Zero-Knowledge Proofs I
Lelantus

Oct. 16, 2019

Overview

- Zero-Knowledge
  - Proving a property about an element without revealing

- Lelantus
  - ZCoin’s Zero-Knowledge protocol
  - Prove that transactions are valid, without revealing anything

Zero-Knowledge Proofs (ZKP)

- A proof about a property without revealing it
- Zero-Knowledge is not magic
- We have already seen several instances of ZKP
  - Signatures are ZK proofs of knowing the secret key
    - In ECC, the secret key a is the discrete logarithm of the public key $A = aG$
    - Also called proof of knowledge of discrete logarithm

Zero-Knowledge Proofs (ZKP)

- Another example we saw:
  - Pedersen Commitment $X = aG + \lambda H$
    - We can proof that $a = 0$ without revealing λ by using X as public key in a signature (s, R), $sH = R + \mathscr{H}(\dots)X$

- Those techniques are called *Non-Interactive Signature-based Proof-of-Knowledge* (NI SPK)

Zero-Knowledge Proofs (ZKP)

- A more general approach is the so called Σ-protocol
- A three way protocol

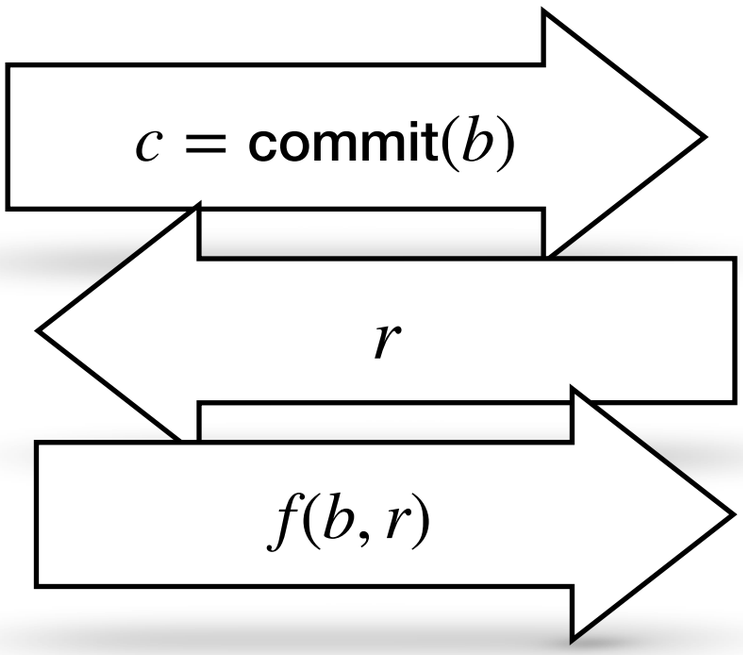
Alice

some value b

compute $f(b, r)$ for some function f

Bob

random r



Accepts if conditions are met

Zero-Knowledge Proofs (ZKP)

- A Zero-Knowledge Σ-protocol to show knowledge of discrete logarithm of $P = pG$

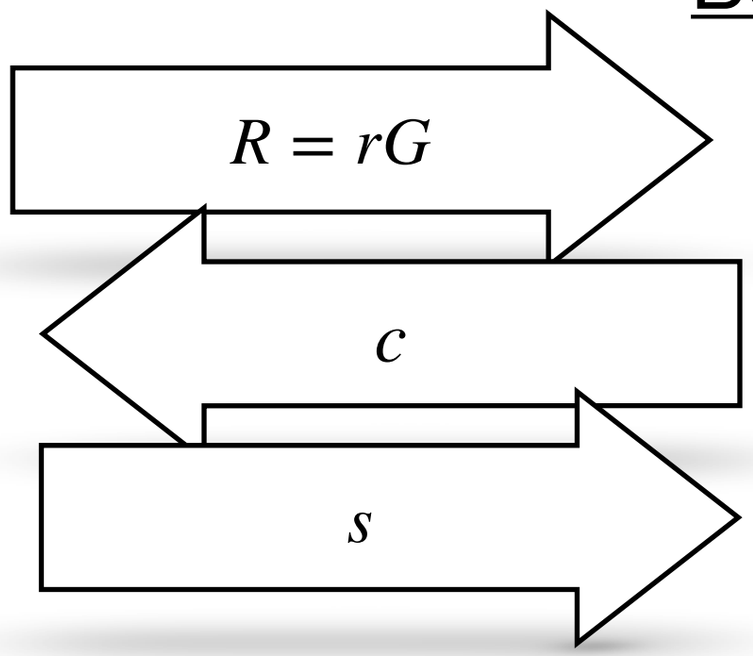
Alice, knows p

random value r

Commit via ECC Point

compute

$f(r, c, p) = s = r + cp$



Bob, knows P

random challenge c

Accepts if

$sG \stackrel{?}{=} R + cP$

Zero-Knowledge Proofs (ZKP)

- A Zero-Knowledge Σ-protocol to show knowledge of discrete logarithm of $P = pG$

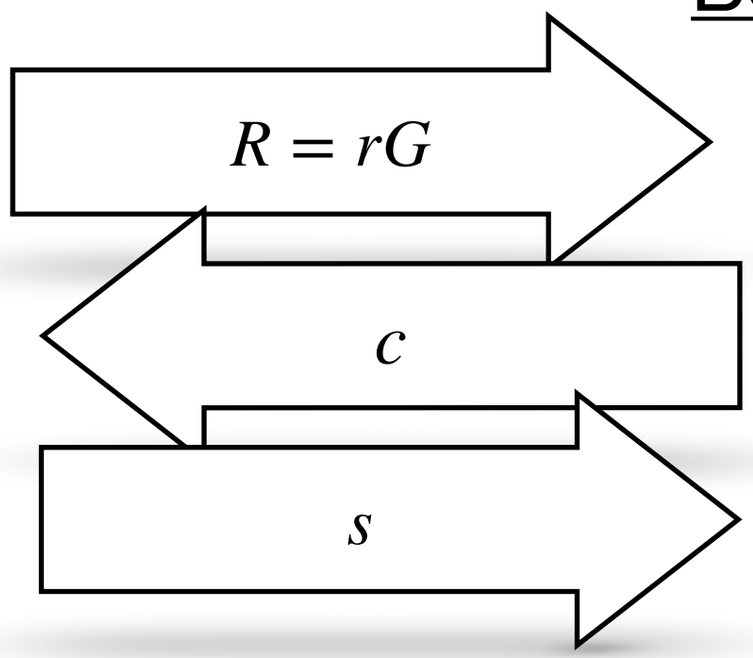
Alice, knows p

random value r

Commit via ECC Point

compute

$f(r, c, p) = s = r + cp$



Bob, knows P

random challenge c

Accepts if

$$sG \stackrel{?}{=} R + cP$$

Same formula as Schnorr Signature

Zero-Knowledge Proofs (ZKP)

- A Zero-Knowledge Σ-protocol to show knowledge of discrete logarithm of $P = pG$

Alice, knows p

random value r, commit via ECC Point $R = rG$

challenge c is a hash using input R, P: $c = \mathcal{H}(R \mid P)$

With $s = r + cp = r + \mathcal{H}(\ldots)p$, the Schnorr Signature is (s, R)

$\Rightarrow$ Hashes can be used to transform an interactive Zero Knowledge proof into a non-interactive proof

Zero-Knowledge Proofs (ZKP)

- Zero-knowledge proofs are often shown as Σ-protocol

    1. Commit some value
    2. accept a challenge
    3. send a function

- With a hash it can be turned into a Non-Interactive proof

Σ-protocol for Pedersen commit as 0 or 1

- Assume we have a Pedersen Commitment $X = aG + \lambda H$

- Before, we have seen a ZKP to show that $a = 0$

- Now, we look at a ZKP to show that $a = 0$ or $a = 1$

Σ-protocol for Pedersen commit as 0 or 1

- A ZKP to show that $a = 0$ or $a = 1$
  - How can that work?

Σ-protocol for Pedersen commit as 0 or 1

- A ZKP to show that $a = 0$ or $a = 1$
  - How can that work?
- The one thing $a = 0$ and $a = 1$ have in common:
  - We proof that $a(1 - a) = 0$

Σ-protocol for Pedersen Commitment as 0 or 1

$$C = mG + rH, \text{ proof } m \in \{0,1\}$$

Step 1

- Alice (knows $C = mG + rH$)
- generates random $a, s, t \in \mathbb{Z}$
- commit and send
  - $c_a = aG + sH$
  - $c_b = (am)G + tH$

Σ-protocol for Pedersen Commitment as 0 or 1

$$C = mG + rH, \text{ proof } m \in \{0,1\}$$

Step 1

- $c_a = aG + sH$
- $c_b = (am)G + tH$

Step 2

$\xleftarrow{x}$ send challenge x

Σ-protocol for Pedersen Commitment as 0 or 1

$$C = mG + rH, \text{ proof } m \in \{0,1\}$$

Step 1

- $c_a = aG + sH$
- $c_b = (am)G + tH$

$\xleftarrow{x}$ Step 2: random x

Step 3

$f = mx + a$

$z_a = rx + s$

$z_b = r(x - f) + t$

$\xrightarrow{f, z_a, z_b}$

Σ-protocol for Pedersen Commitment as 0 or 1

$$C = mG + rH, \text{ proof } m \in \{0,1\}$$

Step 1

- $c_a = aG + sH$
- $c_b = (am)G + tH$

$\xleftarrow{x}$ Step 2: random x

Step 3

$f = mx + a$

$z_a = rx + s$

$z_b = r(x - f) + t$

$\xrightarrow{f,z_a,z_b}$ Accept if and only if:

$$xC + c_a = fG + z_aH$$

$$(x - f)\,C + c_b = 0G + z_bH$$

Σ-protocol for Pedersen Commitment as 0 or 1

$$C = mG + rH, \text{ proof } m \in \{0,1\}$$

Alice sends

$$c_a = aG + sH \qquad c_b = (am)G + tH$$

$$f = mx + a \qquad z_a = rx + s \qquad z_b = r(x - f) + t$$

Bob verifies: $xC + c_a \stackrel{?}{=} fG + z_a H$

$$\begin{aligned} xC + c_a &= x(mG + rH) + (aG + sH) \\ &= xmG + aG + xrH + sH \\ &= (xm + a)G + (xr + s)\,H \\ &= fG + z_a H \end{aligned}$$

Σ-protocol for Pedersen Commitment as 0 or 1

$$C = mG + rH, \text{ proof } m \in \{0,1\}$$

Alice sends

$$c_a = aG + sH \qquad c_b = (am)G + tH$$

$$f = mx + a \qquad z_a = rx + s \qquad z_b = r(x - f) + t$$

Bob verifies: $xC + c_a \stackrel{?}{=} fG + z_a H$

- We do not make any assumption about a, s
- $xC + c_a = (mx + a)G + (\ldots)H$
- If $xC + c_a = fG + (\ldots)H$, we know that $f = mx + a$

Σ-protocol for Pedersen Commitment as 0 or 1

$$C = mG + rH, \text{ proof } m \in \{0,1\}$$

Alice sends

$$c_a = aG + sH \qquad c_b = (am)G + tH$$

$$f = mx + a \qquad z_a = rx + s \qquad z_b = r(x - f) + t$$

Bob verifies: $\left(x - f\right) C + c_b \stackrel{?}{=} 0G + z_b H$

- now we test $m(1 - m) = 0$ property via

$$\left(x - f\right) C + c_b = \left(x - (mx + a)\right) C + c_b$$

$$= \left(x - (mx + a)\right)(mG + rH) + c_b$$

Σ-protocol for Pedersen Commitment as 0 or 1

$$C = mG + rH, \text{ proof } m \in \{0,1\}$$

$$\begin{aligned}
\left(x - f\right) C + c_b &= \left(x - (mx + a)\right) C + c_b \\
&= \left(x - (mx + a)\right)(mG + rH) + c_b \\
&= \left(x - (mx + a)\right) mG + (x - f)rH + c_b \\
&= (xm - m^2x - ma)G + (x - f)rH + (amG + tH) \\
&= (xm - m^2x)G + (x - f)rH + tH \\
&= xm(1 - m)G + \left(r(x - f) + t\right) H \\
&\stackrel{?}{=} 0G + z_bH
\end{aligned}$$

Σ-protocol for Pedersen Commitment as 0 or 1

$$C = mG + rH, \text{ proof } m \in \{0,1\}$$

Alice sends

$$c_a = aG + sH \qquad c_b = (am)G + tH$$

$$f = mx + a \qquad z_a = rx + s \qquad z_b = r(x - f) + t$$

Bob verifies: $\left(x - f\right) C + c_b \stackrel{?}{=} 0G + z_b H$

- now we test $m(1 - m) = 0$ property via

$$\left(x - f\right) C + c_b = 0G + (\ldots)H$$

Σ-protocol for Pedersen Commitment as 0 or 1

$$C = mG + rH, \text{ proof } m \in \{0,1\}$$

Alice sends

$$c_a = aG + sH \qquad c_b = (am)G + tH$$

$$f = mx + a \qquad z_a = rx + s \qquad z_b = r(x - f) + t$$

Bob verifies:

- if $xC + c_a = fG + z_aH$ and $\left(x - f\right) C + c_b = 0G + z_bH$
- then: $f = mx + a$ and $xm(1 - m) = 0$, regardless of x
- Thus we know that $m \in \{0,1\}$

Σ-protocol for Pedersen Commitment as 0 or 1

- Wy do we do this?
- It is very very cool!
- We can use this as building block for more complex proofs
  - 1-in-N Σ-protocols

1-in-N Σ−Protocol

- Assume we have a set of Pedersen Commitments given
- $\{X_1, X_2, \ldots, X_n\}$,
  - each $X_i = m_i G + r_i H$ has
    - amount m_i
    - randomness r_i as blinding value

1-in-N Σ−Protocol

- Assume we have a set of Pedersen Commitments given
- $\{X_1, X_2, \ldots, X_n\}$, each $X_i = m_i G + r_i H$
- Assume we know $X_t = m_t G + r_t H$
- We want to prove that we know one of the X_i

1-in-N Σ-Protocol

- Given:
  - $\{X_1, X_2, \ldots, X_n\}, X_i = m_i G + r_i H, X_t = m_t G + r_t H$
- We want to prove that we know one of the X_i
- Publish related Pedersen Commitment $Y = m_t G + sH$
- Verifier subtracts Y from all Pedersen Commitments
- Proof is now: 1 in $\{X_1 - Y, X_2 - Y, \ldots, X_n - Y\}$ is $0G + (\ldots)H$
  - Technial term: *opens to 0*

1-in-N Σ−Protocol

- New Problem:
  - $\{Y_1, Y_2, \ldots, Y_n\}, Y_i = m_i G + s_i H, Y_t = 0G + s_t H$
- We want to prove that one of the Y_i opens to 0

1-in-N Σ−Protocol

- New Problem:
  - $\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i = m_i G + s_i H$, $Y_t = 0G + s_t H$
- We want to prove that one of the Y_i opens to 0
- Idea:
  - show that $c_1 Y_1 + c_2 Y_2 + \ldots + c_n Y_n$ opens to 0
  - show that each c_i is either 0 or 1
  - show that $\sum c_i$ is 1

1-in-N Σ−Protocol

- New Problem:
  - $\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i = m_i G + s_i H$, $Y_t = 0G + s_t H$
  - given $c_1 Y_1 + c_2 Y_2 + \ldots + c_n Y_n$
  - show that each c_i is either 0 or 1
    - if c is a number, we reveal the secret
    - if c is a group element, we don't know what $c_i Y_i$ means

1-in-N Σ–Protocol

Alice sends

$$c_a = aG + sH \qquad c_b = (am)G + tH$$

$$f = mx + a \qquad z_a = rx + s \qquad z_b = r(x - f) + t$$

Bob verifies:

- if $xC + c_a = fG + z_a H$ and $(x - f)\,C + c_b = 0G + z_b H$
- then: $f = mx + a$ and $xm(1 - m) = 0$, regardless of x
- Thus we know that $m \in \{0,1\}$

- Look at previous proof:
- consider $f = mx + a$
  - Contains the value $m \in \{0,1\}$
  - since a, m is secret, knowing f doesn't reveal m

1-in-N Σ−Protocol

- New Problem:
  - $\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i = m_i G + s_i H$, $Y_t = 0G + s_t H$
  - given $f_1 Y_1 + f_2 Y_2 + \ldots + f_n Y_n$
  - Conduct N parallel Σ protocols for $f_i = m_i x_i + a_i$
    - That gives a proof that $m_i \in \{0,1\}$

1-in-N Σ−Protocol

- New Problem:
  - $\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i = m_i G + s_i H$, $Y_t = 0G + s_t H$
  - now we have

$$f_1 Y_1 + f_2 Y_2 + \ldots + f_n Y_n$$

1-in-N Σ−Protocol

- New Problem:
  - $\{Y_1, Y_2, \ldots, Y_n\}, Y_i = m_i G + s_i H, Y_t = 0G + s_t H$
- now we have

$$f_1 Y_1 + f_2 Y_2 + \ldots + f_n Y_n$$

$$= (m_1 x + a_1) Y_1 + (m_2 x + a_2) Y_2 + \ldots + (m_n x + a_n) Y_n$$

1-in-N Σ−Protocol

- New Problem:
  - $\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i = m_i G + s_i H$, $Y_t = 0G + s_t H$
- now we have

$$f_1 Y_1 + f_2 Y_2 + \ldots + f_n Y_n$$

$$= (m_1 x + a_1) Y_1 + (m_2 x + a_2) Y_2 + \ldots + (m_n x + a_n) Y_n$$

$$= m_k x Y_k + \sum a_k Y_k$$

1-in-N Σ−Protocol

- New Problem:
  - $\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i = m_i G + s_i H$, $Y_t = 0G + s_t H$
- but now we have

$$f_1 Y_1 + f_2 Y_2 + \ldots + f_n Y_n$$

$$= (m_1 x + a_1) Y_1 + (m_2 x + a_2) Y_2 + \ldots + (m_n x + a_n) Y_n$$

$$= m_k x Y_k + \sum a_k Y_k$$

Opens to 0

independent of x,
can be send beforehand
in a Pedersen Commitment

1-in-N Σ−Protocol

New Problem:

- $\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i = m_i G + s_i H$, $Y_t = 0G + s_t H$

Proof:

$f_1 Y_1 + f_2 Y_2 + \ldots + f_n Y_n = m_k x Y_k + \sum a_k Y_k$ opens to 0

- Doable, but not very efficient
  - n is the size of the anonymity set
  - We can do better

1-in-N Σ−Protocol

Summary:

- A efficient, but slightly complicated, protocol
- We can show that we know an index t of an element in the anonymity that opens to 0, i.e. $Y_t = 0G + s_tH$

Building a cryptocurrency

We need

- A way to store the amount
- A way to prevent double spending (an ID, or serial#)
- A blinding factor for anonymity

Pedersen Commitment

- A Pedersen Commitment $X = aG + rH$ can store one secret value

- We need to store 2 secret values (the amount and serial#)

Pedersen Commitment

- A Pedersen Commitment $X = aG + rH$ can store one secret value

- We need to store 2 secret values (the amount and serial#)

$$X = aG + sH + \gamma F$$

amount **serial#** **blinding factor**

Spending a Coin

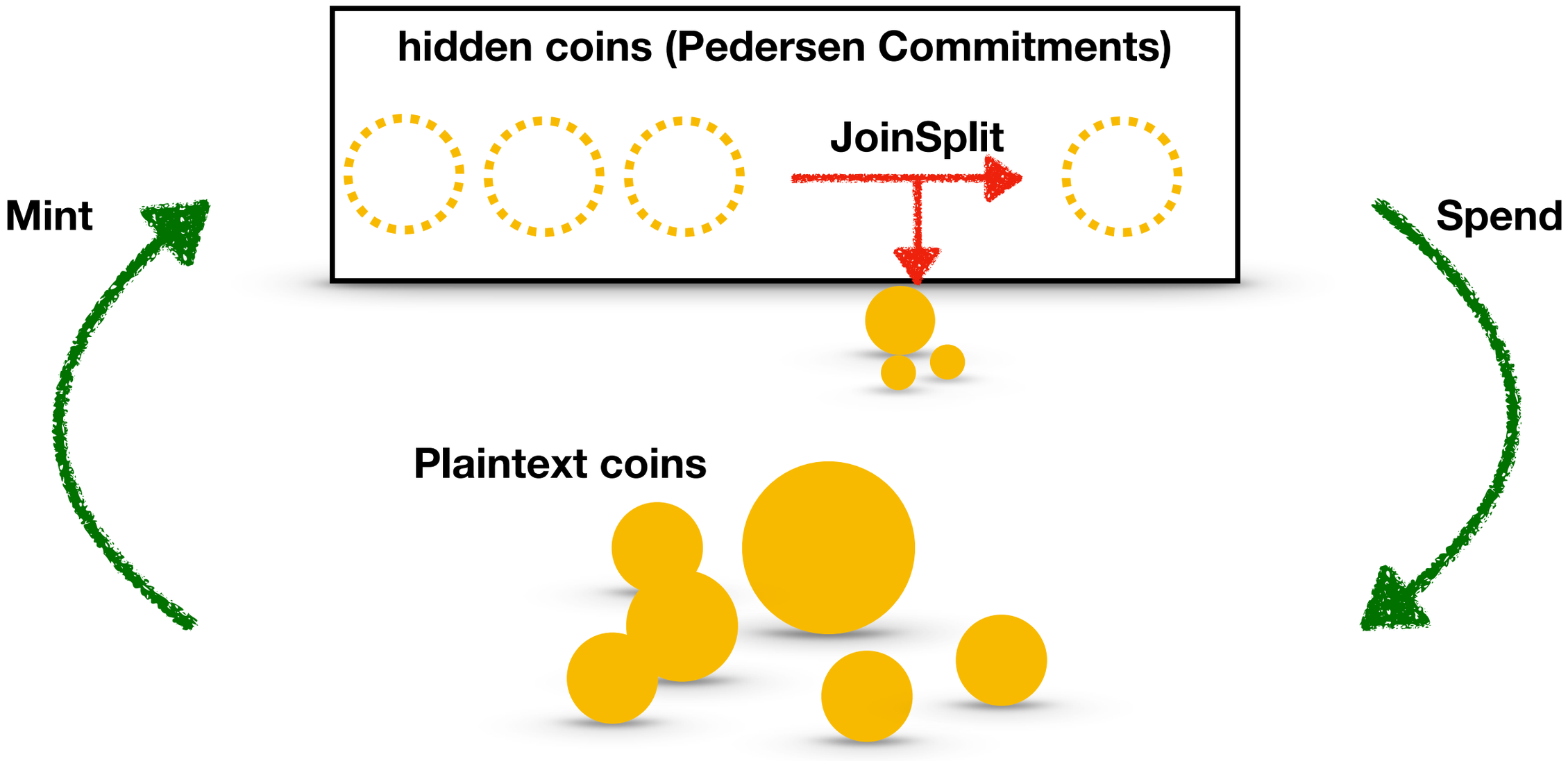
- Node publish serial# z (in plaintext) and anonymity set
- Validator can verify whether this serial numbers has been used before
- Validator creates $\mathcal{S} = \{X_i - zG \,|\, X_i \text{ in anonymity set}\}$
- Node publishes a 1-in-N proof that one of the X_i opens to 0, i.e. $X_i = aG + 0H + \gamma F$

Efficiency comparison

	Anonymity Set Size	Trusted Setup	Cryptographic Assumptions	Proof Size(KB)	Proof Time(s)	Verification Time(ms)
Monero	10	No	Well-tested	2.1	1	47
Zerocash	2^{32}	Yes	Relatively New	0.3	1-20	8
Zerocoin	$2^{1}3$	Yes	Well-tested	25	0.2	200
Lelantus 1	2^{14}	No	Well-tested	1.5	1.2	12 [1]
Lelantus 2	2^{16}	No	Well-tested	1.5	5.2	35 [2]

Lelantus

- Coins are Pedersen Commitments
  - Value, Serial number, blinding factor $X = vG + sH + \gamma F$



Lelantus Mint

- Delete a plaintext coin, create a hidden coin
  - Hidden Coins: $X = vG + sH + \gamma F$
- Publish a coin + proof that the value of the coin
  - Proof of knowledge of discrete logarithm

$$X - vG = sH + \gamma F$$

E.g. (c, d, α), so that

$$c = \mathscr{H}\left(G \,|\, H \,|\, F \,|\, c(X - vG) + dH + \alpha F\right)$$

Leleantus Spend

- Simply open the commitment (v, s, γ) to show that

$$X = vG + sH + \gamma F$$

- Amount v will be deposited to your account, ready to use

Leleantus JoinSplit

- Similarly to MimbleWimble transactions:

$$\mathsf{In}_1 + \ldots + \mathsf{In}_n - \mathsf{Out}_1 - \ldots - \mathsf{Out}_m - \underbrace{eG}_{\text{extra output}} = \underbrace{0G + 0H + \varepsilon F}_{\text{transaction kernel}}$$

1. For every input, present a 1-in-N Σ-protocol
   - publish Serial #, 1-in-N proof provides transaction input

$$\underbrace{c_1}_{=0} Y_1 + \underbrace{c_2}_{=0} Y_2 + \ldots + \underbrace{c_t}_{=1} Y_t + \ldots + \underbrace{c_n}_{=0} Y_n = \underbrace{Z}_{vG+0F+\gamma' F}$$

Leleantus JoinSplit

- Similarly to MimbleWimble transactions:

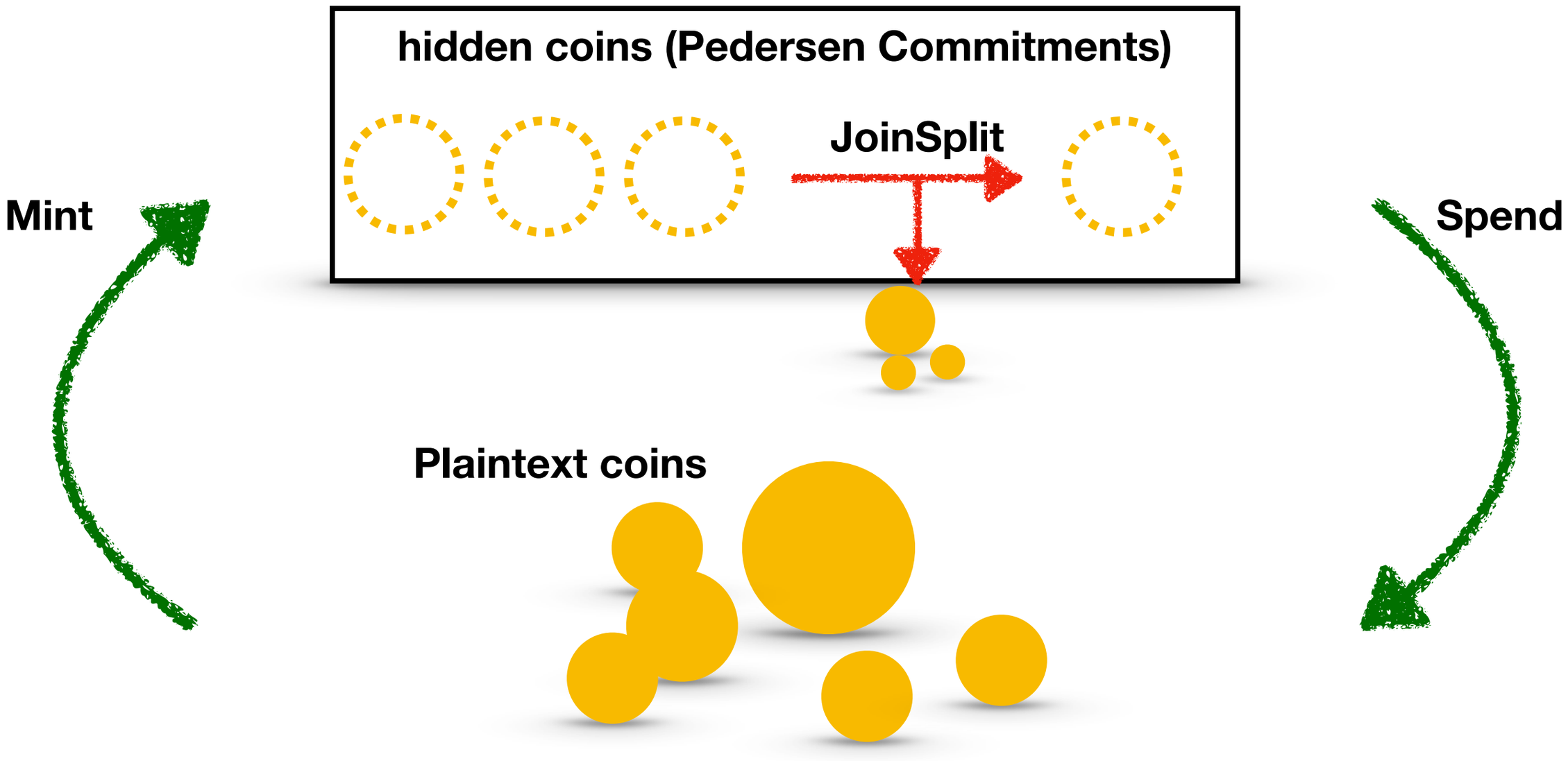
$$T = \mathsf{In}_1 + \ldots + \mathsf{In}_n - \mathsf{Out}_1 - \ldots - \mathsf{Out}_m - eG = \underbrace{0G + 0H + \varepsilon F}_{\text{transaction kernel}}$$

2. Proof that transaction kernel *only consists of Fs* with Schnorr Signature

$$(s, R), \text{ so that } \underbrace{sF}_{\text{only F}} = \underbrace{R}_{\text{only F}} + \mathscr{H}(R \,|\, T) \underbrace{T}_{\text{only F}}$$

Lelantus

- Coins are Pedersen Commitments
  - Value, Serial number, blinding factor $X = vG + sH + \gamma F$



1-in-N Σ-Protocol

Efficient encoding of the coefficients

Z{ero}{Coin|Cash}

- ZCash, Zerocoin, ZCoin all work somehow similarly
- 1-in-N proof that someone knows a token
- Double spend prevention via serial number

Z{ero}{Coin|Cash}

ZCoin

- used Lelantus

Z{ero}{Coin|Cash}

Zerocash

- Uses a Merkle Tree to store hidden coins

- 1-in-N proof is therefore aProof-of-Knowledge about an entry in the Merkle Tree

  - zk-SNARKS

Z{ero}{Coin|Cash}

Zerocoin (originally only fixed size values $X = sH + \gamma F$)

- Programming bugs, i.e. “=“ vs “==“, or insufficient checks to allow spending the same serial number twice

$$\gamma F = (\gamma + \underbrace{\qquad\qquad}_{\text{how many elements in curve}} \text{grouporder} \qquad\qquad)F$$

- Attack vector: Serial numbers can be chose freely. Bob sees Alice using a serial number, he can quickly mint and spend a coin with the same serial number. This makes the coin for Alice unusable

- April 2018: A unrecoverable cryptographic problem. Two ZK proofs were used:
  - (1) proof of knowledge of a minted coin
  - (2) proof of knowledge of a serial number
  - The part that joins these two proofs (that the coin known in (1) is the one with the serial#) was flawed

- Original Zerocoin stopped. Complete redo (also using zk-SNARKs) in the making

Appendix

Detailed description of the 1-in-N Σ-protocol using the binary representation of indices (25 = `00011001`)

1-in-N Σ−Protocol

- Problem:
  - $\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i = m_i G + s_i H$, $Y_t = 0G + s_t H$
- Efficient encoding of the coefficients:
  - Proof that you know a value t so that

$$c_1 Y_1 + c_2 Y_2 + \ldots + c_n Y_n \text{ opens to } 0$$

  - Assume each index i is given in binary format
    - (i.e. i=0110101)

1-in-N $\Sigma-$Protocol

- New Problem:
  - $\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i = m_i G + s_i H$, $Y_t = 0G + s_t H$
- Efficient approach:
  - represent the index i in binary form, i.e. $t = 11001$
  - for each digit a separate variable $c_0 c_1 c_2 c_3 c_4 c_5$
    - Instead of N secret {0,1} coefficients, only $O(\log(N))$
  - Details are more complex, at the end of the lecture (if time permits)

Indices

	Binary Representation				
Y_{10}	0	1	0	1	0
Y_{17}	1	0	0	0	1
Y_{20}	1	0	1	0	0
Y_{25}	1	1	0	0	1
Y_{27}	1	1	0	1	1
Y_{28}	1	1	1	0	0

Anonymity Set

Our Element

Kronecker-Delta

Binary Representation

Y_{25}	1	1	0	0	1
Y_{10} 01010	0	1	1	0	0
Y_{17} 10001	1	0	1	1	1
Y_{20} 10100	1	0	0	1	0
Y_{25} 11001	1	1	1	1	1
Y_{27} 11011	1	1	1	0	1
Y_{28} 11100	1	1	0	1	0

Anonymity Set

Our Element

Same as our element (1) or not (0)

Kronecker-Delta

Anonymity Set

	Binary Representation					
Y_{25}	1	1	0	0	1	Product
Y_{10} 01010	0	1	1	0	0	0
Y_{17} 10001	1	0	1	1	1	0
Y_{20} 10100	1	0	0	1	0	0
Y_{25} 11001	1	1	1	1	1	1
Y_{27} 11011	1	1	1	0	1	0
Y_{28} 11100	1	1	0	1	0	0

Our Element

Product of all numbers in each line

Kronecker-Delta

		Binary Representation				
Y_{25}		**1**	**1**	**0**	**0**	**1**
Y_{17}	**10001**	**1** $\delta(1,17_1)$	**0** $\delta(2,17_2)$	**1** $\delta(3,17_3)$	**1** $\delta(4,17_4)$	**1** $\delta(5,17_5)$
Y_{20}	**10100**	**1** $\delta(1,20_1)$	**0** $\delta(2,20_2)$	**0** $\delta(3,20_3)$	**1** $\delta(4,20_4)$	**0** $\delta(5,20_5)$

Define $\delta(k, l_k)$ as the agreement in the k^{th} digit between

- The index of the element we own
- The index of the element in the anonymity set

Kronecker-Delta

Define $\delta(k, l_k)$ as the agreement in the k^{th} digit between

- The index of the element we own
- The index of the element in the anonymity set

The product of the values in each line

$$\delta_l = \prod_k \delta(k, l_k) = \delta(1,l_1) \cdot \delta(2,l_2) \cdot \delta(3,l_3) \cdot \delta(4,l_4) \cdot \delta(5,l_5)$$

is the indicator function of our secret element

Kronecker-Delta

Define $\delta(k, l_k)$ as the agreement in the k^{th} digit between

- The index of the element we own
- The index of the element in the anonymity set

The product of the values in each line

$$\delta_l = \prod_k \delta(k, l_k) = \delta(1,l_1) \cdot \delta(2,l_2) \cdot \delta(3,l_3) \cdot \delta(4,l_4) \cdot \delta(5,l_5)$$

is the indicator function of our secret element.

- $\delta_l = 1$ only for our element

Kronecker-Delta

Define $\delta(k, l_k)$ as the agreement in the k^{th} digit between

- The index of the element we own
- The index of the element in the anonymity set

We need

- $\delta(1,0)$ for the first digit to be 0
- $\delta(1,1)$ for the first digit to be 1
- $\delta(2,0)$ for the second digit to be 0
- $\delta(2,1)$ for the second digit to be 1
- $\delta(3,0)$ for the third digit to be 0
- $\delta(3,1)$ for the third digit to be 1
- ...

Number of values:
$O\left(\log(N)\right)$

1-in-N Σ—Protocol

$\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i = m_i G + s_i H$, $Y_t = 0G + s_t H$

Given $\delta(k, i_k)$, $\delta_l = \prod_k \delta(k, l_k)$. Show that

1. Each δ_l is 0 or 1
2. $\delta_1 Y_1 + \delta_2 Y_2 + \ldots + \delta_n Y_n$ opens to 0

1-in-N $\Sigma-$Protocol

Let's focus on $\delta_l = \prod_k \delta(k, l_k)$

- Using the 0/1 protocol, we hide for each digit the value $\delta(k,0)$ with

$$f_{l,0} = \delta(k,0)x + a_{l,0}$$

We have

- $f_{0,0}$ for the first digit to be 0
- $f_{1,0}$ for the second digit to be 0
- $f_{3,0}$ for the third digit to be 0
- …

$\delta(k,0)$ is 1 if the k^{th}digit is 0
$\delta(k,0)$ is 0 if the k^{th}digit is 1

We have $f_{0,1} = x - f_{0,0}$

1-in-N Σ−Protocol

Let's focus on $\delta_l = \prod_k \delta(k, l_k)$

- hide
- $\delta(k,0)$ in $f_{l,0} = \delta(k,0)x + a_{l,0}$
- $\delta(k,1)$ in $f_{l,1} = x - f_{l,0}$

Instead of $\delta_l = \prod_k \delta(k, l_k)$, consider the product $p_l(x) = \prod_k f_{k,l_k}$

1-in-N $\Sigma-$Protocol

Let's focus on $\delta_l = \prod_k \delta(k, l_k)$

$\delta(k,0)$ in $f_{l,0} = \delta(k,0)x + a_{l,0}$

$\delta(k,1)$ in $f_{l,1} = x - f_{l,0}$

The product $p_l(x) = \prod_k f_{k,l_k}$ is

m is the number of binary digits

$$p_l(x) = \prod_k \left(\delta(k, l_k)x + a_{k,l_k} \right) = \delta_l x^m + \sum_k^{m-1} p_{l,k} x^k$$

1-in-N Σ—Protocol

$$p_l(x) = \prod_k \left(\delta(k, l_k)x + a_{k,l_k} \right) = \delta_l x^m + \sum_k^{m-1} p_{l,k} x^k$$

- The value δ_l is the secret parameter we need
  - 1 for our own element, 0 for everything else

1-in-N $\Sigma-$Protocol

$$p_l(x) = \prod_k \left(\delta(k, l_k)x + a_{l,k}\right) = \delta_l x^m + \sum_k^{m-1} p_{l,k} x^k$$

- The other term $p_{l,k}$ are independent of challenge x
  - Can be computed ahead of time
  - Can be transmitted as Pedersen Commitments

1-in-N Σ−Protocol

$\{Y_1, Y_2, \ldots, Y_n\}$, $Y_i = m_i G + s_i H$, $Y_t = 0G + s_t H$

1. Generate random values ρ_k and compute $p_{l,k}$

   Transmit $Q_k = \sum_i p_{l,k} Y_i + \rho_k H$ (Pedersen Comm)

2. For all $\delta(k, l_k)$ values start a separate Σ-protocol

   - For an anonymity set of 1024, e.g., we need only 10 parallel 0/1 zero-knowledge proofs
   - Results in $f_{k,0} = \delta(k,0)x + a_{k,0}$
     - A commitment for the k^{th} digit to be 0

1-in-N Σ—Protocol

$$\{Y_1, Y_2, \ldots, Y_n\},\ Y_i = m_i G + s_i H,$$

$$Y_t = 0G + s_t H$$

$f_{k,2_k}$ is the $f_{k,0}$ or $f_{k,1}$, depending on whether the k^{th} digit of the second index is 0 or 1

3. Send $z_d = s_t x^n - \sum_k^{n-1} \rho_k x^k$

4. Verifier checks:

$$\left(\prod_k f_{k,1_k}\right) Y_1 + \left(\prod_k f_{k,2_k}\right) Y_2 + \ldots + \left(\prod_k f_{k,n_k}\right) Y_n + \sum_k^{n-1} x^{-k} Q_k = 0G + z_d H$$

Complete Description

ck is commitment key, i.e. two group elements G, H

$\mathcal{P}(ck, (c_0, \ldots, c_{N-1}), (\ell, r))$		$\mathcal{V}(ck, (c_0, \ldots, c_{N-1}))$
For $j = 1, \ldots, n$ $r_j, a_j, s_j, t_j, \rho_k \leftarrow \mathbb{Z}_q$ $c_{\ell_j} = \mathrm{Com}_{ck}(\ell_j; r_j)$ $c_{a_j} = \mathrm{Com}_{ck}(a_j; s_j)$ $c_{b_j} = \mathrm{Com}_{ck}(\ell_j a_j; t_j)$ $c_{d_k} = \prod_i c_i^{p_{i,k}} \mathrm{Com}_{ck}(0; \rho_k)$ using $k = j - 1$ and $p_{i,k}$ from (1)	$c_{\ell_1}, c_{a_1}, c_{b_1}, c_{d_0}, \ldots,$ $c_{\ell_n}, c_{a_n}, c_{b_n}, c_{d_{n-1}} \longrightarrow$	Accept if and only if $c_{\ell_1}, \ldots, c_{d_{n-1}} \in \mathcal{C}_{ck}$ $f_1, \ldots, z_d \in \mathbb{Z}_q$ For all $j \in \{1, \ldots, n\}$ $c_{\ell_j}^x c_{a_j} = \mathrm{Com}_{ck}(f_j; z_{a_j})$ $c_{\ell_j}^{x - f_j} c_{b_j} = \mathrm{Com}_{ck}(0; z_{b_j})$ $\prod_i c_i^{\prod_{j=1}^n f_{j,i_j}} \cdot \prod_{k=0}^{n-1} c_{d_k}^{-x^k}$ $= \mathrm{Com}_{ck}(0; z_d)$ using $f_{j,1} = f_j$ and $f_{j,0} = x - f_j$
	$\longleftarrow x \leftarrow \{0,1\}^\lambda$	
For $j = 1, \ldots, n$ $f_j = \ell_j x + a_j$ $z_{a_j} = r_j x + s_j$ $z_{b_j} = r_j (x - f_j) + t_j$ $z_d = r x^n - \sum_{k=0}^{n-1} \rho_k x^k$	$f_1, z_{a_1}, z_{b_1}, \ldots,$ $f_n, z_{a_n}, z_{b_n}, z_d \longrightarrow$	

Commitment $Com_{ck}(x, y) = xG + yH$

Multiplicative notation $aG \mapsto g^a$