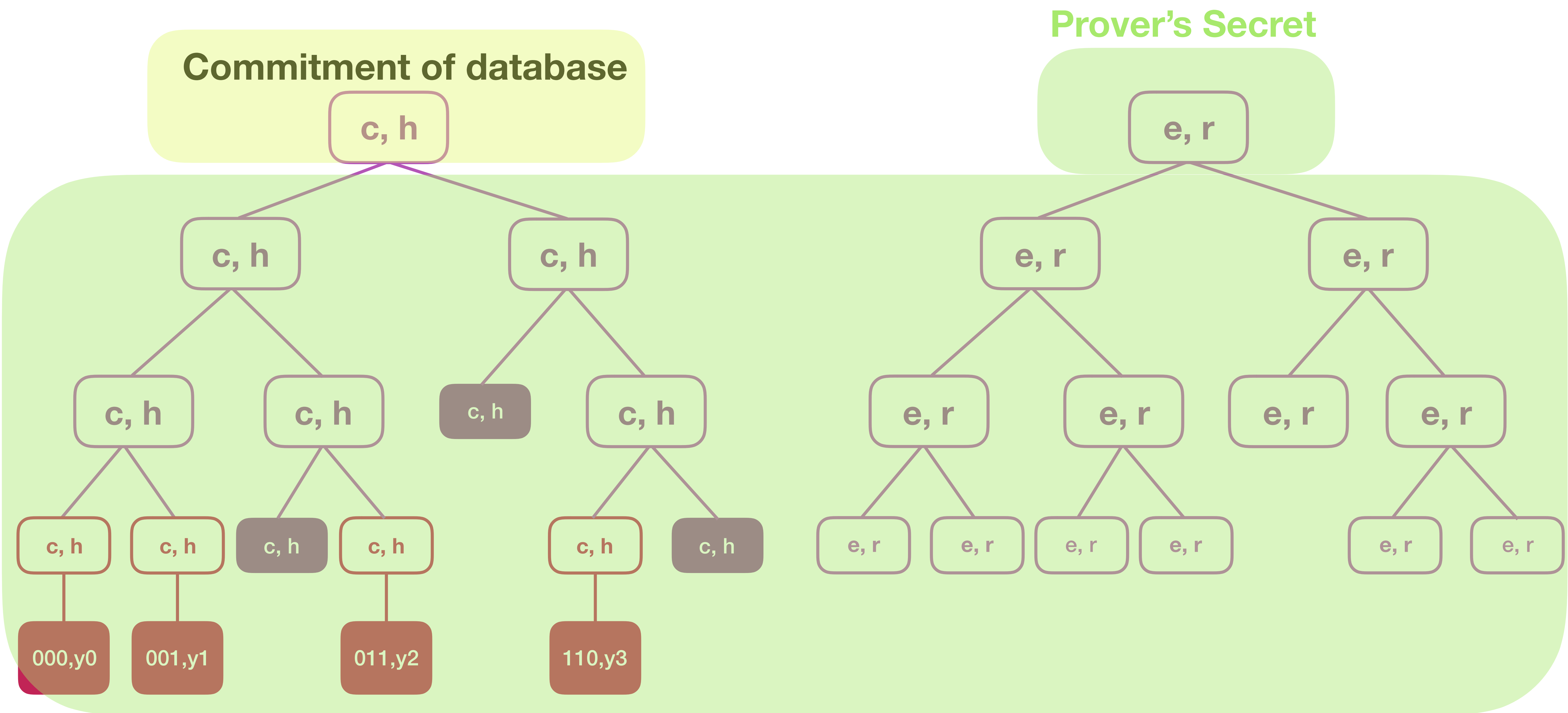


x

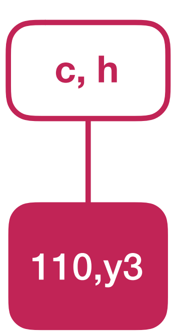
$m = H(c_{x0},h_{x0},c_{x1},h_{x1})$

$c = g^{m}(h_{x})^{r_{x}}$

$h_{x} = h^{e_{x}}$



Data



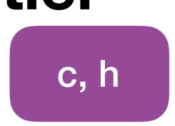
$x = 110$

$D(x) = y$

$c = g^{y}(h_{110})^{r_{110}}$

$h_{110} = h^{e_{110}}$

Frontier

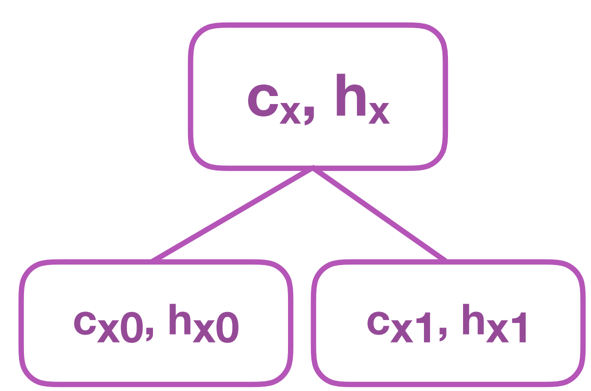


$x = 10$

$c = g^{0}(h_{10})^{r_{10}}$

$h_{10} = g^{e_{10}}$

Node

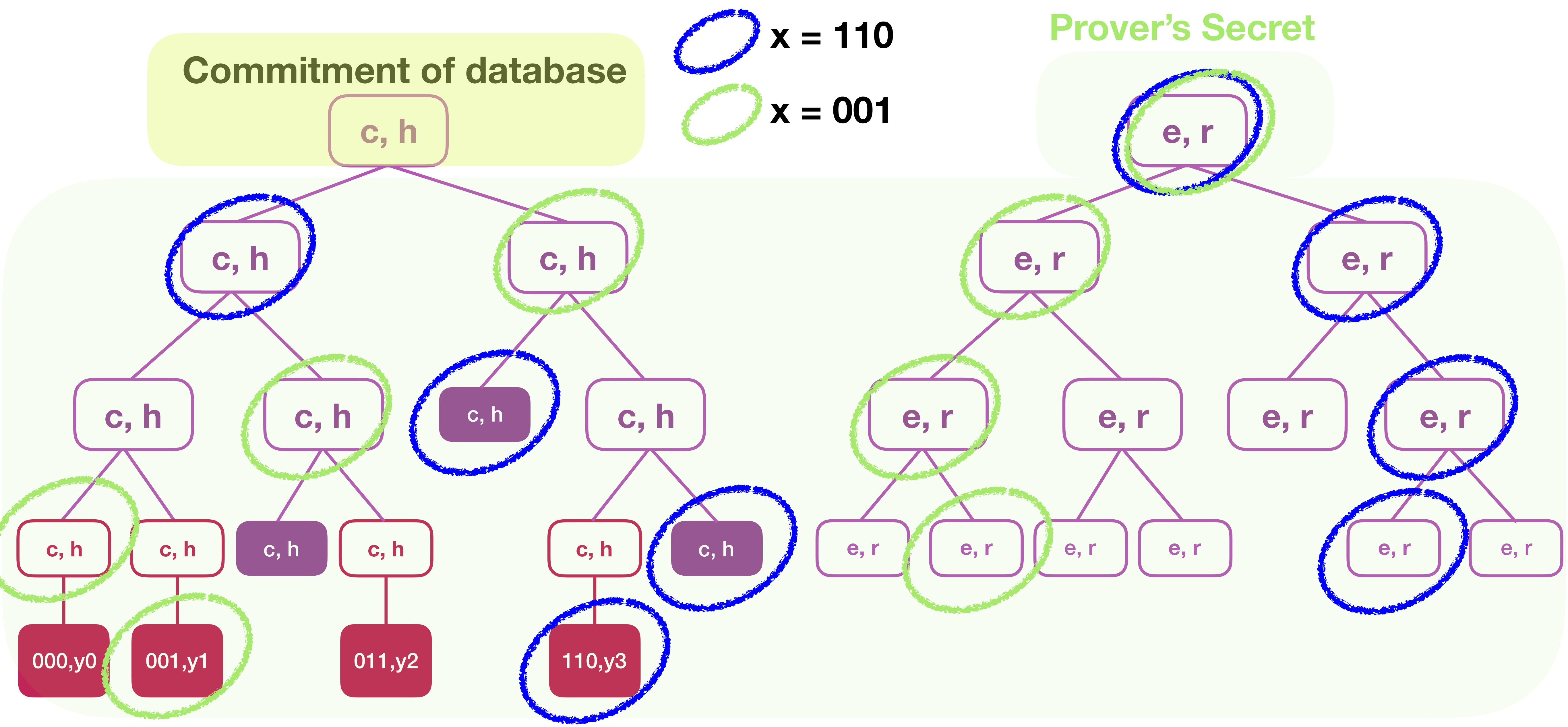


x

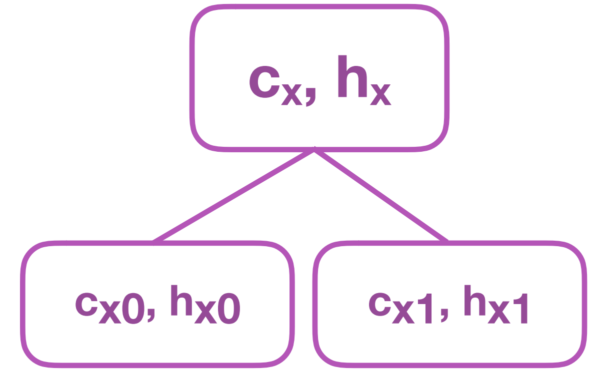
$m = H(c_{x0},h_{x0},c_{x1},h_{x1})$

$c = g^{m}(h_x)^{r_x}$

$h_x = h^{e_x}$



Node



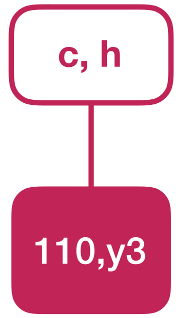
x

$m = H(c_{x0}, h_{x0}, c_{x1}, h_{x1})$

$c = g^m (h_x)^{r_x}$

$h_x = h^{e_x}$

Data



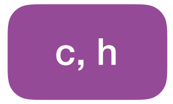
x = 110

D(x) = y

$c = g^y (h_{110})^{r_{110}}$

$h_{110} = h^{e_{110}}$

Frontier

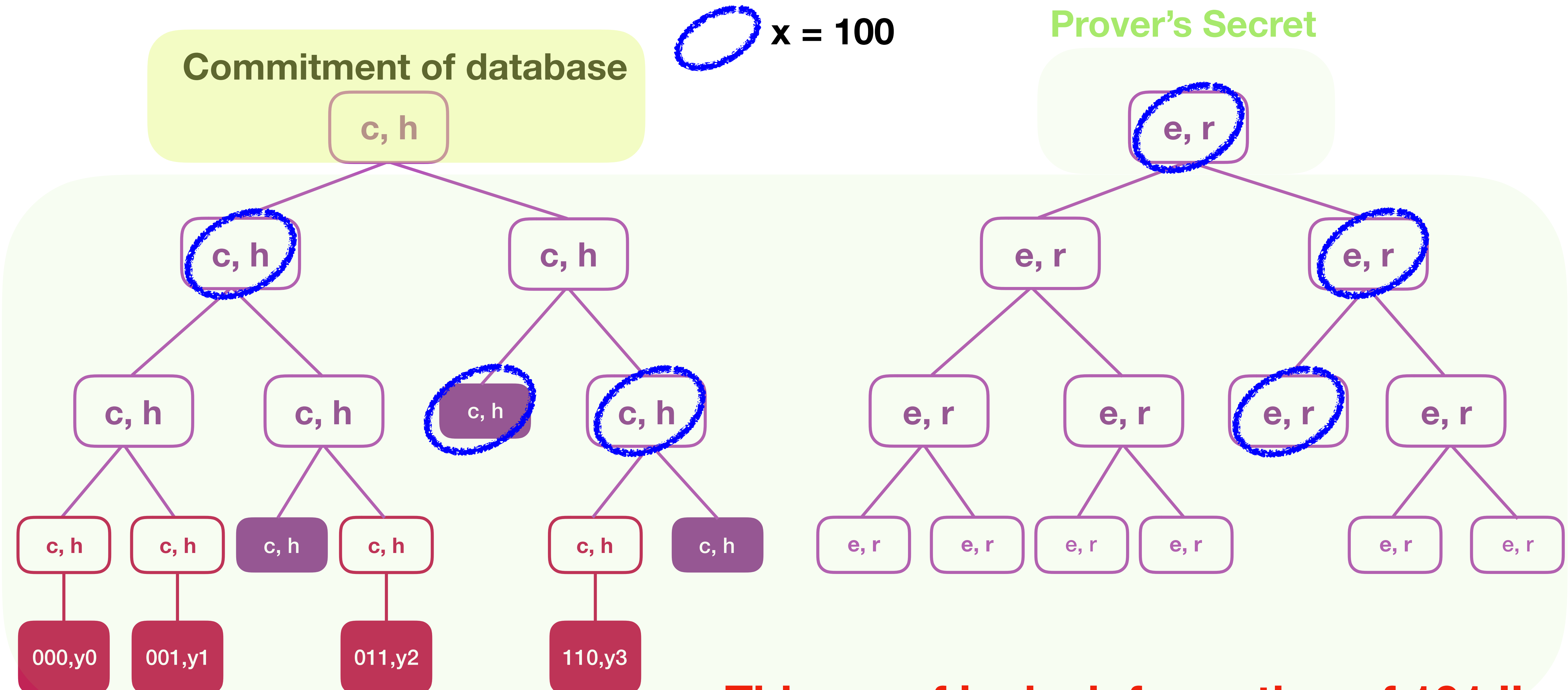


x = 10

$c = g^0 (h_{10})^{r_{10}}$

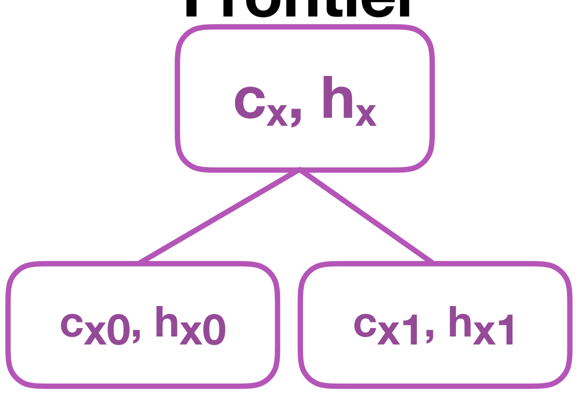
$h_{10} = g^{e_{10}}$

DEXON



This proof leaks information of 101 !!

Frontier



x

$m = H(c_{x0}, h_{x0}, c_{x1}, h_{x1})$

$c_x = g^m (h_x)^{r_x}$

$h_x = g^{e_x}$

$x = 100$

$c = g^0 (h_{100})^{r_{100}}$

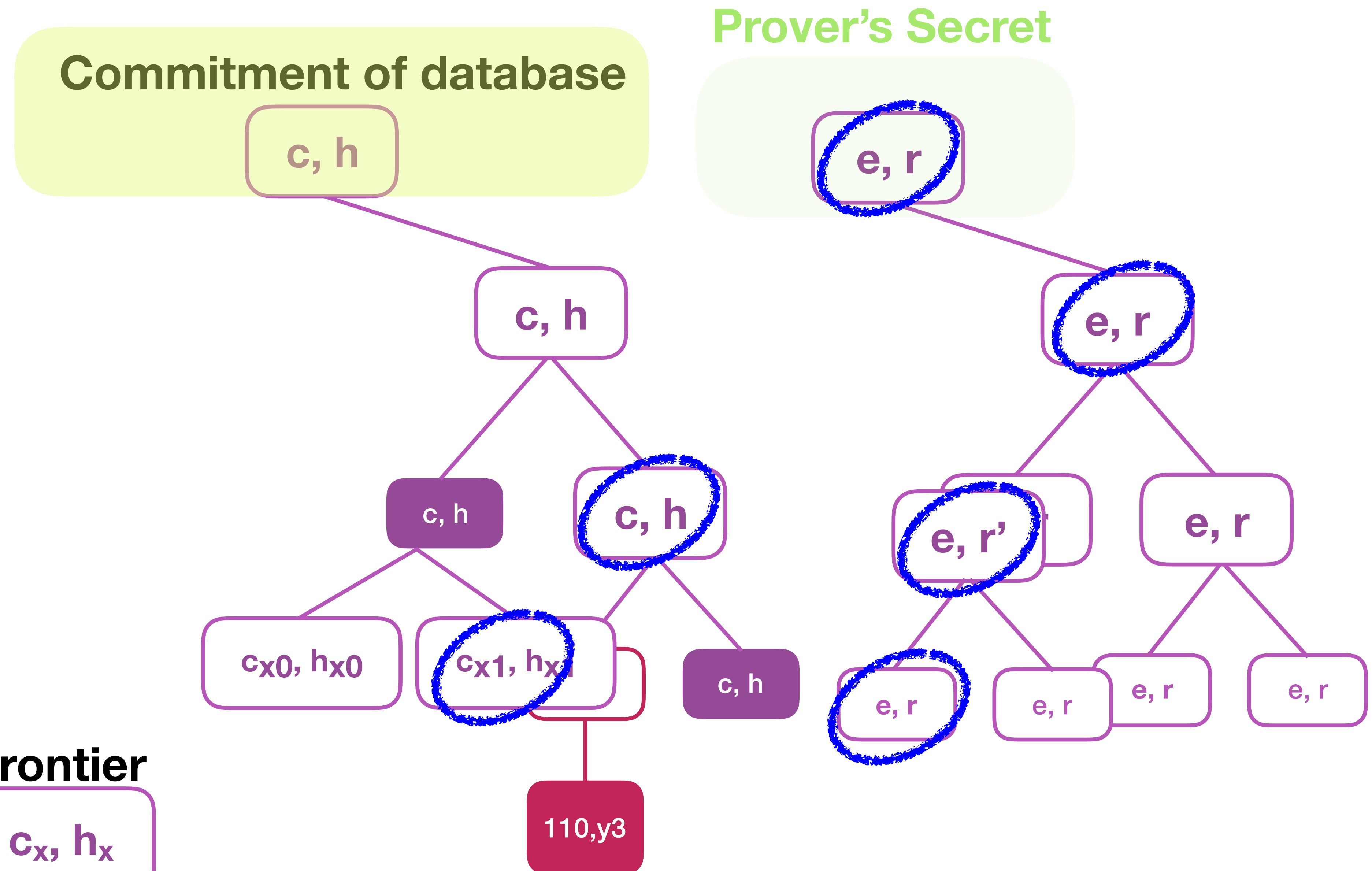
$h_{100} = g^{e_{100}}$

$x = 101$

$c = g^0 (h_{10})^{r_{10}}$

$h_{10} = g^{e_{10}}$

$x = 100$

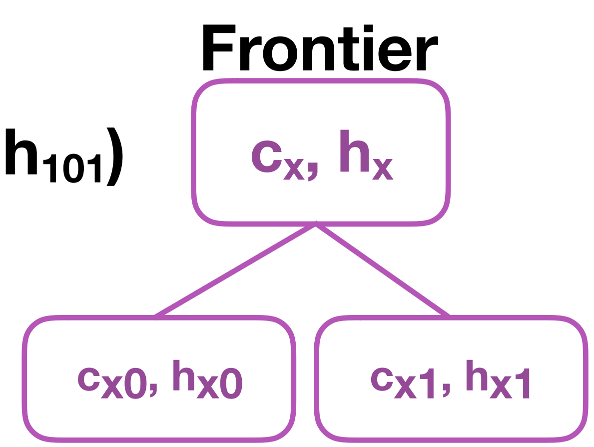


$x = 10$

$m = H(c_{100}, h_{100}, c_{101}, h_{101})$

$c_{10} = g^0 (h_{10})^{r_{10}}$

$h_{10} = g^{e_{10}}$



$x = 100$

$c_{100} = g^0 (h_{100})^{r_{100}}$

$h_{100} = g^{e_{100}}$

$x = 101$

$c_{101} = g^0 (h_{101})^{r_{101}}$

$h_{101} = g^{e_{101}}$

$c_x = g^0 (h_x)^{r_x} = g^m (h_x)^{r'_x}$

$r'_x = (e_x r_x - m)/e_x$

Bulletproof

- Core technique - improving range proof
  - Zero knowledge proof for v in $[0,2^k-1]$
- Improve range proof

- Bulletproofs: Short Proofs for Confidential Transactions and More
- Benedikt B¨unz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell
- IEEE S&P 2018

Notation

- Let $\mathbf{g} = (g_1,g_2,\ldots,g_n)$ be the n-dimensional vector of generators of group G
- Given $\mathbf{x} = (x_1,x_2,\ldots,x_n)$
- Define $\mathbf{g}^{\mathbf{x}} = g_1^{x_1} g_2^{x_2} \ldots g_n^{x_n}$

Inner-Product Argument

- Goal: given two independent generator **g**, **h** in G^n and $\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$, $c=\langle \mathbf{a},\mathbf{b} \rangle$ in Z_p, how could the prover convince a verifier that prover knows **a**, **b** in Z_p^n
- Not zero-knowledge
- Trivial way: send a, b to verifier
- How to reduce the size of proof?

Verifiable Function

- Alice knows $F(x) = a_0 + a_1 x$
- Bob wants to know $F(b)$
- How could Bob be convinced that Alice sent him $F(b)$ without revealing F?

$$
\begin{tikzcd}
F(x) = a_0 + a_1 x \arrow[r, "\text{Challenge } b"] \arrow[d] & a_0 + a_1 b \arrow[d, "g^{a_0 + a_1 b}"'] \\
\text{Public: } g^{a_0}, g^{a_1} \arrow[r, "(g^{a_0})(g^{a_1})^b"'] & =?
\end{tikzcd}
$$

- We prove that the commitment $P=\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{<\mathbf{a},\mathbf{b}>}$ that the prover can convince the verifier in lower size of proof.
- $\mathbf{g} = \mathbf{g_1 g_2}$, $\mathbf{g_1}$ is former n/2 dimension of g, $\mathbf{g_2}$ is later n/2 dimension of g

Challenge x

$\mathbf{a}, \mathbf{b}$ → $\mathbf{a}' = x\mathbf{a_1} + x^{-1}\mathbf{a_2}$, $\mathbf{b}' = x^{-1}\mathbf{b_1} + x\mathbf{b_2}$

$\mathbf{g_1}^{x^{-1}\mathbf{a}'}\mathbf{g_2}^{x\mathbf{a}'}\mathbf{h_1}^{x\mathbf{b}'}\mathbf{h_2}^{x^{-1}\mathbf{b}'}u^{<\mathbf{a}',\mathbf{b}'>}$

$L = \mathbf{g_2}^{\mathbf{a_1}}\mathbf{h_1}^{\mathbf{b_2}}\,u^{<\mathbf{a_1},\mathbf{b_2}>}$

$R = \mathbf{g_1}^{\mathbf{a_2}}\mathbf{h_2}^{\mathbf{b_1}}\,u^{<\mathbf{a_2},\mathbf{b_1}>}$

$L^{x^2}PR^{x^{-2}}$

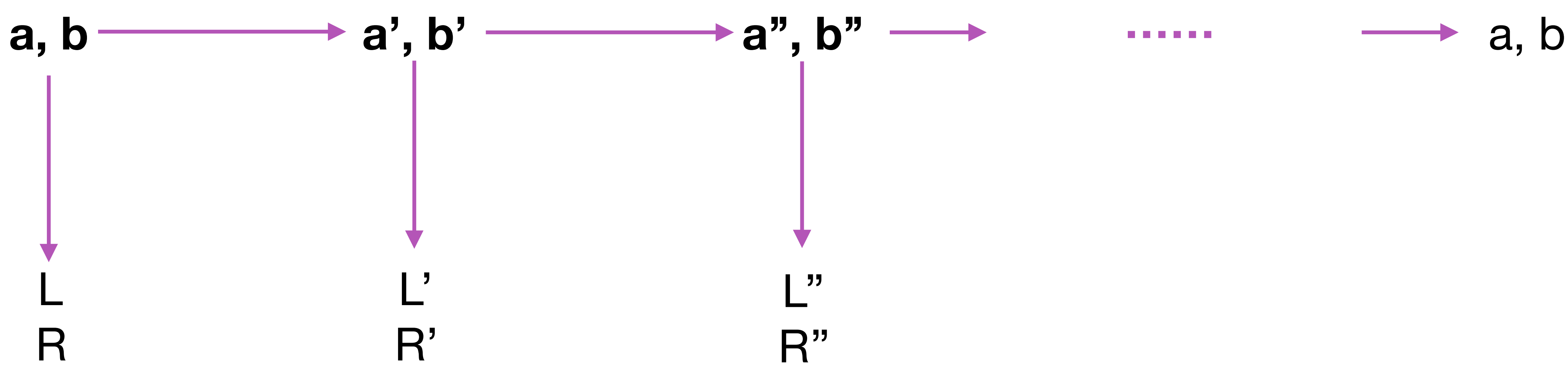
$L^{x^2} =$		$\mathbf{g_2}^{x^2\mathbf{a_1}}$	$\mathbf{h_1}^{x^2\mathbf{b_2}}$		$u^{x^2<\mathbf{a_1},\mathbf{b_2}>}$
$P =$	$\mathbf{g_1}^{\mathbf{a_1}}$	$\mathbf{g_2}^{\mathbf{a_2}}$	$\mathbf{h_1}^{\mathbf{b_1}}$	$\mathbf{h_2}^{\mathbf{b_2}}$	$u^{<\mathbf{a_1},\mathbf{b_1}>+<\mathbf{a_2},\mathbf{b_2}>}$
$R^{x^{-2}} =$	$\mathbf{g_1}^{x^{-2}\mathbf{a_2}}$			$\mathbf{h_2}^{x^{-2}\mathbf{b_1}}$	$u^{x^{-2}<\mathbf{a_2},\mathbf{b_1}>}$
	$\mathbf{g_1}^{x^{-1}\mathbf{a}'}$	$\mathbf{g_2}^{x\mathbf{a}'}$	$\mathbf{h_1}^{x\mathbf{b}'}$	$\mathbf{h_2}^{x^{-1}\mathbf{b}'}$	$u^{<\mathbf{a}',\mathbf{b}'>}$

- Is this completeness?
- Is this soundness?
- Is this zero-knowledge?

- We prove that the commitment $P=\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{\langle \mathbf{a},\mathbf{b}\rangle}$ that the prover can convince the verifier in lower size of proof.
- $\mathbf{g} = \mathbf{g_1}\mathbf{g_2}$, $\mathbf{g_1}$ is former n/2 dimension of g, $\mathbf{g_2}$ is later n/2 dimension of g

$$P' = (g_1^{x^{-1}} g_2^{x})^{a'} (h_1^{x^{-1}} h_2^{x})^{b'} u^{\langle a', b' \rangle}$$

$P=\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}u^{\langle \mathbf{a},\mathbf{b}\rangle}$



Total 2log(n) Group elements and 2 Zp elements
instead of 2n Zp elements

Inner-Product Range Proof

- We want to prove v in $[0,2^n-1]$ without leaking any information about v except the range size.
- Given a commitment $V=h^r g^v$, we want to prove v in $[0,2^n-1]$
- We first write down the mathematical description on the range condition

$$\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v \quad \text{and} \quad \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^n \quad \text{and} \quad \mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n$$

$$\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v \quad \text{and} \quad \langle \mathbf{a}_L \ , \ \mathbf{a}_R \circ \mathbf{y}^n \rangle = 0 \quad \text{and} \quad \langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R \ , \ \mathbf{y}^n \rangle = 0.$$

$$z^2 \cdot \langle \mathbf{a}_L, \mathbf{2}^n \rangle + z \cdot \langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R \ , \ \mathbf{y}^n \rangle + \langle \mathbf{a}_L \ , \ \mathbf{a}_R \circ \mathbf{y}^n \rangle = z^2 \cdot v.$$

$$z^2 \cdot \langle \mathbf{a}_L, \mathbf{2}^n \rangle + z \cdot \langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R \ , \ \mathbf{y}^n \rangle + \langle \mathbf{a}_L \ , \ \mathbf{a}_R \circ \mathbf{y}^n \rangle = z^2 \cdot v.$$

$$\left\langle \mathbf{a}_L - z \cdot \mathbf{1}^n \ , \ \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n) + z^2 \cdot \mathbf{2}^n \right\rangle = z^2 \cdot v + \delta(y, z)$$

$$\delta(y, z) = (z - z^2) \cdot \langle \mathbf{1}^n, \mathbf{y}^n \rangle - z^3 \langle \mathbf{1}^n, \mathbf{2}^n \rangle \in \mathbb{Z}_p$$

$$\begin{aligned} l(X) &= (\mathbf{a}_L - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X \\ r(X) &= \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{2}^n \\ t(X) &= \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2 \end{aligned}$$

$$\mathbf{a}_L \in \{0,1\}^n \text{ s.t.} \langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$$
$$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n \in \mathbb{Z}_p^n$$
$$\alpha \xleftarrow{\$} \mathbb{Z}_p$$
$$A = h^{\alpha} \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} \in \mathbb{G}$$
$$\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^n$$
$$\rho \xleftarrow{\$} \mathbb{Z}_p$$
$$S = h^{\rho} \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G}$$

A, S

$$y, z \xleftarrow{\$} \mathbb{Z}_p^{\star}$$

y, z

$$l(X) = (\mathbf{a}_L - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X \qquad \in \mathbb{Z}_p^n[X]$$
$$r(X) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{2}^n \qquad \in \mathbb{Z}_p^n[X]$$
$$t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2 \qquad \in \mathbb{Z}_p[X]$$
$$t_0 = v \cdot z^2 + \delta(y, z).$$
$$\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_p$$
$$T_i = g^{t_i} h^{\tau_i} \in \mathbb{G}, \quad i = \{1, 2\}$$

T_1, T_2

$$x \xleftarrow{\$} \mathbb{Z}_p^{\star}$$

x

$$\mathbf{l} = l(x) = \mathbf{a}_L - z \cdot \mathbf{1}^n + \mathbf{s}_L \cdot x \in \mathbb{Z}_p^n$$
$$\mathbf{r} = r(x) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot x) + z^2 \cdot \mathbf{2}^n \in \mathbb{Z}_p^n$$
$$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p$$
$$\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 \cdot \gamma \in \mathbb{Z}_p$$
$$\mu = \alpha + \rho \cdot x \in \mathbb{Z}_p$$

$\tau_x, \mu, \hat{t}, \mathbf{l}, \mathbf{r}$

$$h'_i = h_i^{(y^{-i+1})} \in \mathbb{G}, \quad \forall i \in [1, n]$$
$$g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} V^{z^2} \cdot g^{\delta(y,z)} \cdot T_1^x \cdot T_2^{x^2}$$
$$P = A \cdot S^x \cdot \mathbf{g}^{-z} \cdot (\mathbf{h}')^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n} \in \mathbb{G}$$
$$P \stackrel{?}{=} h^{\mu} \cdot \mathbf{g}^{\mathbf{l}} \cdot (\mathbf{h}')^{\mathbf{r}}$$
$$\hat{t} \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p$$

$$\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v \quad \text{and} \quad \mathbf{a}_L \circ \mathbf{a}_R = \mathbf{0}^n \quad \text{and} \quad \mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n$$

$$\left\langle \mathbf{a}_L - z \cdot \mathbf{1}^n \;,\; \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n) + z^2 \cdot \mathbf{2}^n \right\rangle = z^2 \cdot v + \delta(y, z)$$

$A = h^{\alpha} \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} \in \mathbb{G}$ *commitment to* $\mathbf{a}_L$ *and* $\mathbf{a}_R$

$S = h^{\rho} \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G}$ *commitment to* $\mathbf{s}_L$ *and* $\mathbf{s}_R$

$T_i = g^{t_i} h^{\tau_i} \in \mathbb{G}, \quad i = \{1, 2\}$ $\quad t_0 = v \cdot z^2 + \delta(y, z).$

blinding vectors $\mathbf{s}_L, \mathbf{s}_R$

$\mathbf{l} = l(x) = \mathbf{a}_L - z \cdot \mathbf{1}^n + \mathbf{s}_L \cdot x \in \mathbb{Z}_p^n$

$\mathbf{r} = r(x) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot x) + z^2 \cdot \mathbf{2}^n \in \mathbb{Z}_p^n$

$t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2$

$\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 \cdot \gamma \in \mathbb{Z}_p$

$$g^{t(x)} = g^{z^2 v} \; g^{\delta} \; g^{x t_1} \; g^{x^2 t_2}$$
$$h^{\tau_x} = h^{z^2 r} \; h^{x \tau_1} \; h^{x^2 \tau_2}$$
$$= V^{z^2} \; g^{\delta} \; T_1{}^{x} \; T_2{}^{x^2}$$

$h'_i = h_i^{(y^{-i+1})} \in \mathbb{G}, \quad \forall i \in [1, n]$

$g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} V^{z^2} \cdot g^{\delta(y,z)} \cdot T_1^x \cdot T_2^{x^2}$

$P = A \cdot S^x \cdot \mathbf{g}^{-z} \cdot (\mathbf{h}')^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n} \in \mathbb{G}$

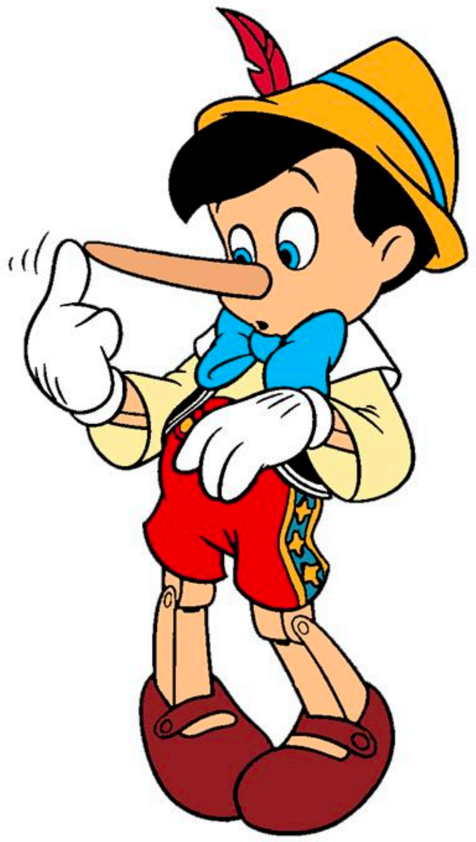
$P \stackrel{?}{=} h^{\mu} \cdot \mathbf{g}^{\mathbf{l}} \cdot (\mathbf{h}')^{\mathbf{r}}$

$\hat{t} \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p$

$$g^{l(x)} = g^{a_L} \; g^{x s_L} \; g^{-z 1^n}$$
$$(h')^{r(x)} = h^{a_R} \; h^{x s_R} \; (h')^{z y^n + z^2 2^n}$$
$$h^{\mu} = h^{\alpha} \; h^{x \rho}$$
$$= A \; S^x \; g^{-z 1^n} \; (h')^{z y^n + z^2 2^n}$$

Pinocchio Protocol

- Bryan Parno, Jon Howell, Craig Gentry, Mariana Raykova
- Pinocchio: Nearly Practical Verifiable Computation
- IEEE S&P 2013
- Describe a correct input/output for an arithmetic circuit by an equation
- Verifier checks the equality of the equation
  = Prover correctly computes the circuit



V1) define target function $T(x)$

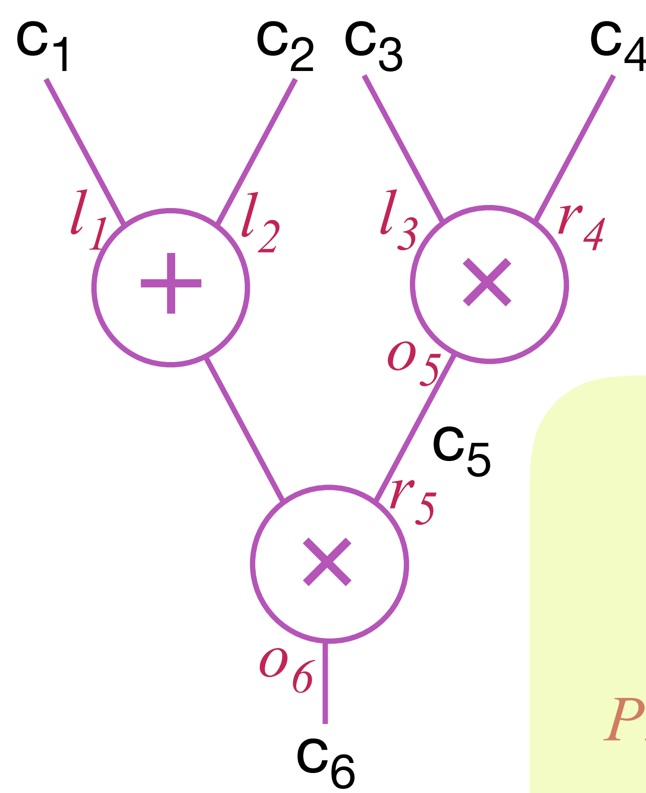
$$T(x)$$

V2) define circuit $\boldsymbol{l}$, $\boldsymbol{r}$, $\boldsymbol{o}$

$$\boldsymbol{l} = \{l_i\}$$

$$\boldsymbol{r} = \{r_i\}$$

$$\boldsymbol{o} = \{o_i\}$$



P1) compute assignment $\{c_i\}$

$$(c_1, c_2, c_3, c_4, c_5, c_6) = (*,*,*,*,*,*)$$

P2) compute circuit formula $L(x)$, $R(x)$, $O(x)$

$$L(x)$$

$$R(x)$$

$$O(x)$$

P3) compute response polynomial $H(x)$

$$P(x) = L(x)R(x)\text{-}O(x)$$

$$H(x) = P(x)/T(x)$$

V3) verify $L(x)$, $R(x)$, $O(x)$, $H(x)$

$$L(x)R(x)\text{-}O(x) =?\ H(x)\ T(x)$$

V1) define target function $T(x)$

$$T(x) = (x-1)(x-2)$$

V2) define circuit ***l***, ***r***, ***o***

$$l_3 = r_4 = o_5 = 2-x$$

$$l_1 = l_2 = r_5 = o_6 = x-1$$

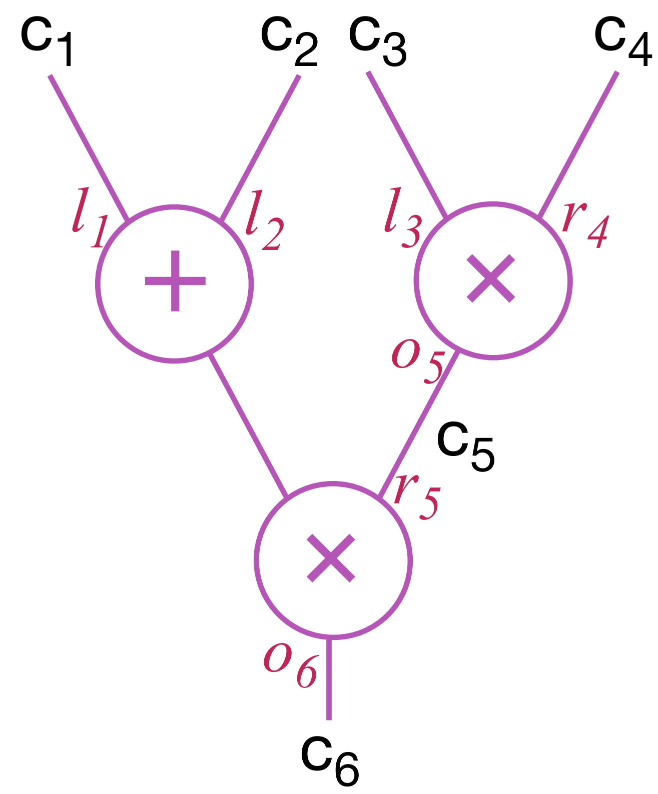
$$\boldsymbol{l} = \{l_1 = x-1,\ l_2 = x-1,\ l_3 = 2-x\}$$

$$\boldsymbol{r} = \{r_4 = 2-x,\ r_5 = x-1\}$$

$$\boldsymbol{o} = \{o_5 = 2-x,\ o_6 = x-1\}$$

V3) verify $L(x)$, $R(x)$, $O(x)$, $H(\mathrm{x})$

$$L(x)R(x)-O(x) =?\ H(x)\ T(x)$$



P1) compute assignment $\{c_i\}$

$$(c_1+c_2)c_3c_4=c_6$$

$$c_3c_4=c_5$$

$$(c_1,\ c_2,\ c_3,\ c_4,\ c_5,\ c_6) = (2,2,3,1,3,12)$$

P2) compute circuit formula $L(x)$, $R(x)$, $O(x)$

$$L(x) = 2(x-1)+2(x-1)+3(2-x) = x+2$$

$$R(x) = (x-1)+3(x-1) = 2x-1$$

$$O(x) = 3(x-1)+12(x-1) = 9x-6$$

P3) compute response polynomial $H(\mathrm{x})$

$$P(x) = L(x)R(x)-O(x)$$

$$= 2x^2-3x+1 = 2(x-1)(x-2)$$

$$H(x) = P(x)/T(x) = 2$$

V1) define target function $T(x)$

$$T(x)$$

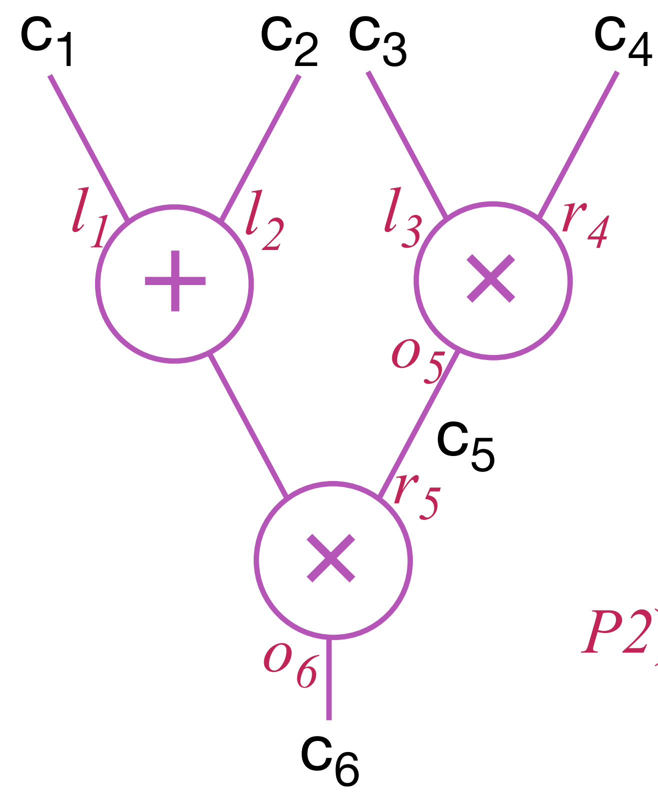
V2) pick a random s (challenge),
V3) define circuit ***l***, ***r***, ***o***

$$\boldsymbol{l} = \{g^{l_i(s)}\}$$

$$\boldsymbol{r} = \{g^{r_i(s)}\}$$

$$\boldsymbol{o} = \{g^{o_i(s)}\}$$

$$\boldsymbol{s} = \{g^{s^i}\}$$



P1) compute assignment $\{c_i\}$

$$(c_1, c_2, c_3, c_4, c_5, c_6) = (*,*,*,*,*,*)$$

P2) compute circuit formula $L(x)$, $R(x)$, $O(x)$

$$g^{L(s)} = \Pi(g^{l_i(s)})^{c_i}$$

$$g^{R(s)} = \Pi(g^{r_i(s)})^{c_i}$$

$$g^{O(s)} = \Pi(g^{o_i(s)})^{c_i}$$

P3) compute response polynomial H(x)

$$H(x) = P(x)/T(x) = \Sigma\, h_i x^i$$

$$g^{H(s)} = \Pi(g^{s^i})^{h_i}$$

P4) send the proof $(g^{L(s)}, g^{R(s)}, g^{O(s)}, g^{H(s)})$ to Verifier

V3) verify $L(x)$, $R(x)$, $O(x)$, H(x)

$$e(g^{L(s)}, g^{R(s)}) \;/\; e(g^{O(s)}, g) =?\; e(g^{H(s)}, g^{T(s)})$$

$$e(g,g)^{L(s)R(s)\text{-}O(s)} =?\; e(g,g)^{H(s)T(s)}$$

- Is this zero-knowledge?
  - No
  - mask the secret!

- How to make sure the Prover use the commitment from verifier?
  - By α-pair commitment
  - Alice send $A_1 = g^x$, $A_2 = g^{x\alpha}$ to Bob
  - Bob compute $B_1 = g^{xb}$, $B_2 = g^{x\alpha b}$ and send to Alice
  - Alice check $B_1^{\alpha} =? B_2$

Fiat-Shamir Transform
From Interactive Proof to Non-Interactive Proof

- convert a protocol into a non-interactive protocol
  - secure
  - full zero-knowledge
  - in the random oracle model
  - Fiat-Shamir heuristic

- E.g.
  - y = H(A,S)
  - z = H(A,S, y)

$$\mathbf{a}_L \in \{0,1\}^n \text{ s.t.} \langle \mathbf{a}_L, \mathbf{2}^n \rangle = v$$
$$\mathbf{a}_R = \mathbf{a}_L - \mathbf{1}^n \in \mathbb{Z}_p^n$$
$$\alpha \xleftarrow{\$} \mathbb{Z}_p$$
$$A = h^{\alpha} \mathbf{g}^{\mathbf{a}_L} \mathbf{h}^{\mathbf{a}_R} \in \mathbb{G}$$
$$\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_p^n$$
$$\rho \xleftarrow{\$} \mathbb{Z}_p$$
$$S = h^{\rho} \mathbf{g}^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R} \in \mathbb{G}$$

$$\xrightarrow{A,\ S}$$

$$y, z \xleftarrow{\$} \mathbb{Z}_p^{\star}$$

$$\xleftarrow{y,\ z}$$

$$l(X) = (\mathbf{a}_L - z \cdot \mathbf{1}^n) + \mathbf{s}_L \cdot X \qquad \in \mathbb{Z}_p^n[X]$$
$$r(X) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot X) + z^2 \cdot \mathbf{2}^n \qquad \in \mathbb{Z}_p^n[X]$$
$$t(X) = \langle l(X), r(X) \rangle = t_0 + t_1 \cdot X + t_2 \cdot X^2 \qquad \in \mathbb{Z}_p[X]$$
$$t_0 = v \cdot z^2 + \delta(y,z).$$
$$\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_p$$
$$T_i = g^{t_i} h^{\tau_i} \in \mathbb{G}, \quad i = \{1,2\}$$

$$\xrightarrow{T_1,\ T_2}$$

$$x \xleftarrow{\$} \mathbb{Z}_p^{\star}$$

$$\xleftarrow{x}$$

$$\mathbf{l} = l(x) = \mathbf{a}_L - z \cdot \mathbf{1}^n + \mathbf{s}_L \cdot x \in \mathbb{Z}_p^n$$
$$\mathbf{r} = r(x) = \mathbf{y}^n \circ (\mathbf{a}_R + z \cdot \mathbf{1}^n + \mathbf{s}_R \cdot x) + z^2 \cdot \mathbf{2}^n \in \mathbb{Z}_p^n$$
$$\hat{t} = \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p$$
$$\tau_x = \tau_2 \cdot x^2 + \tau_1 \cdot x + z^2 \cdot \gamma \in \mathbb{Z}_p$$
$$\mu = \alpha + \rho \cdot x \in \mathbb{Z}_p$$

$$\xrightarrow{\tau_x,\ \mu,\ \hat{t},\ \mathbf{l},\ \mathbf{r}}$$

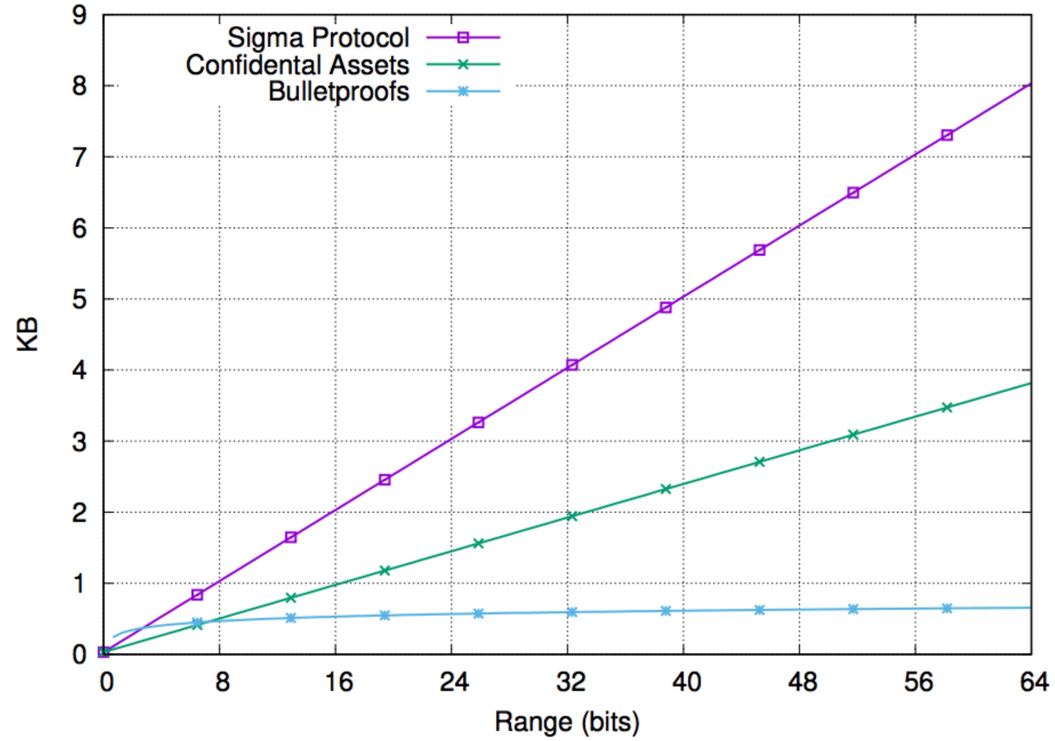
$$h'_i = h_i^{(y^{-i+1})} \in \mathbb{G}, \quad \forall i \in [1,n]$$
$$g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} V^{z^2} \cdot g^{\delta(y,z)} \cdot T_1^x \cdot T_2^{x^2}$$
$$P = A \cdot S^x \cdot \mathbf{g}^{-z} \cdot (\mathbf{h}')^{z \cdot \mathbf{y}^n + z^2 \cdot \mathbf{2}^n} \in \mathbb{G}$$
$$P \stackrel{?}{=} h^{\mu} \cdot \mathbf{g}^{\mathbf{l}} \cdot (\mathbf{h}')^{\mathbf{r}}$$
$$\hat{t} \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r} \rangle \in \mathbb{Z}_p$$

Performance

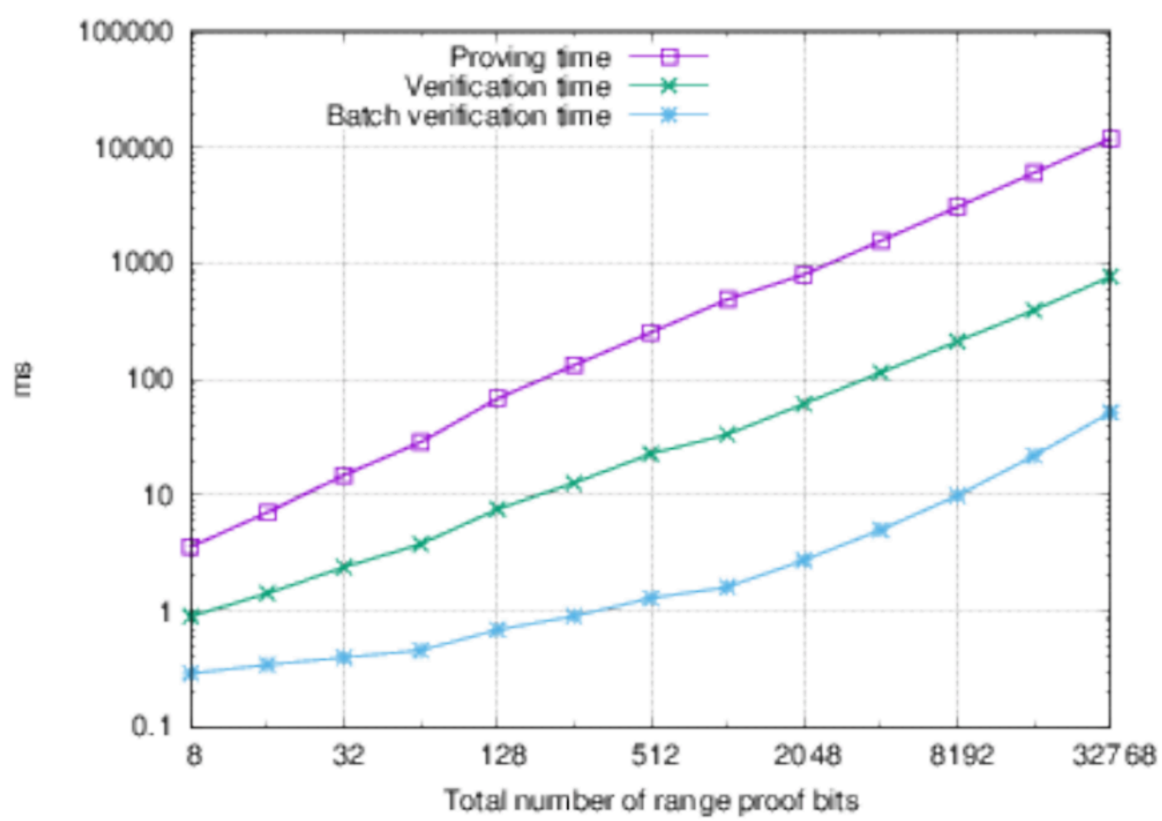
Problem size	Gates	π Size (bytes)	Timing (ms) prove	verify	batch
Range proofs (range $\times$ aggregation size)					
8 bit	8	482	3.7	0.9	0.28
16 bit	16	546	7.2	1.4	0.33
32 bit	32	610	15	2.4	0.38
64 bit	64	675	29	3.9	0.45
64 bit $\times$ 2	128	739	57	6.2	0.55
per range	64	370	29	3.1	0.28
64 bit $\times$ 4	256	803	111	10.4	0.71
per range	64	201	28	2.6	0.18
64 bit $\times$ 8	512	932	213	18.8	1.08
per range	64	117	27	2.4	0.13
64 bit $\times$ 16	1024	932	416	33.2	1.58
per range	64	59	26	2.1	0.10
64 bit $\times$ 32	2048	996	812	61.0	2.67
per range	64	32	25	1.9	0.083
64 bit $\times$ 64	4096	1060	1594	114	4.91
per range	64	17	25	1.8	0.077
64 bit $\times$ 128	8192	1124	3128	210	9.75
per range	64	8.8	25	1.6	0.076
64 bit $\times$ 256	16384	1189	6171	392	21.03
per range	64	4.6	24	1.5	0.082
64 bit $\times$ 512	32768	1253	12205	764	50.7
per range	64	2.5	24	1.5	0.10

Input size	Gates	π Size (bytes)	Timing (ms) prove	verify	batch
Pedersen hash preimage (input size)					
48 bit	128	864	88	6.4	0.72
96 bit	256	928	172	10.6	0.93
192 bit	512	992	335	19.1	1.33
384 bit	1024	1056	659	33.6	2.12
768 bit	2048	1120	1292	61.6	3.66
1536 bit	4096	1184	2551	114.9	6.93
3072 bit	8192	1248	5052	213.4	13.20
Unpadded SHA256 preimage					
512 bit	25400	1376	19478	749.9	41.52

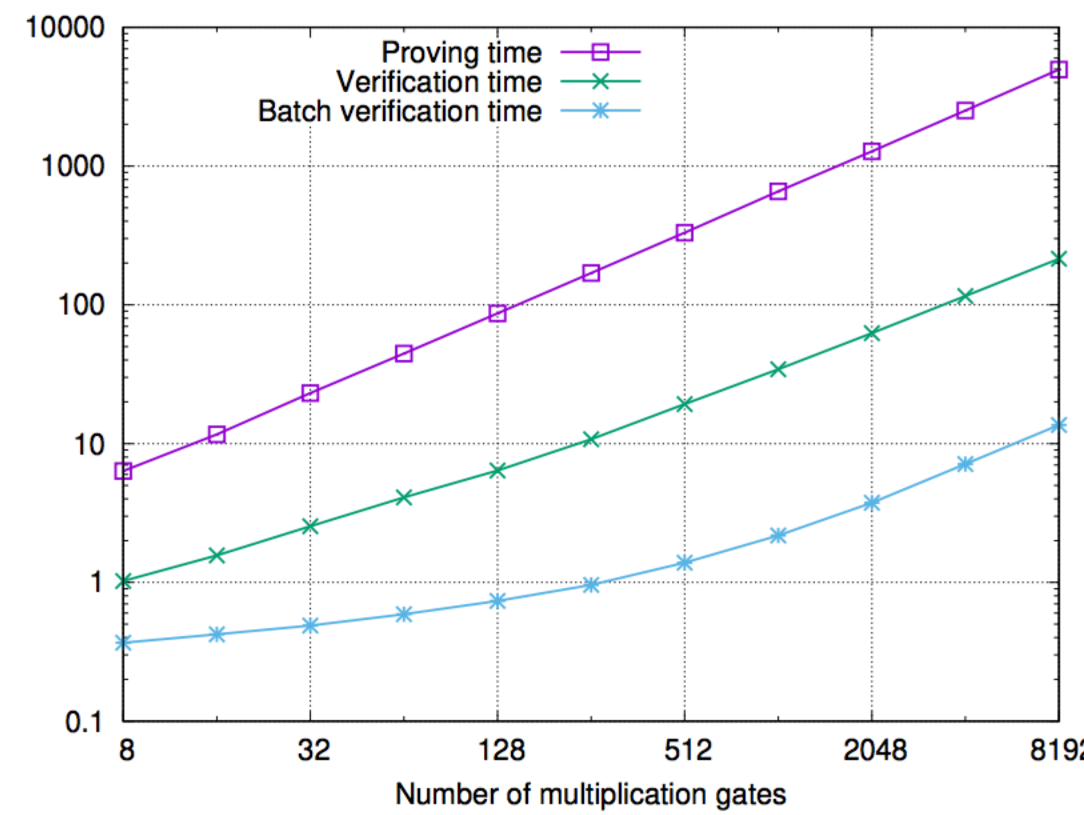
Bulletproof



Proof size

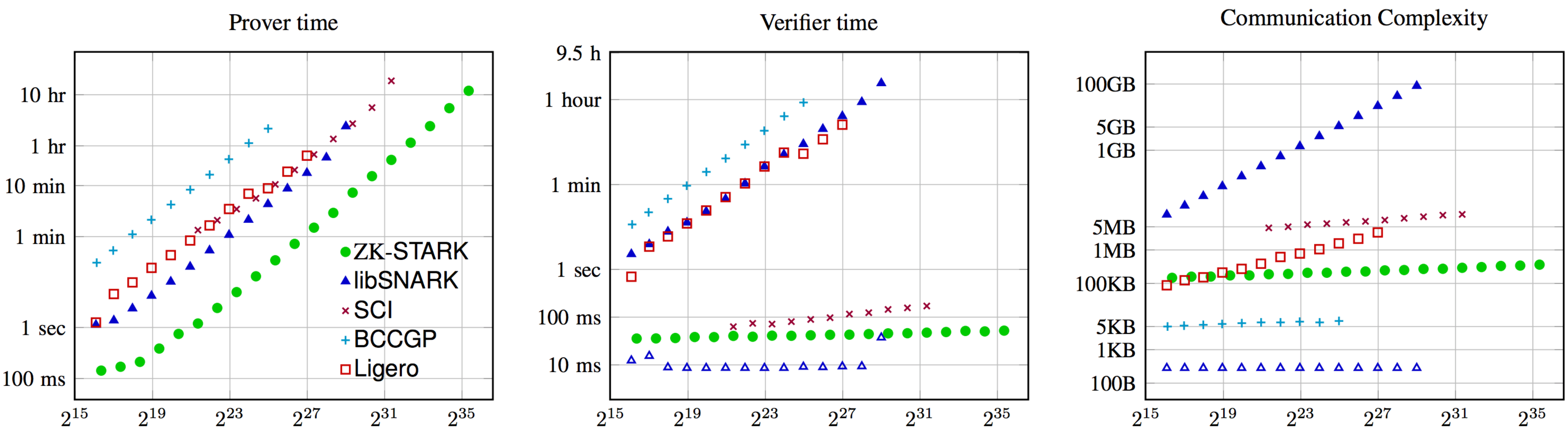


Computation timing



Computation timing for Pedersen Hash

zk-SNARK & zk-START



Take away

To prove a 128bit secure Hash…

	Proving time	Verifying time	Proof Size
zk-SNARK	2 mins	0.005 sec	288 bytes
zk-STARK	40 sec	0.08 sec	120 KB
Bulletproof	0.3 sec	0.02 sec	1 KB