SNARK Flipper

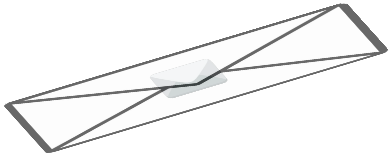
Flip and prove multiple instances efficiently

Joint work with
Anca Nitulescu, Carla Rafóls

Nikitas Paslis
Universitat Pompeu Fabra

In Brief

Flip := Folding via IPP



What are SNARKs?

zk-SNARK

Zero-Knowledge
does not leak anything about the witness

Succinctness
proof size independent of NP witness size

zk-SNARK

Non-Interactivity
no exchange between prover and verifier

Knowledge Soundness
a witness can be efficiently extracted from the prover

Argument
soundness holds only against computationally bounded provers

Applications?

Proof of storage

Storage Providers

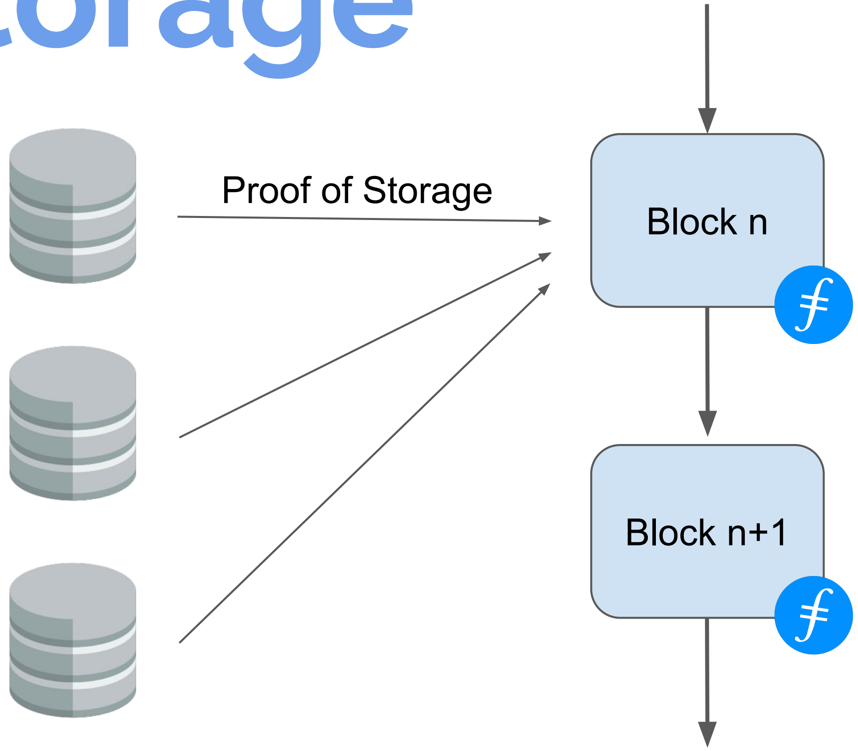
- onboard storage capacity
- earn block rewards
- regularly prove the storage

= **Provers**

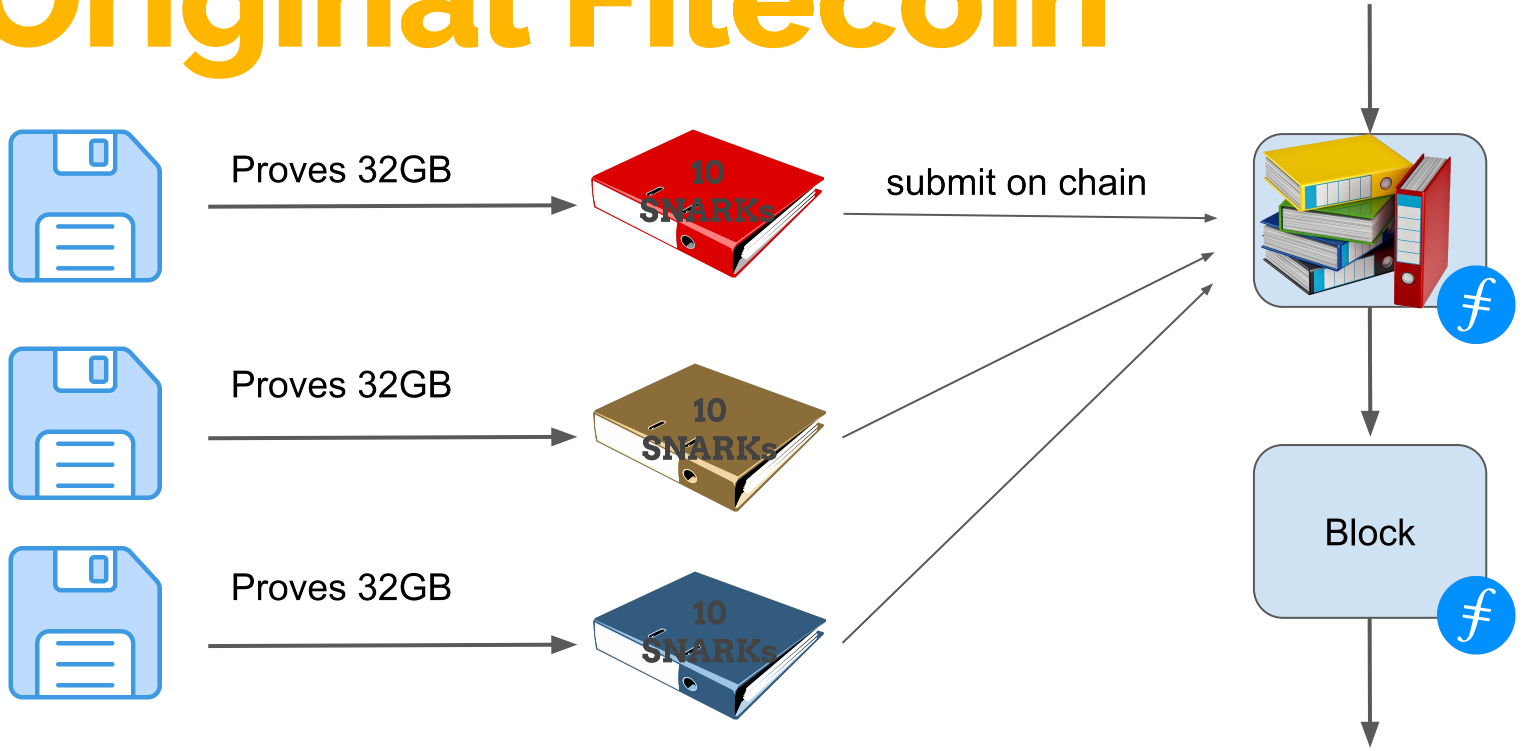
Nodes in network

- ensure data is being stored, maintained, and secured
- need to check proofs of space

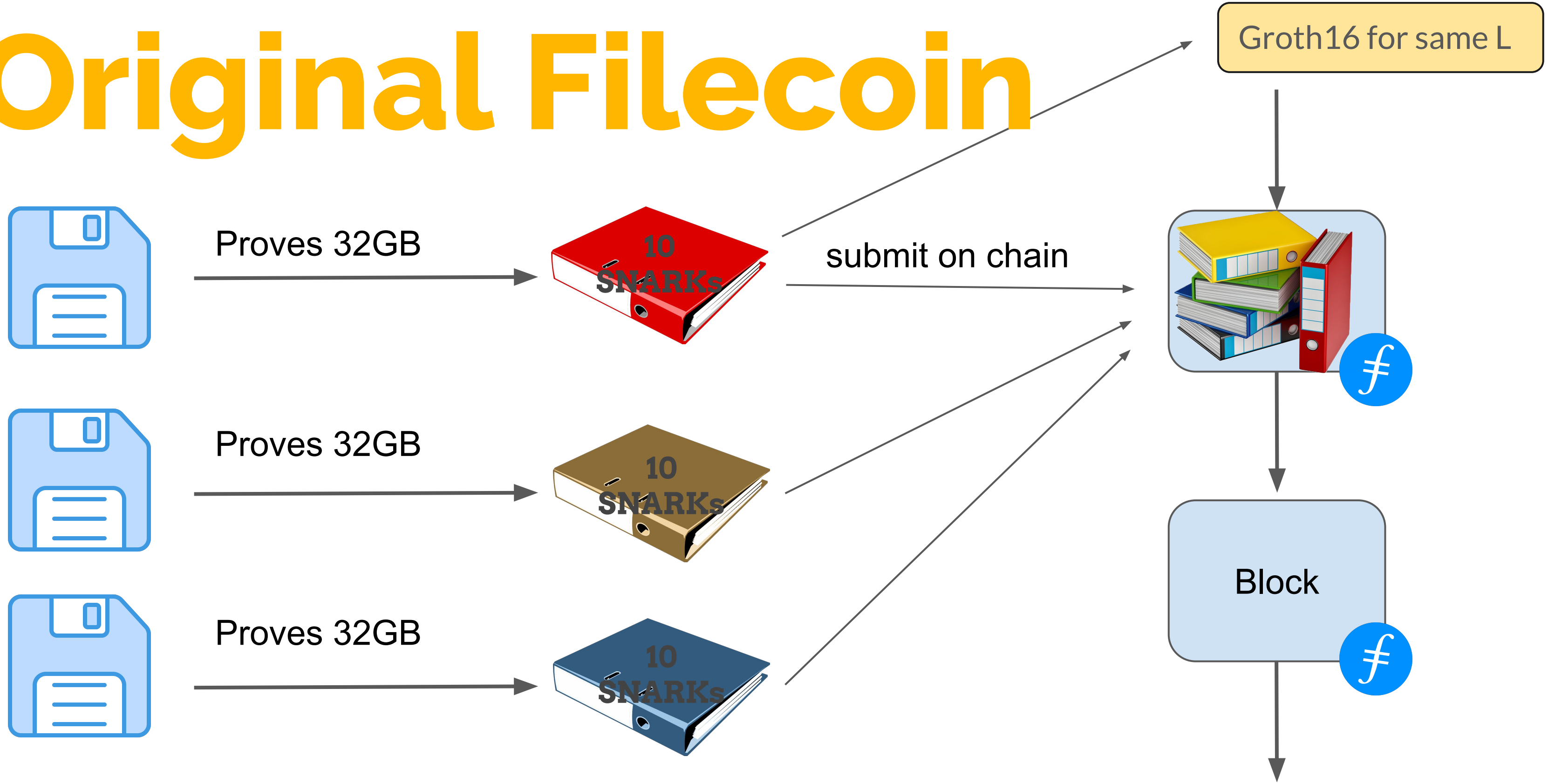
= **Verifiers**



Original Filecoin



Original Filecoin



How it works?

Groth16

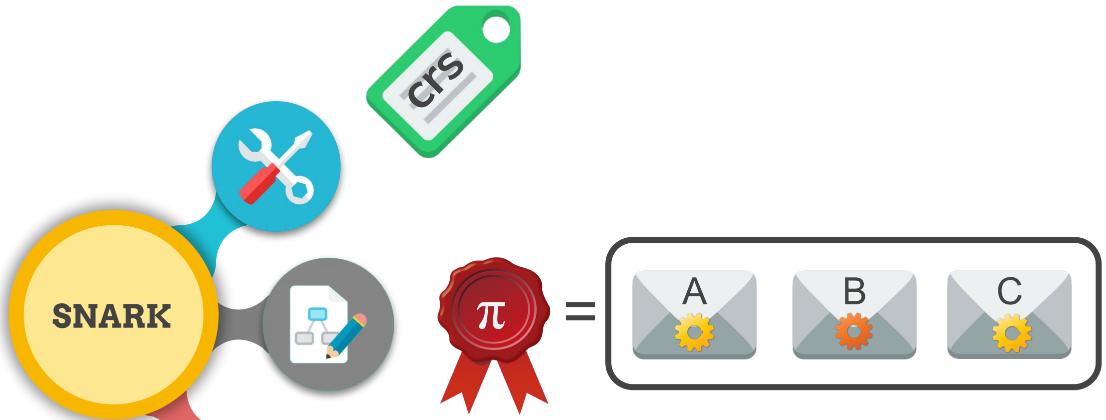
Bilinear Groups

$$\langle g \rangle = \mathbb{G}_1, \ \langle h \rangle = \mathbb{G}_2$$

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

$$e(g^a, h^b) = e(g, h)^{ab}$$

Groth16



$e(A, B) = e(C, D)$

Bilinear Groups

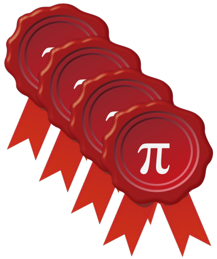
$\langle g \rangle = \mathbb{G}_1, \ \langle h \rangle = \mathbb{G}_2$

$e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$

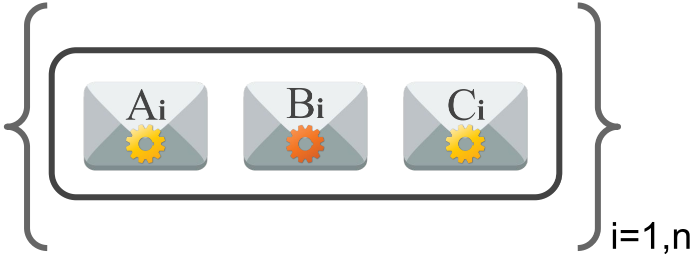
$e(g^a, h^b) = e(g, h)^{ab}$

Many SNARKs

Proofs



=



$e(A_1, B_1) = e(C_1, D)$

$e(A_2, B_2) = e(C_2, D)$

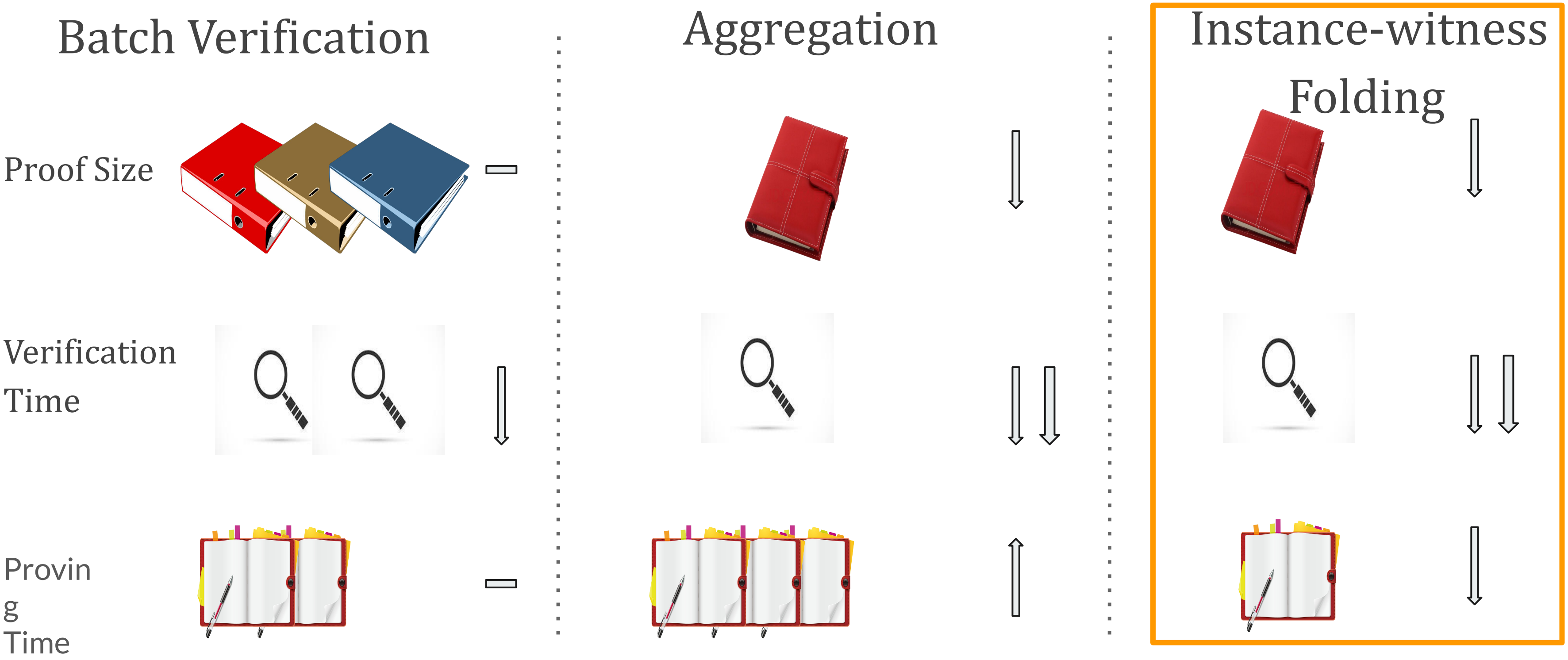
…

$e(A_n, B_n) = e(C_n, D)$

Verify many SNARKs

	Batch Verification		Aggregation		Instance-witness Folding	
Proof Size						
Verification Time						
Provin g Time						

Verify many SNARKs

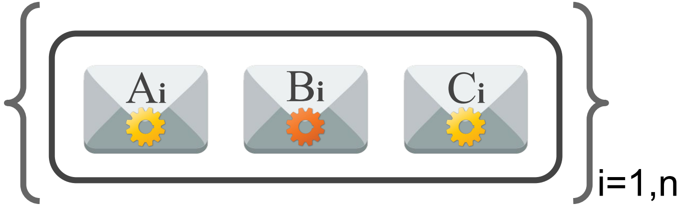


SnarkPack Aggregation

(Gailly, Maller, Nitulescu)

SNARK Aggregation

Aggregation



$$Z_{AB} = \prod e(A_i, B_i^{r_i})$$

$$Z_C = \prod C_i^{r_i}$$

$$\Longrightarrow \quad Z_{AB} = Z_C$$

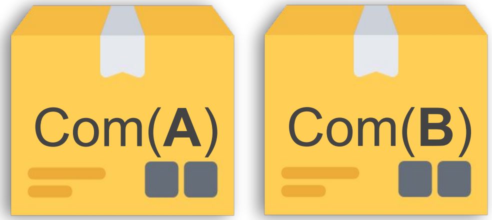
Tools: GIPP

Proofs for Inner Pairing Products and Applications - Bünz, Maller, Mishra, Tyagi, Vesely

$$\langle \mathbf{A}, \mathbf{b} \rangle = \prod A_i^{b_i}$$

$$\langle \mathbf{A}, \mathbf{B} \rangle = \prod e(A_i, B_i)$$

$A_i \in \mathbb{G}_1$, $B_i \in \mathbb{G}_2$, $b_i \in \mathbb{Z}_q$



Split & Collapse

$$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1)\, e(A_2, B_2) \ldots e(A_n, B_n)$$

$$\mathbf{A} = (A_1, A_2, \ldots A_n)$$

$$\mathbf{B} = (B_1, B_2 \ldots B_n)$$

Split & Collapse

$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1)\, e(A_2, B_2) \ldots e(A_n, B_n)$

$\mathbf{A} = (A_1, A_2, \ldots A_n)$

$\mathbf{B} = (B_1, B_2 \ldots B_n)$

$\mathbf{A}_{left}$ $\mathbf{A}_{right}$

$\mathbf{B}_{left}$ $\mathbf{B}_{right}$

Split & Collapse

$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1)\, e(A_2, B_2) \ldots e(A_n, B_n)$

$\mathbf{A} = (A_1, A_2, \ldots A_n)$

$\mathbf{B} = (B_1, B_2 \ldots B_n)$

$\mathbf{A}_{left}$ $\mathbf{A}_{right}$

$\mathbf{B}_{left}$ $\mathbf{B}_{right}$

$\mathbf{L} = \langle \mathbf{A}_{right}, \mathbf{B}_{left} \rangle$

$\mathbf{R} = \langle \mathbf{A}_{left}, \mathbf{B}_{right} \rangle$

Split & Collapse

$$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1)\, e(A_2, B_2) \ldots e(A_n, B_n)$$

$\mathbf{A} = (A_1, A_2, \ldots A_n)$

$\mathbf{B} = (B_1, B_2 \ldots B_n)$

$\mathbf{A}_{left}$ $\mathbf{A}_{right}$

$\mathbf{B}_{left}$ $\mathbf{B}_{right}$

$\mathbf{A'} = (A'_1, \ldots A'_{n/2})$

$\mathbf{B'} = (B'_1, \ldots B'_{n/2})$

Split & Collapse

$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1)\, e(A_2, B_2) \ldots e(A_n, B_n)$

$\mathbf{A} = (A_1, A_2, \ldots A_n)$

$\mathbf{B} = (B_1, B_2 \ldots B_n)$

$\langle \mathbf{A'}, \mathbf{B'} \rangle$

$\mathbf{A'} = (A'_1, \ldots A'_{n/2})$

$\mathbf{B'} = (B'_1, \ldots B'_{n/2})$

Fast Verification

Com(**A**),Com(**B**)
Com(**A'**),Com(**B'**)
...

$\rightarrow$

$Z_{AB} = \langle \mathbf{A}, \mathbf{B} \rangle$

Fast Verification

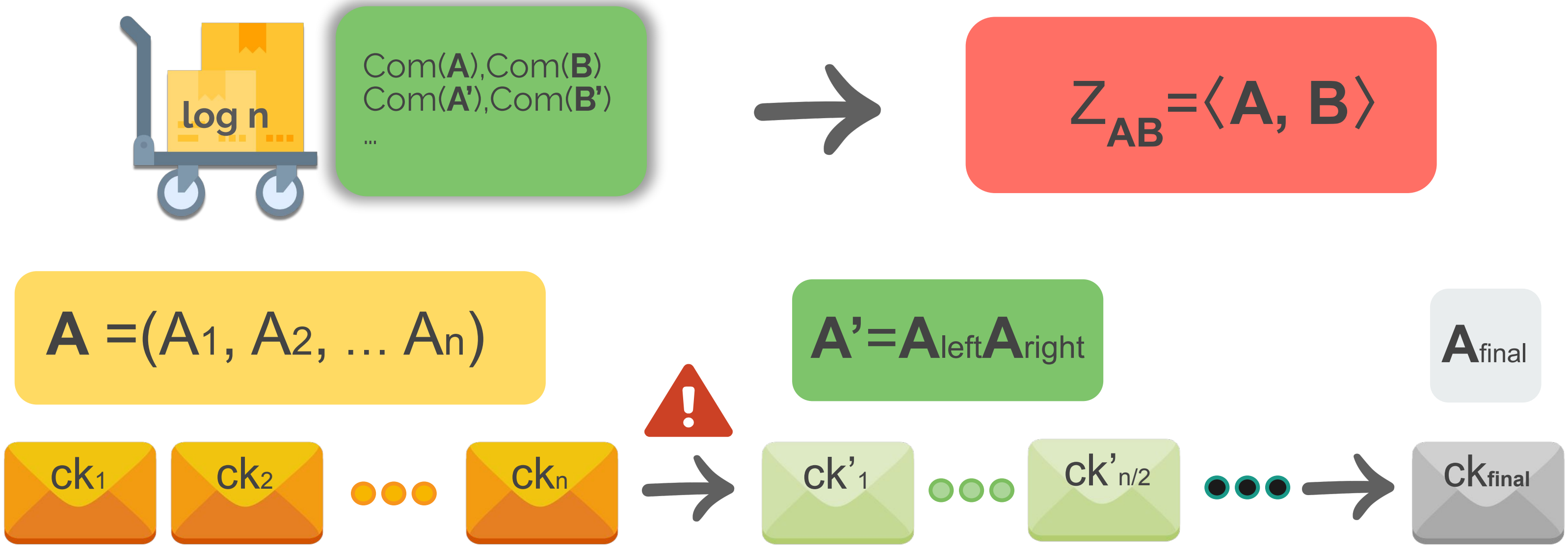
Com(**A**),Com(**B**)
Com(**A'**),Com(**B'**)
...

→

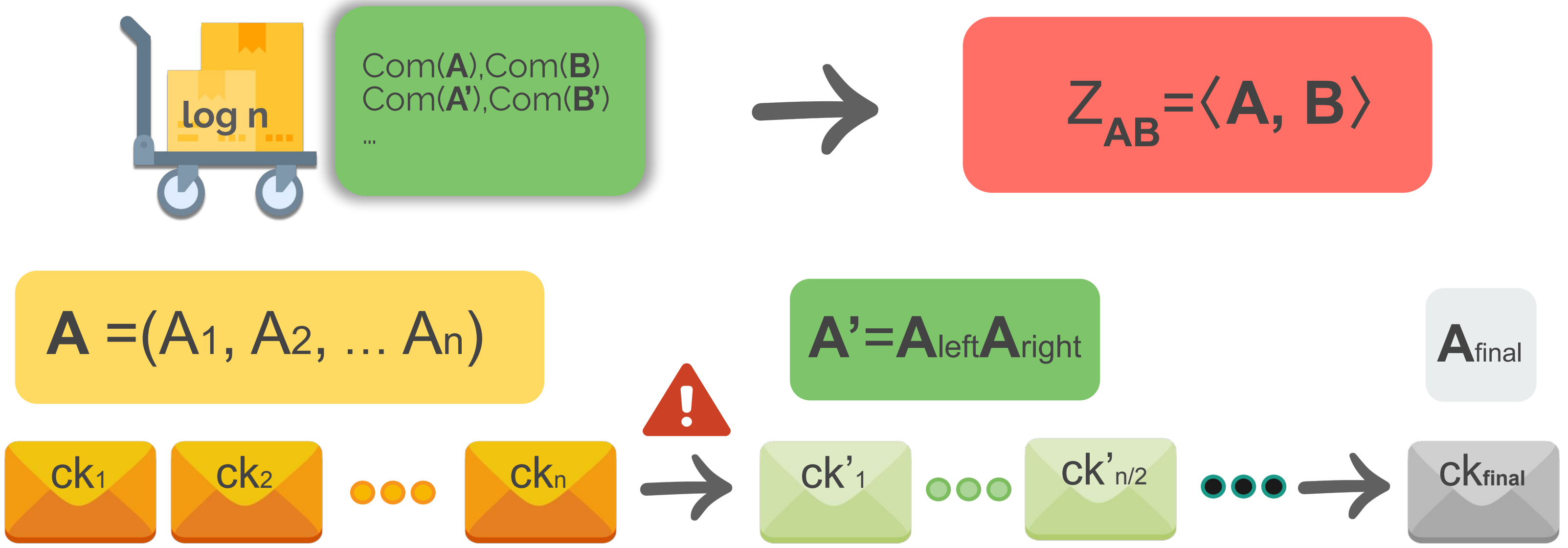
$Z_{AB}=\langle \mathbf{A}, \mathbf{B}\rangle$

$\mathbf{A} = (A_1, A_2, \ldots A_n)$ → $\mathbf{A'} = \mathbf{A}_{left}\mathbf{A}_{right}$ ••• → $\mathbf{A}_{final}$

Fast Verification

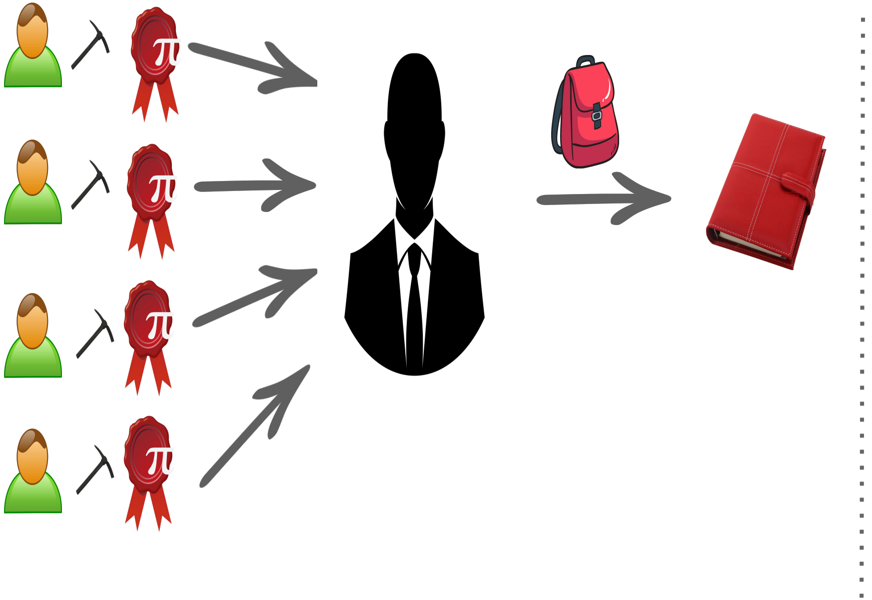


Fast Verification

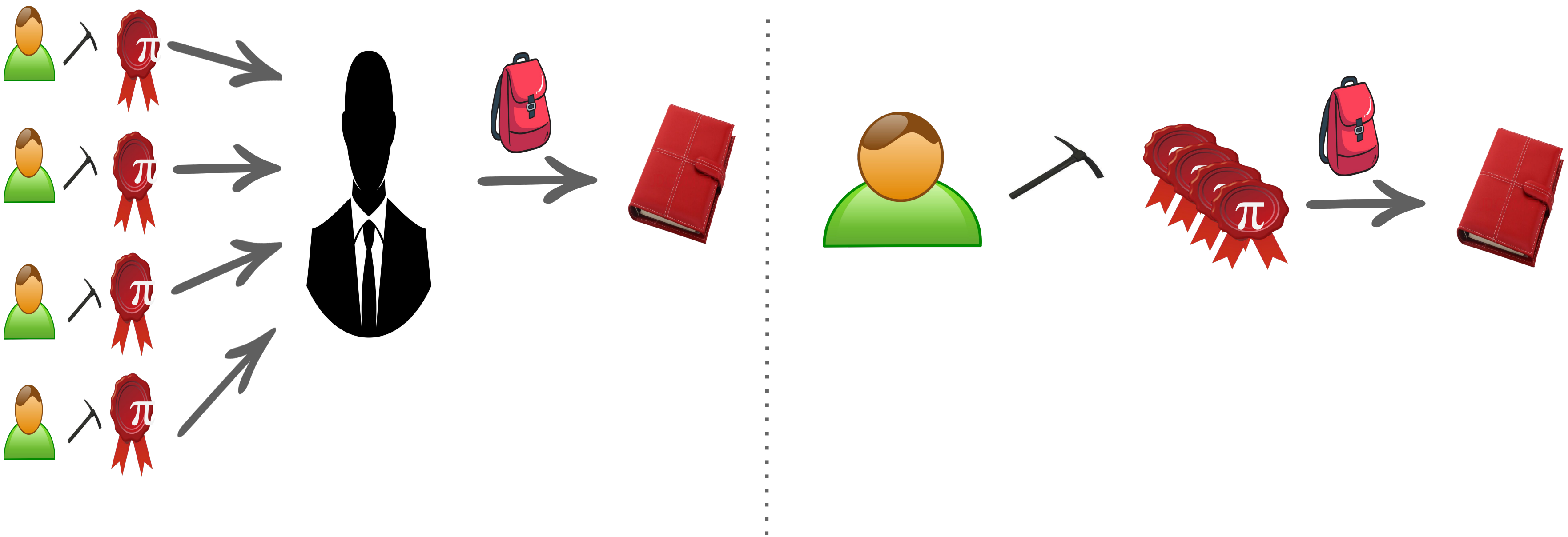


Transparent key -> linear verifier   Structured key -> logarithmic verifier

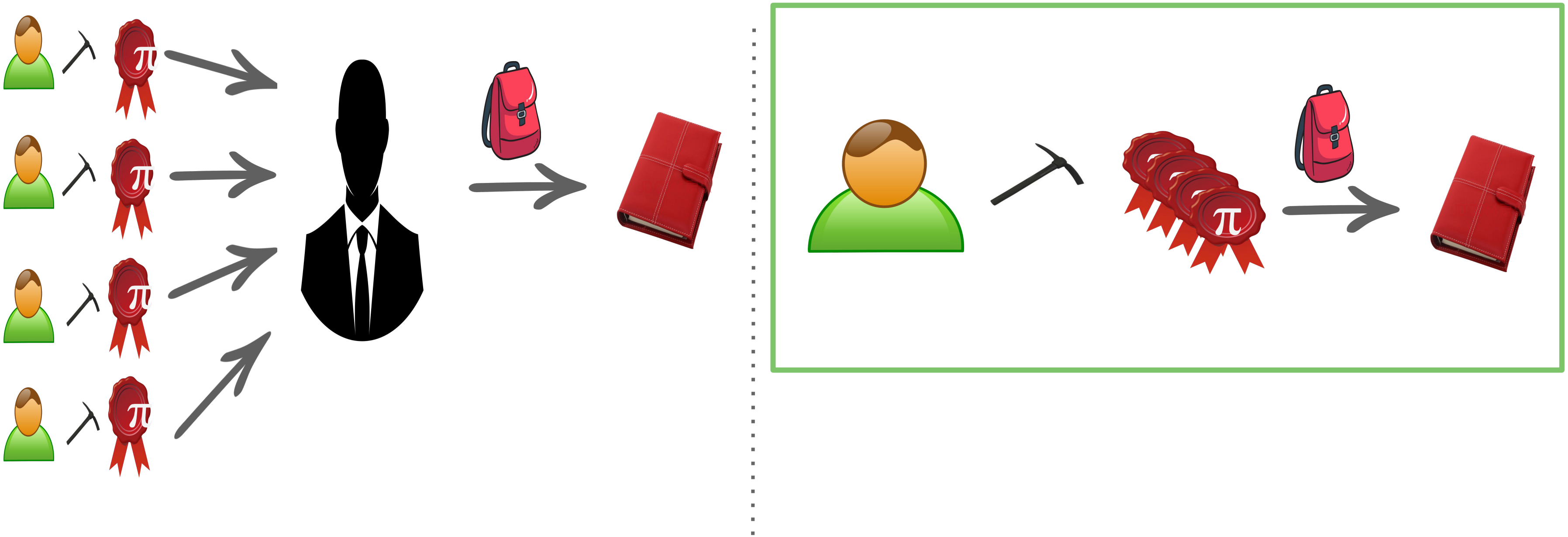
SnarkPack aggregation

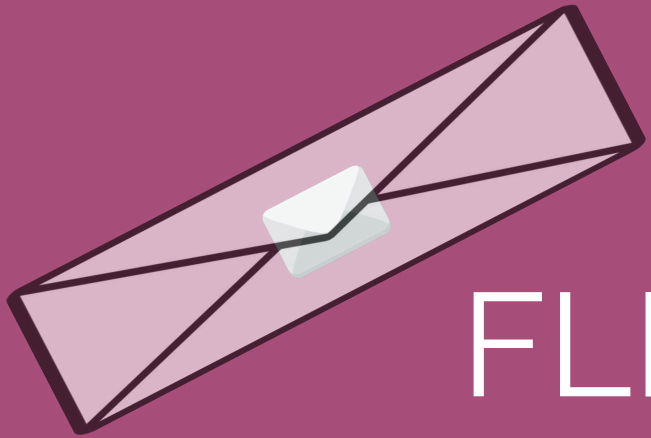


SnarkPack aggregation



SnarkPack aggregation





FLIP

R1CS Extensions

Relaxed

$$\mathcal{L}^{\text{relaxed}}_{\mathbf{A},\mathbf{B},\mathbf{C}} = \{(u, \mathbf{x}, \mathbf{e}) \in \mathbb{F} \times \mathbb{F}^l \times \mathbb{F} \mid \exists \mathbf{w} \in \mathbb{F}^{m-l} \text{ s.t. } \mathbf{z} = \begin{pmatrix} u \\ \mathbf{x} \\ \mathbf{w} \end{pmatrix} \wedge \mathbf{A}\mathbf{z} \circ \mathbf{B}\mathbf{z} = u\mathbf{C}\mathbf{z} + \mathbf{e}\}$$

R1CS Extensions

Relaxed

$$\mathcal{L}^{\text{relaxed}}_{\mathbf{A},\mathbf{B},\mathbf{C}} = \{(u, \mathbf{x}, \mathbf{e}) \in \mathbb{F} \times \mathbb{F}^l \times \mathbb{F} \mid \exists \mathbf{w} \in \mathbb{F}^{m-l} \text{ s.t.}$$

$$\mathbf{z} = \begin{pmatrix} u \\ \mathbf{x} \\ \mathbf{w} \end{pmatrix} \wedge \mathbf{Az} \circ \mathbf{Bz} = u\mathbf{Cz} + \mathbf{e}\}$$

u=1 + **e=0** → R1CS

R1CS Extensions

Relaxed

$$\mathcal{L}^{\text{relaxed}}_{\mathbf{A},\mathbf{B},\mathbf{C}} = \{(u, \mathbf{x}, \mathbf{e}) \in \mathbb{F} \times \mathbb{F}^l \times \mathbb{F} \mid \exists \mathbf{w} \in \mathbb{F}^{m-l} \text{ s.t.}$$
$$\mathbf{z} = \begin{pmatrix} u \\ \mathbf{x} \\ \mathbf{w} \end{pmatrix} \wedge \mathbf{Az} \circ \mathbf{Bz} = u\mathbf{Cz} + \mathbf{e}\}$$

u=1 + e=0 → R1CS

Committed Relaxed

$$\mathcal{L}^{\text{c-relaxed}}_{\mathsf{ck}_1,\mathbf{A},\mathbf{B},\mathbf{C}} = \{(u, \mathbf{x}, [e]_1, [w]_1) \in \mathbb{F} \times \mathbb{F}^l \times \mathcal{C}^2 \mid \exists (\mathbf{w}, \mathbf{e}) \in \mathbb{F}^{m-l} \times \mathbb{F}^m \text{ s.t.}$$
$$[w]_1 = \mathsf{Com}_{\mathsf{ck}_1}(\mathsf{pp}, \mathbf{w}) \wedge [e]_1 = \mathsf{Com}_{\mathsf{ck}_1}(\mathsf{pp}, \mathbf{e}) \wedge$$
$$((u, \mathbf{x}, \mathbf{e}), \mathbf{w}) \in \mathcal{R}^{\text{relaxed}}_{\mathbf{A},\mathbf{B},\mathbf{C}}\}$$

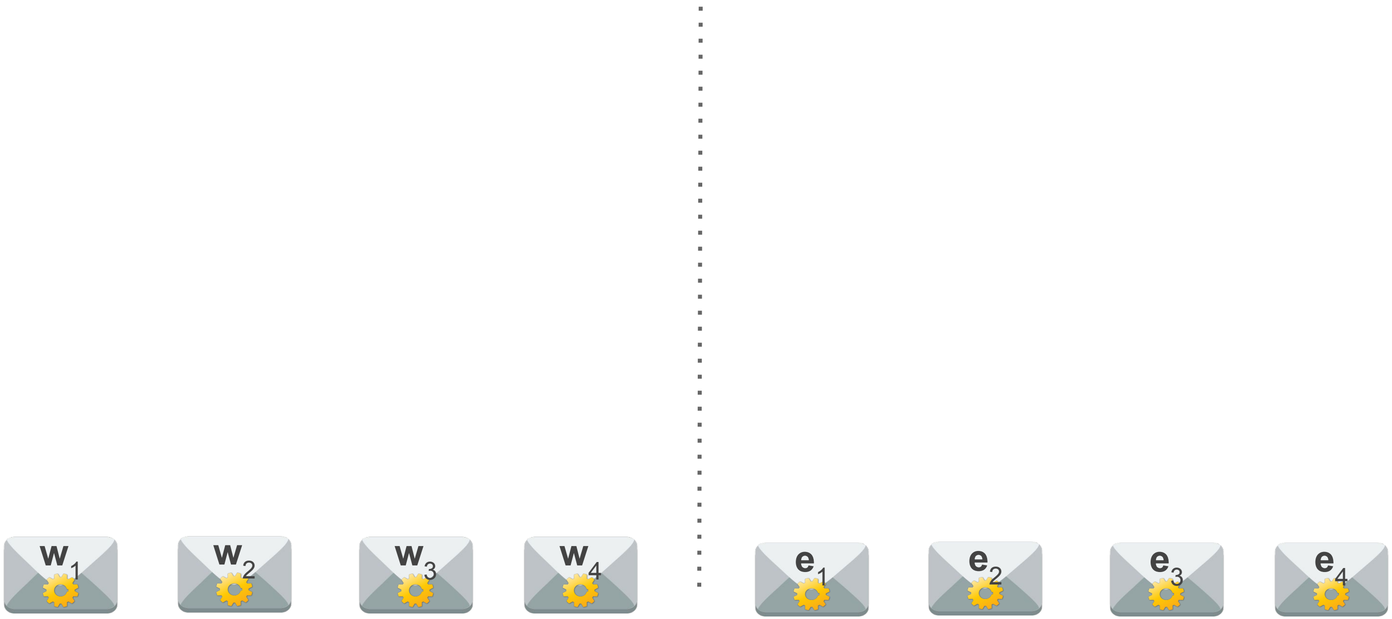
Nova style Folding

$i \in \{1,2\}$: $x_i = (\mathbf{x}_i, u_i, [e_i]_1, [w_i]_1)$, $w_i = (\mathbf{w}_i, \mathbf{e}_i)$

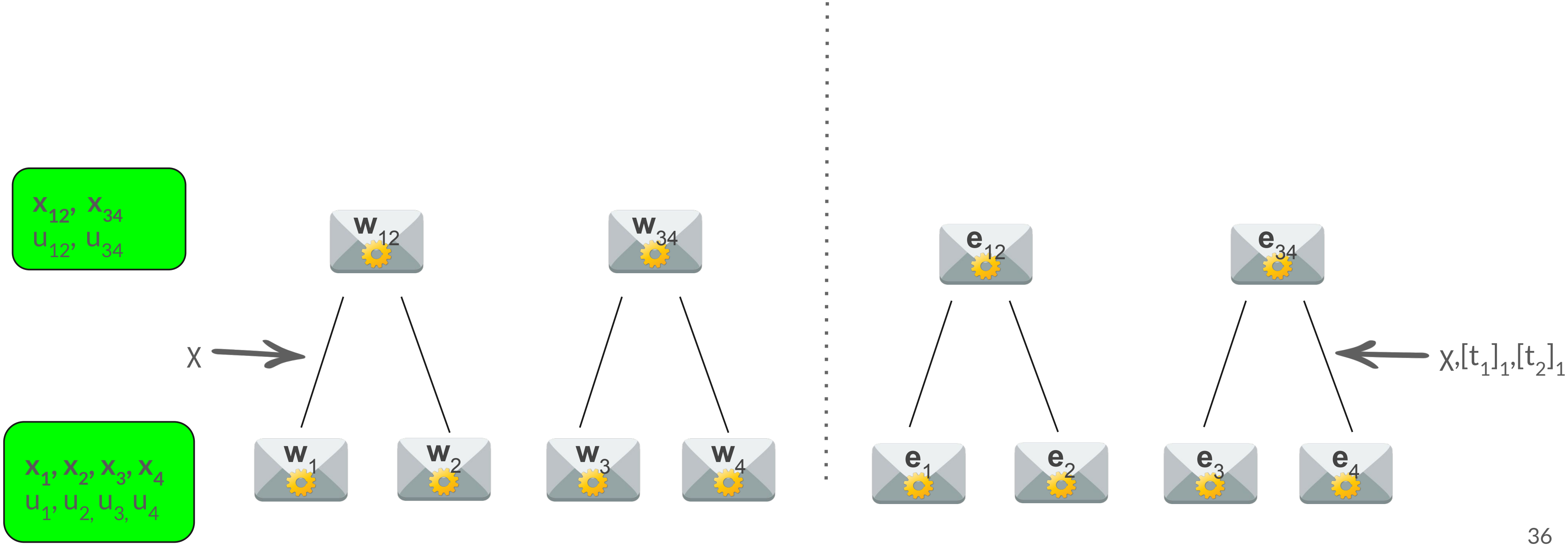
$$
\begin{array}{lcl}
\mathsf{P}: \ q_i = (x_i, w_i) & & \mathsf{V}: \ x_i \\
\hline
\mathbf{z}_i = (u_i, \mathbf{x}_i^\top, \mathbf{w}_i^\top)^\top & & \\
\mathbf{t} = \mathbf{A}\mathbf{z}_1 \circ \mathbf{B}\mathbf{z}_2 + \mathbf{A}\mathbf{z}_2 \circ \mathbf{B}\mathbf{z}_1 & & \\
\qquad - u_1\mathbf{C}\mathbf{z}_2 - u_2\mathbf{C}\mathbf{z}_1 & & \\
[t]_1 = \mathsf{Com}_{\mathsf{ck}_1}(\mathrm{pp}_1, \mathbf{t}) & \xrightarrow{\quad [t]_1 \quad} & \\
 & \xleftarrow{\quad \chi \quad} & \chi \leftarrow \mathbb{F} \\
\mathbf{e} = \mathbf{e}_1 + \chi\mathbf{t} + \chi^2\mathbf{e}_2 & & [e]_1 = [e_1]_1 + \chi[t] + \chi^2[e_2]_1 \\
\mathbf{w} = \mathbf{w}_1 + \chi\mathbf{w}_2 & & [w]_1 = [w_1]_1 + \chi[w_2]_1 \\
 & & u = u_1 + \chi u_2 \\
 & & \mathbf{x} = \mathbf{x}_1 + \chi\mathbf{x}_2 \\
w = (\mathbf{w}, \mathbf{e}) & & x = (u, \mathbf{x}, [e]_1, [w]_1)
\end{array}
$$

Nova style Folding

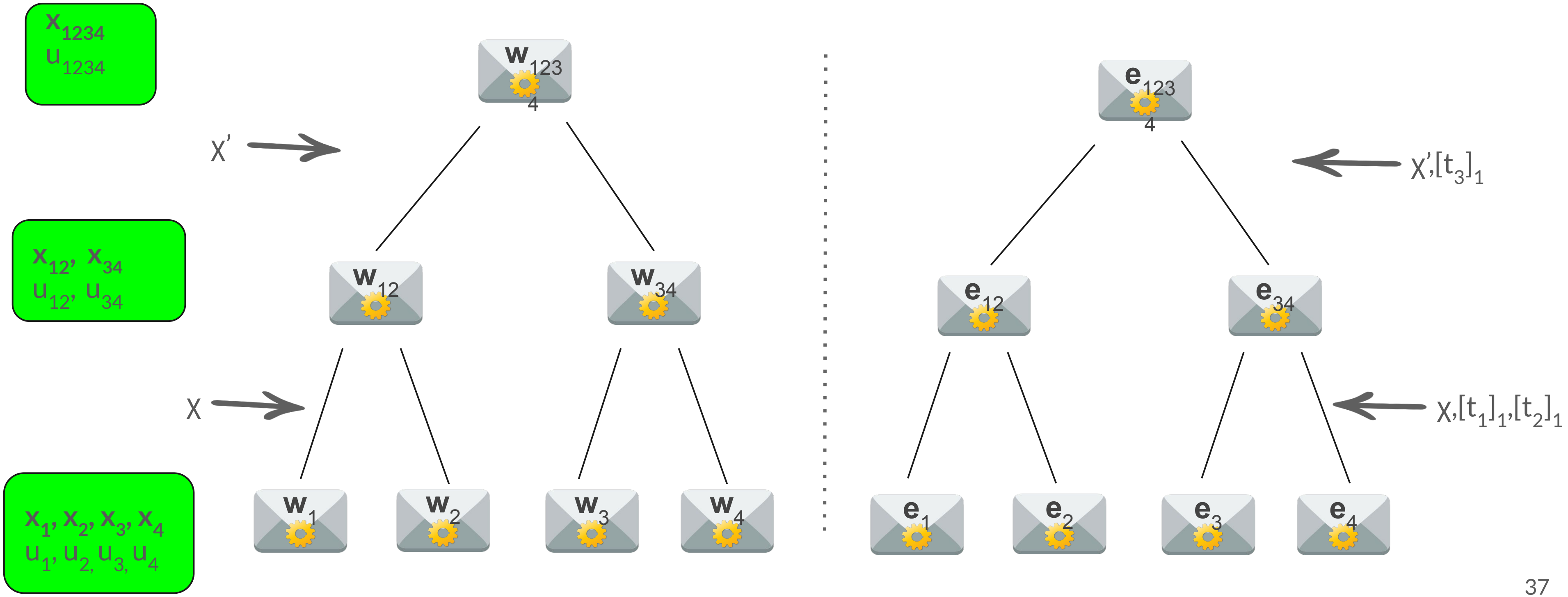
x_1, x_2, x_3, x_4
u_1, u_2, u_3, u_4



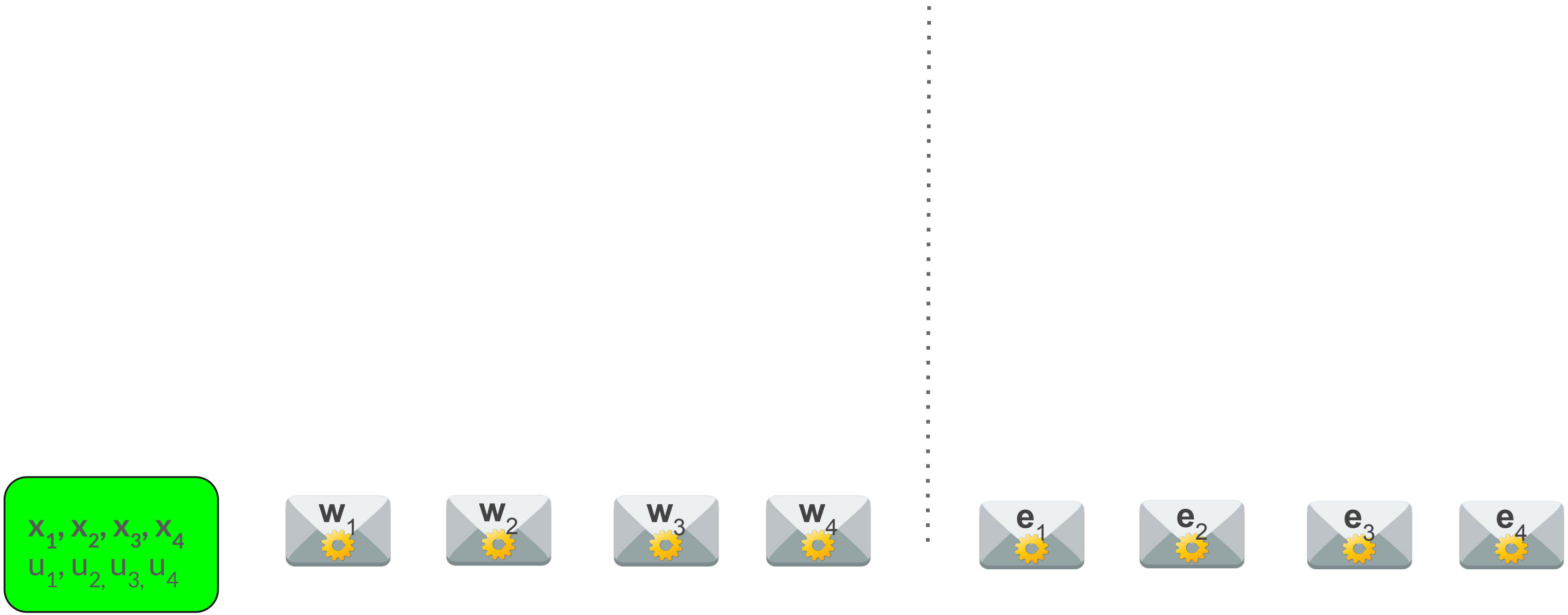
Nova style Folding



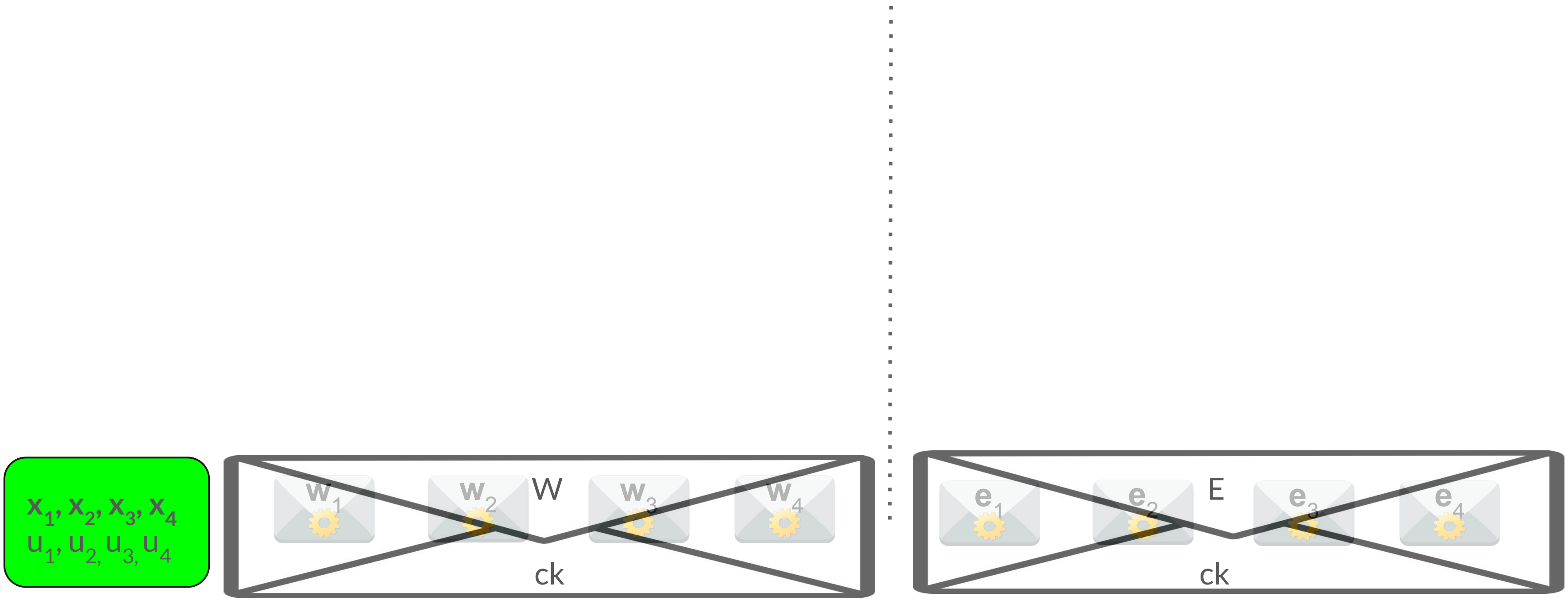
Nova style Folding



Flip: Folding via IPP

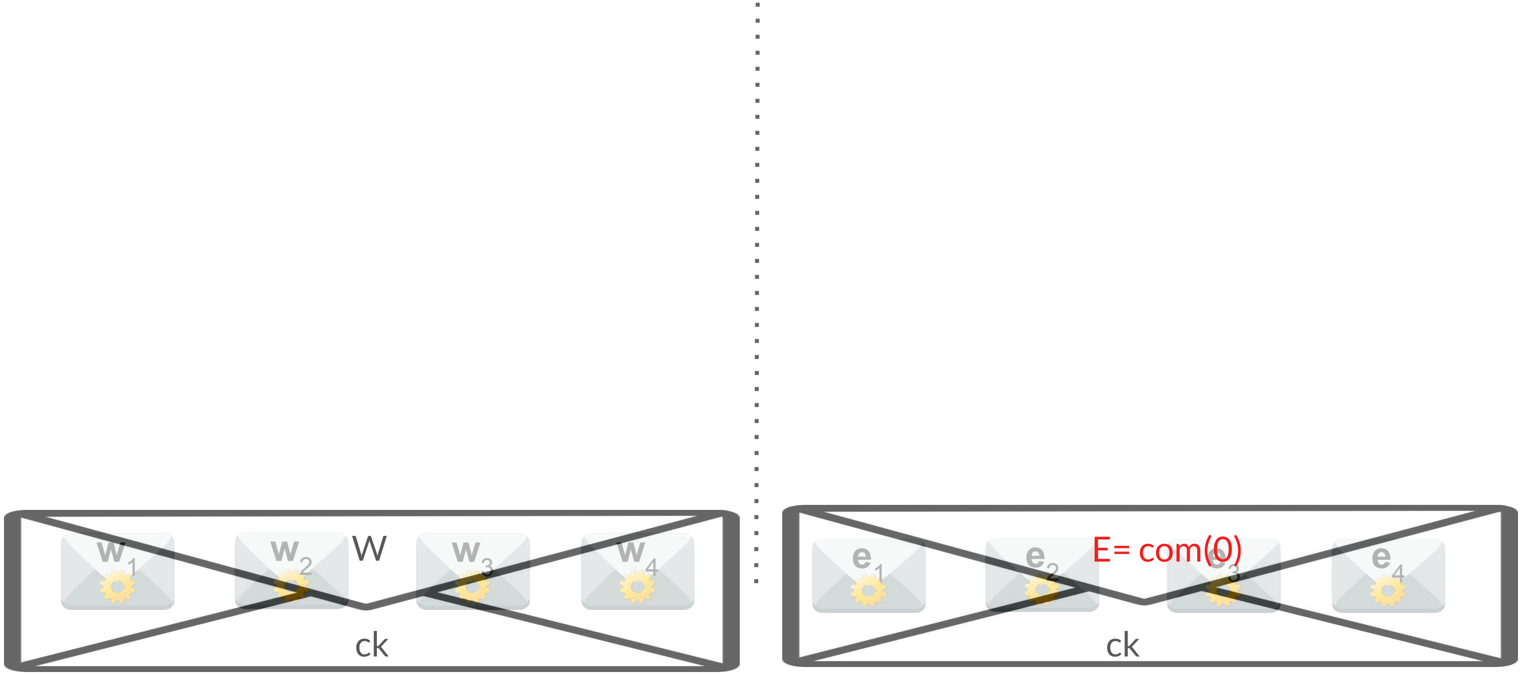


Flip: Folding via IPP

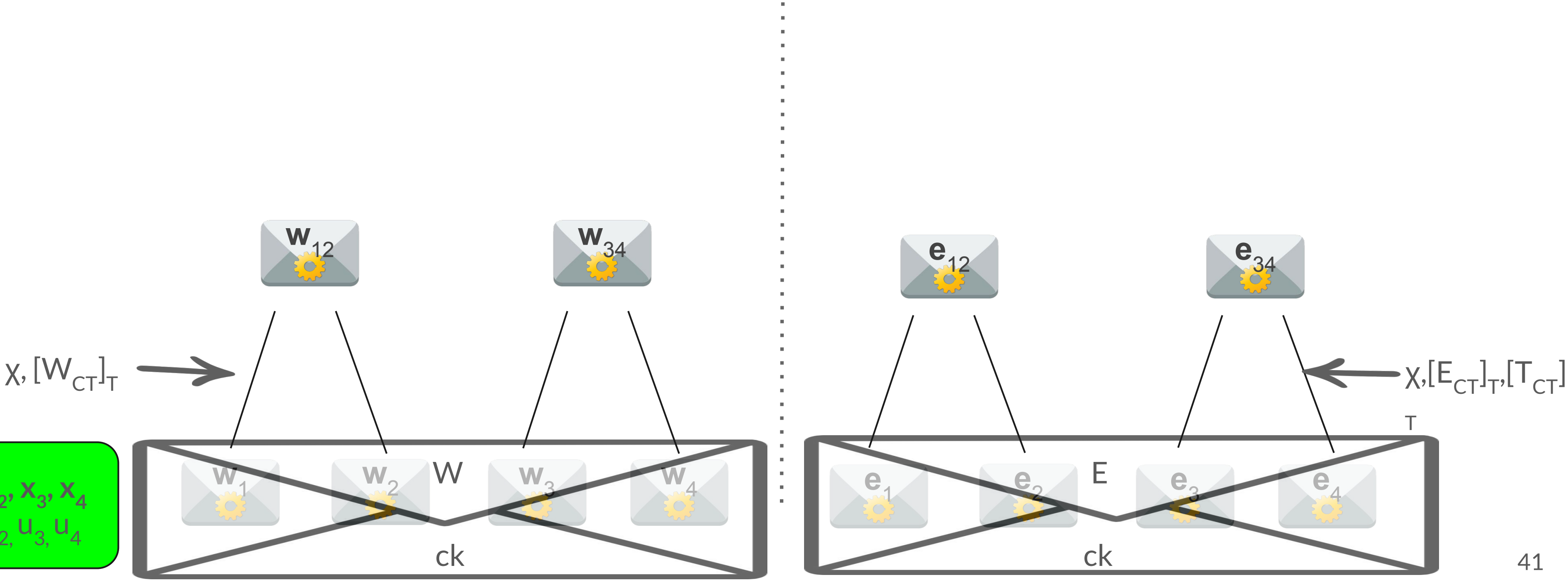


Flip: Folding via IPP

x_1, x_2, x_3, x_4
u_1, u_2, u_3, u_4

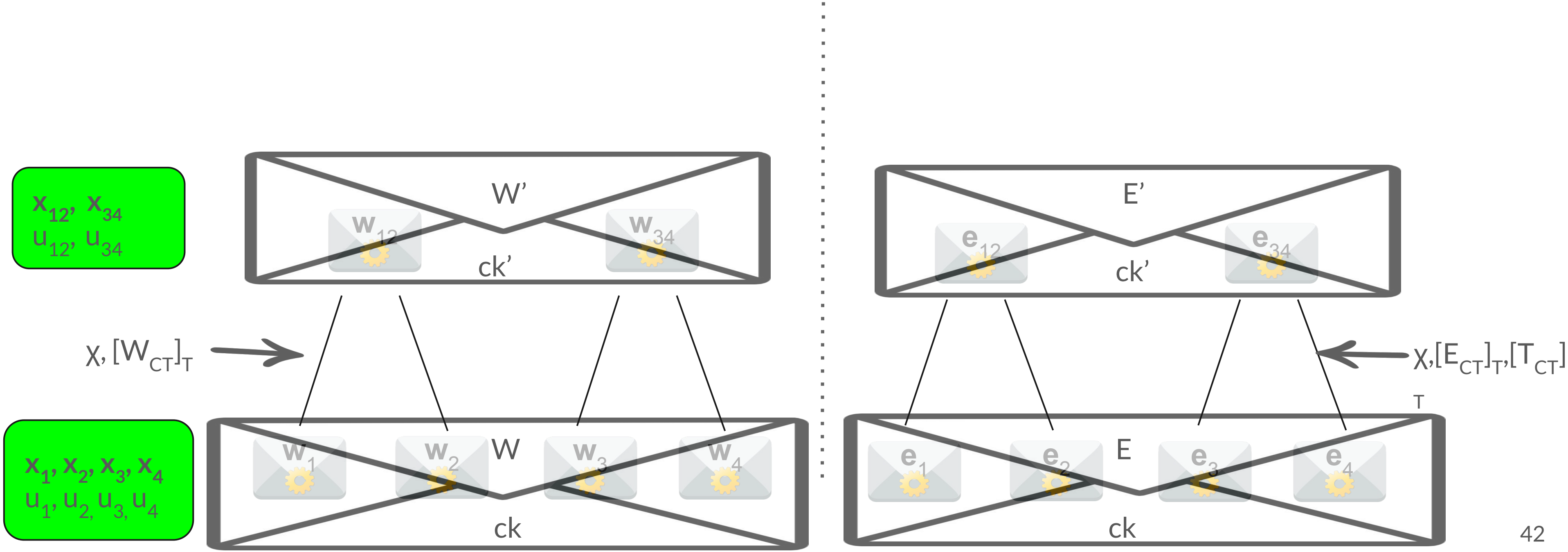


Flip: Folding via IPP

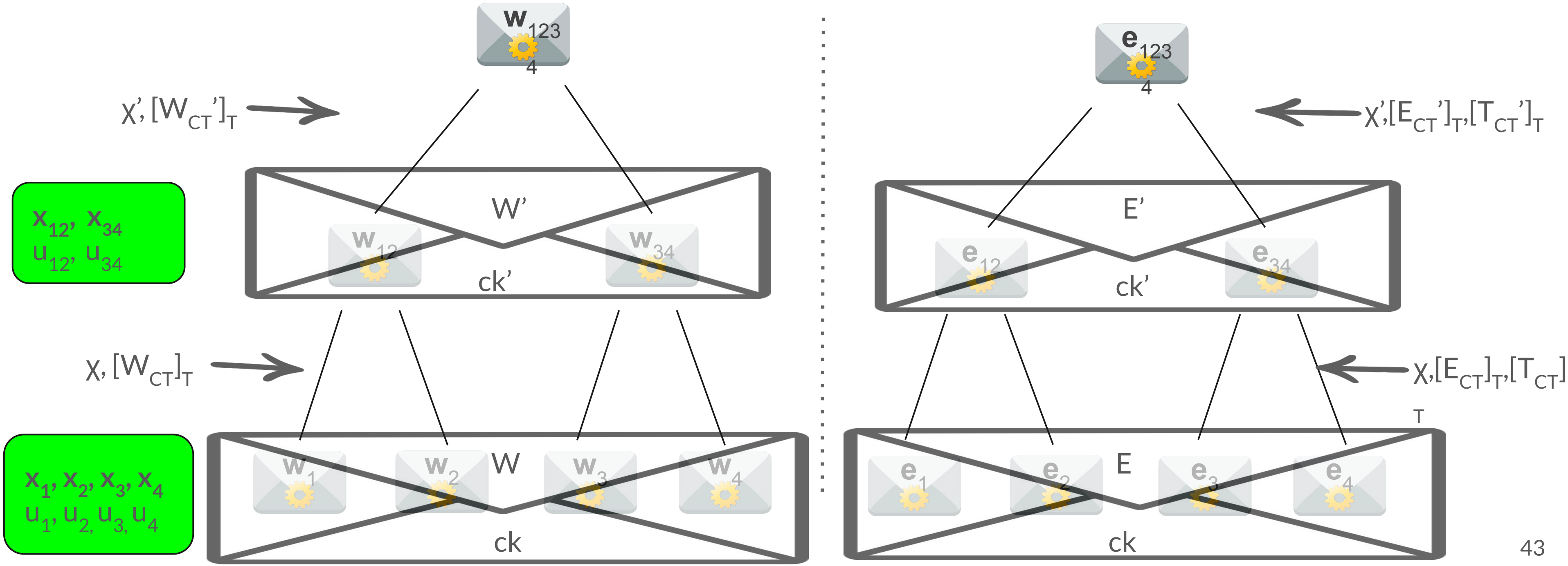


x_1, x_2, x_3, x_4
u_1, u_2, u_3, u_4

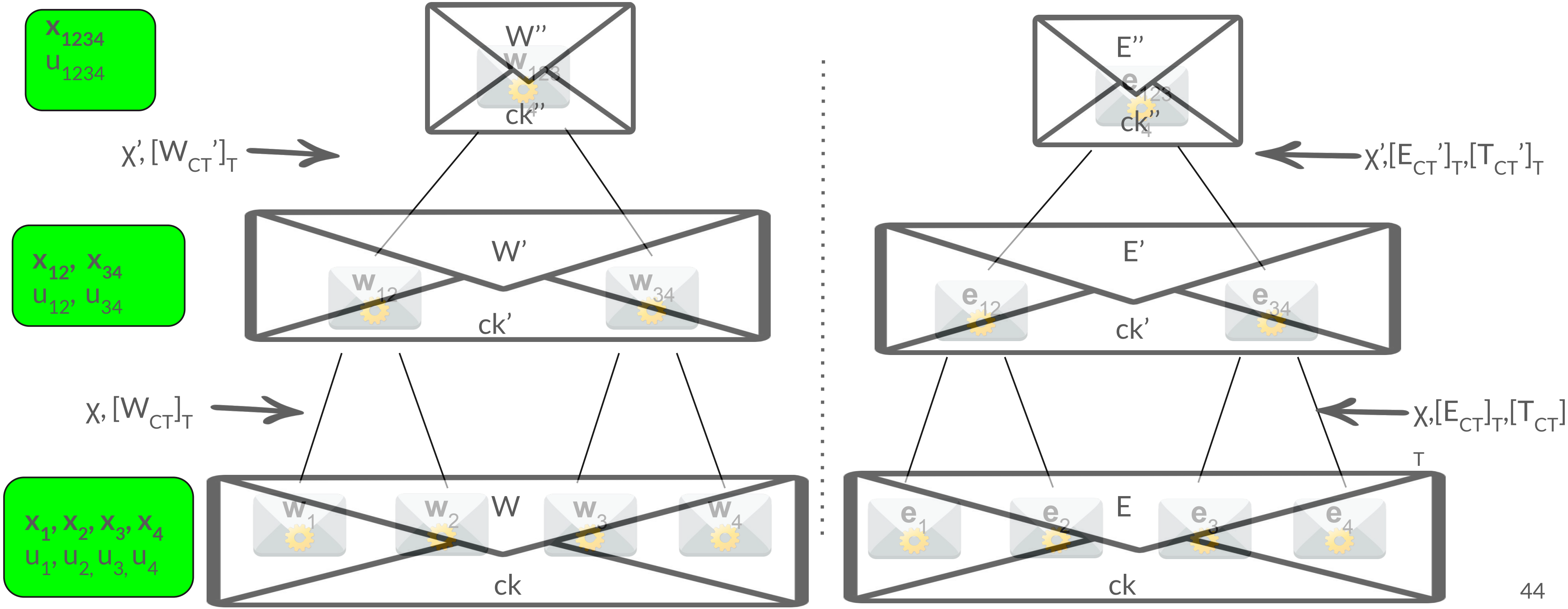
Flip: Folding via IPP



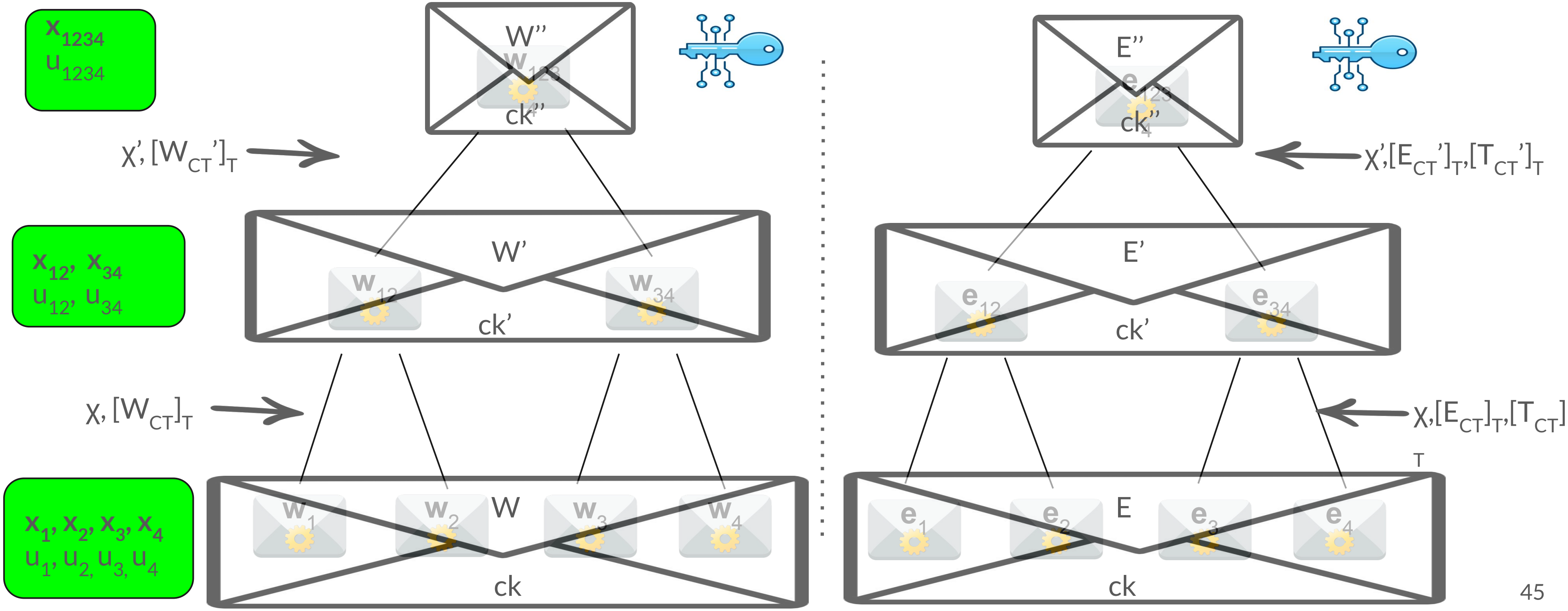
Flip: Folding via IPP



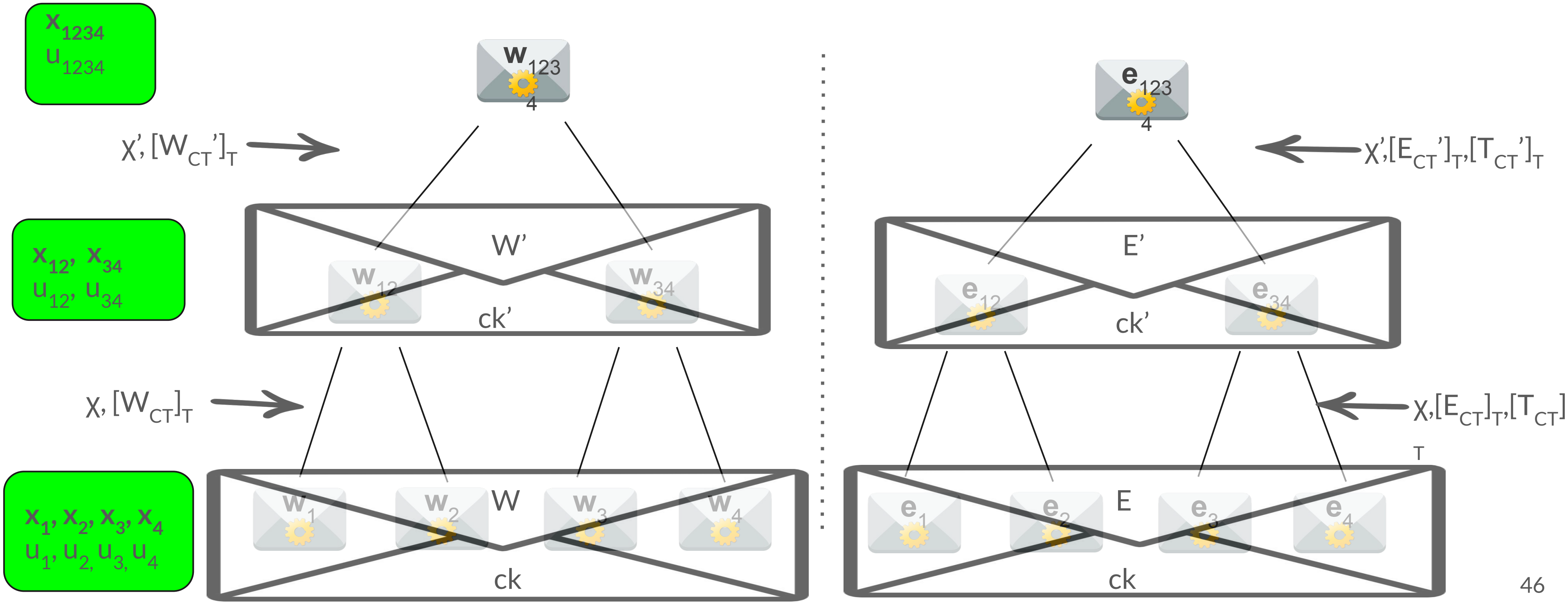
Flip: Folding via IPP



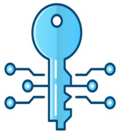
Flip: Folding via IPP

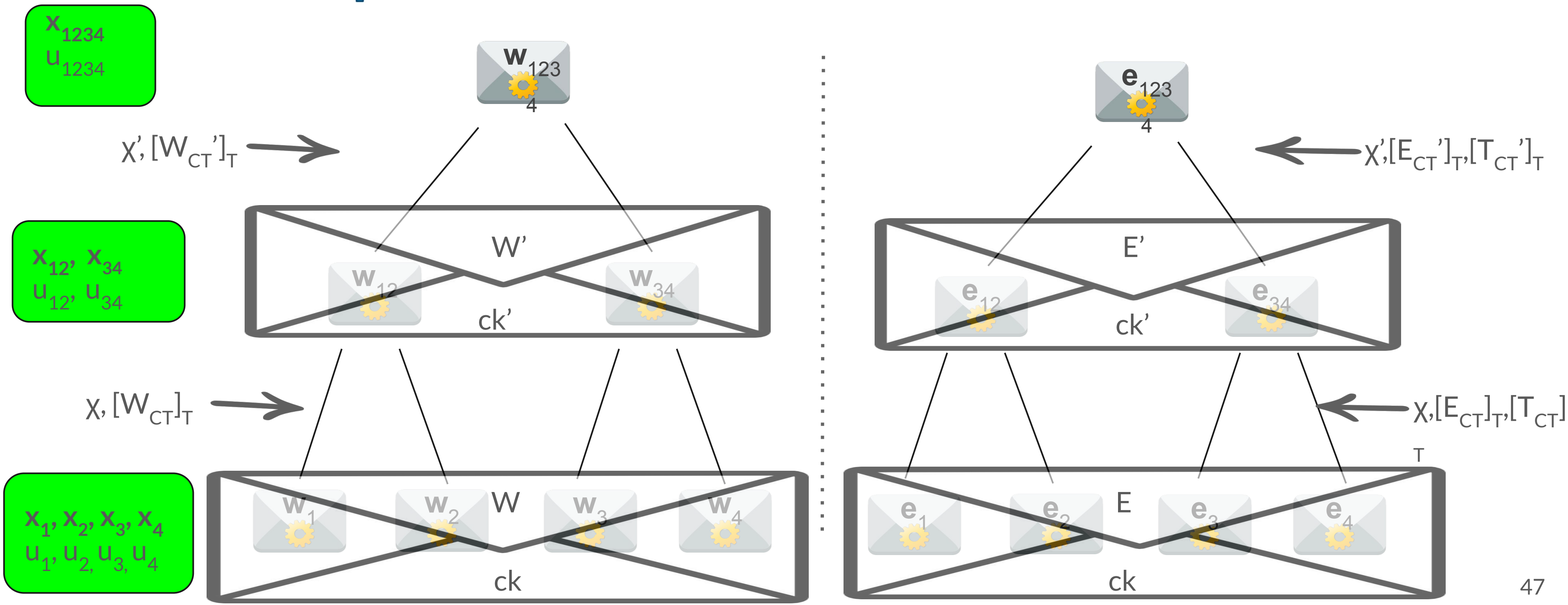


Flip: Folding via IPP



Flip: Folding via IPP

Check  correctness succinctly



Flip: Folding via IPP

Flipp

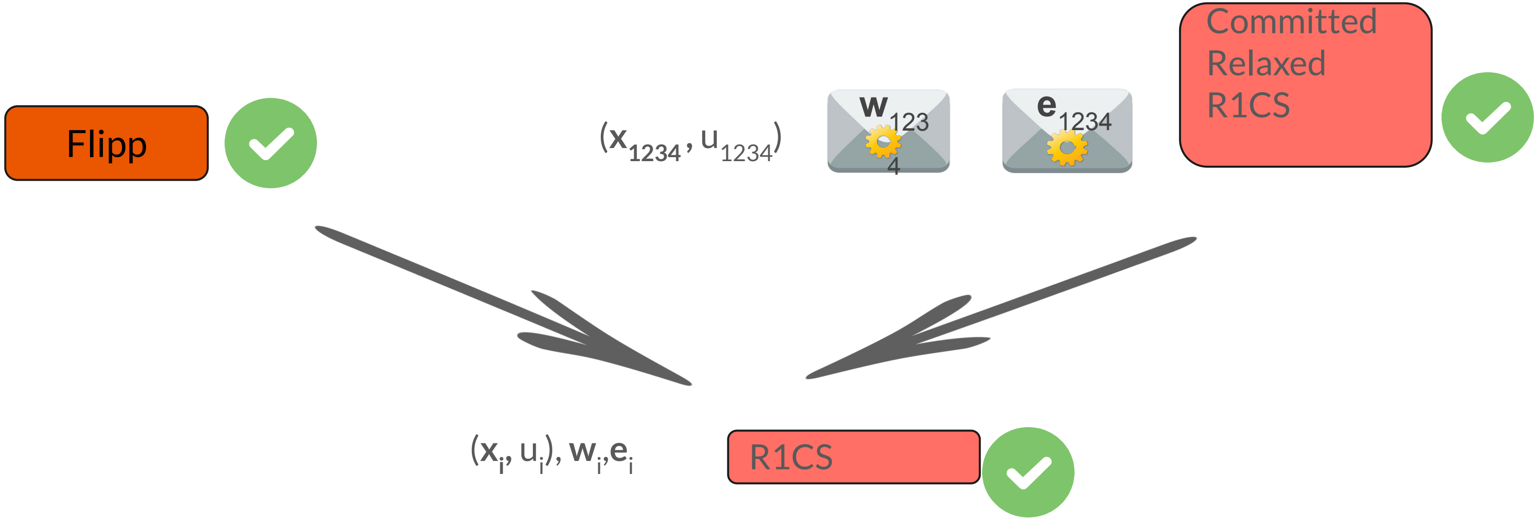
Flip: Folding via IPP

Flipp

(x_{1234}, u_{1234}) w_{1234} e_{1234}

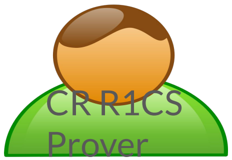
Committed Relaxed R1CS

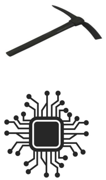
Flip: Folding via IPP



Proving Final

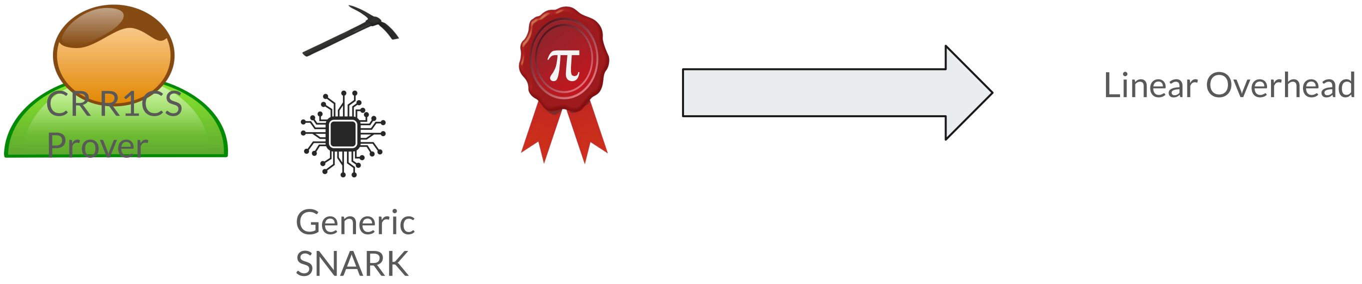
Proving final statement



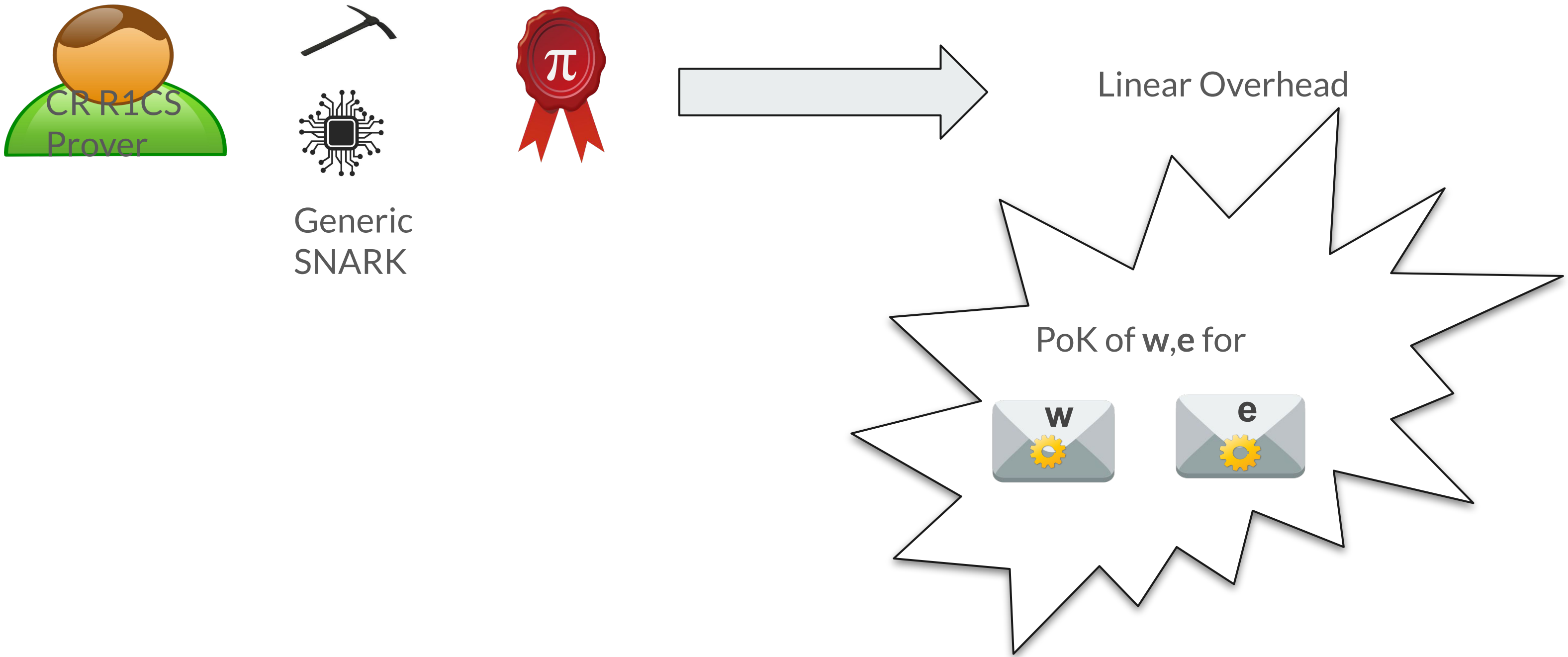


Generic
SNARK

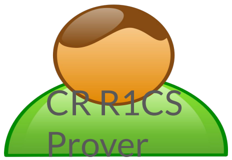
Proving final statement

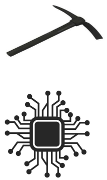


Proving final statement



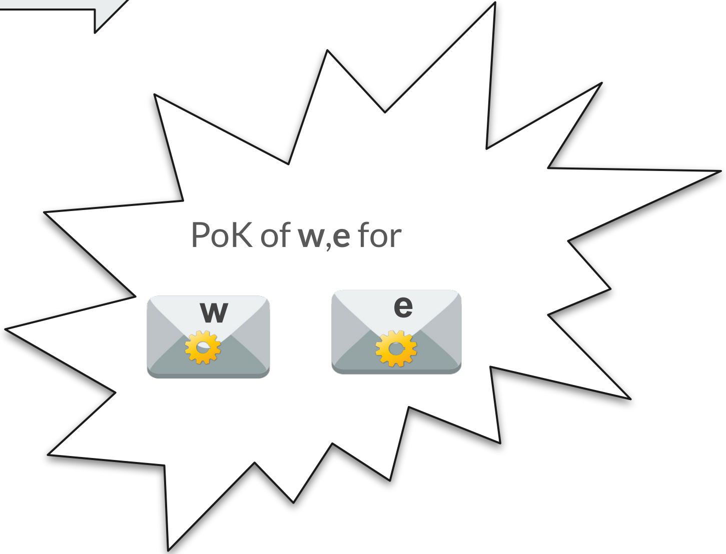
Proving final statement





Generic
SNARK

Linear Overhead



Tailored SNARK for CR R1CS

Committed Relaxed Groth16

Committed Relaxed Groth16

Support
Commit and
Prove

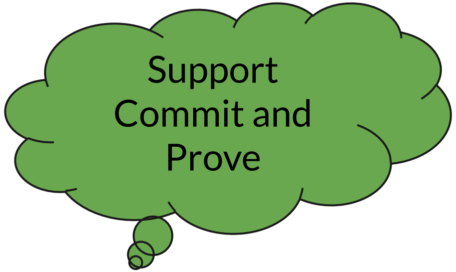
Committed Relaxed Groth16

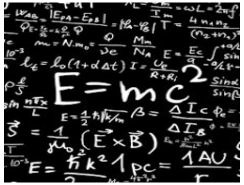
Support Commit and Prove

OR

use generic commit and prove or sparse matrix lincheck techniques

Committed Relaxed Groth16





OR

use generic commit and prove or sparse matrix lincheck techniques

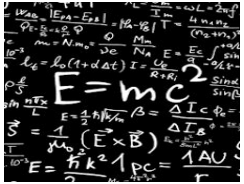
Committed Relaxed Groth16

Support Commit and Prove

OR

use generic commit and prove or sparse matrix lincheck techniques

Support error terms of CR R1CS



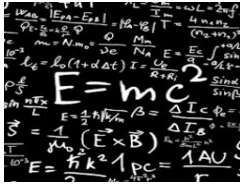
Committed Relaxed Groth16

$$(\mathsf{srs}_{\mathsf{P}}, \mathsf{srs}_{\mathsf{V}}) := \mathsf{srs} \leftarrow \left(\begin{array}{l} \Big[\alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \{u_j(x)\beta + v_j(x)\alpha + w_j(x) + \gamma\ell_j(x)\}_{j=0}^{l}, \\ \left\{ \dfrac{u_j(x)\beta + v_j(x)\alpha + w_j(x) + \gamma\ell_j(x)}{\delta} \right\}_{j=l+1}^{m}, \{x^i t(x)/\delta\}_{i=0}^{n-2} \Big]_1, \\ [\beta, \delta, \gamma, \{x^i\}_{i=0}^{n-1}]_2,\ [\alpha\beta, t(x)]_T, H \end{array} \right)$$

Committed Relaxed Groth16

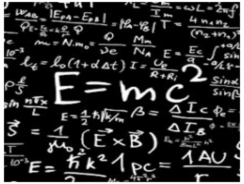
$$(\mathsf{srs}_\mathsf{P}, \mathsf{srs}_\mathsf{V}) := \mathsf{srs} \leftarrow \begin{pmatrix} \left[\alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \{u_j(x)\beta + v_j(x)\alpha + w_j(x) + \underline{\gamma\ell_j(x)}\}_{j=0}^{l}, \right. \\ \left. \left\{ \frac{u_j(x)\beta + v_j(x)\alpha + w_j(x) + \underline{\gamma\ell_j(x)}}{\delta} \right\}_{j=l+1}^{m}, \{x^i t(x)/\delta\}_{i=0}^{n-2} \right]_1, \\ [\beta, \delta, \underline{\gamma}, \{x^i\}_{i=0}^{n-1}]_2, \ [\alpha\beta, t(x)]_T, H \end{pmatrix}$$

Committed Relaxed Groth16



$$[A]_1\,[B]_2 - [C]_1\,[\delta]_2 - \left(\sum_{j=0}^{l} z_j\,[u_j(x)\beta + v_j(x)\alpha + w_j(x) + \gamma\ell_j(x)]_1 - [e]_1\,u^{-1}\right)[1]_2 + [w]_1\,[\gamma]_2 = u\,[\alpha\beta]_T$$

Committed Relaxed Groth16



$$[A]_1\,[B]_2 - [C]_1\,[\delta]_2 - \left(\sum_{j=0}^{l} z_j\left[u_j(x)\beta + v_j(x)\alpha + w_j(x) + \underline{\gamma\ell_j(x)]_1}\right. - \underline{[e]_1\,u^{-1}}\right)[1]_2 + \underline{[w]_1\,[\gamma]_2} = u\,[\alpha\beta]_T$$

Summary

We examine the problem of proof aggregation when a single prover is producing multiply proofs. Our contributions are the following:

Summary

We examine the problem of proof aggregation when a single prover is producing multiply proofs. Our contributions are the following:

- **Flipp**: a protocol that uses inner pairing product techniques to fold with logarithmic communication committed relaxed R1CS instances.

Summary

We examine the problem of proof aggregation when a single prover is producing multiply proofs. Our contributions are the following:

- **Flipp**: a protocol that uses inner pairing product techniques to fold with logarithmic communication committed relaxed R1CS instances.

- **Committed Relaxed Groth16**: a modification of Groth16 for proving committed relaxed R1CS instances.

Conclusion

Flip and prove aggregation:

- massive reduction in prover time compared to SnarkPack, in the setting of a single prover
- no need for expensive arithmetization of verifier circuit inside the prover
- no need for (half) cycles of elliptic curves

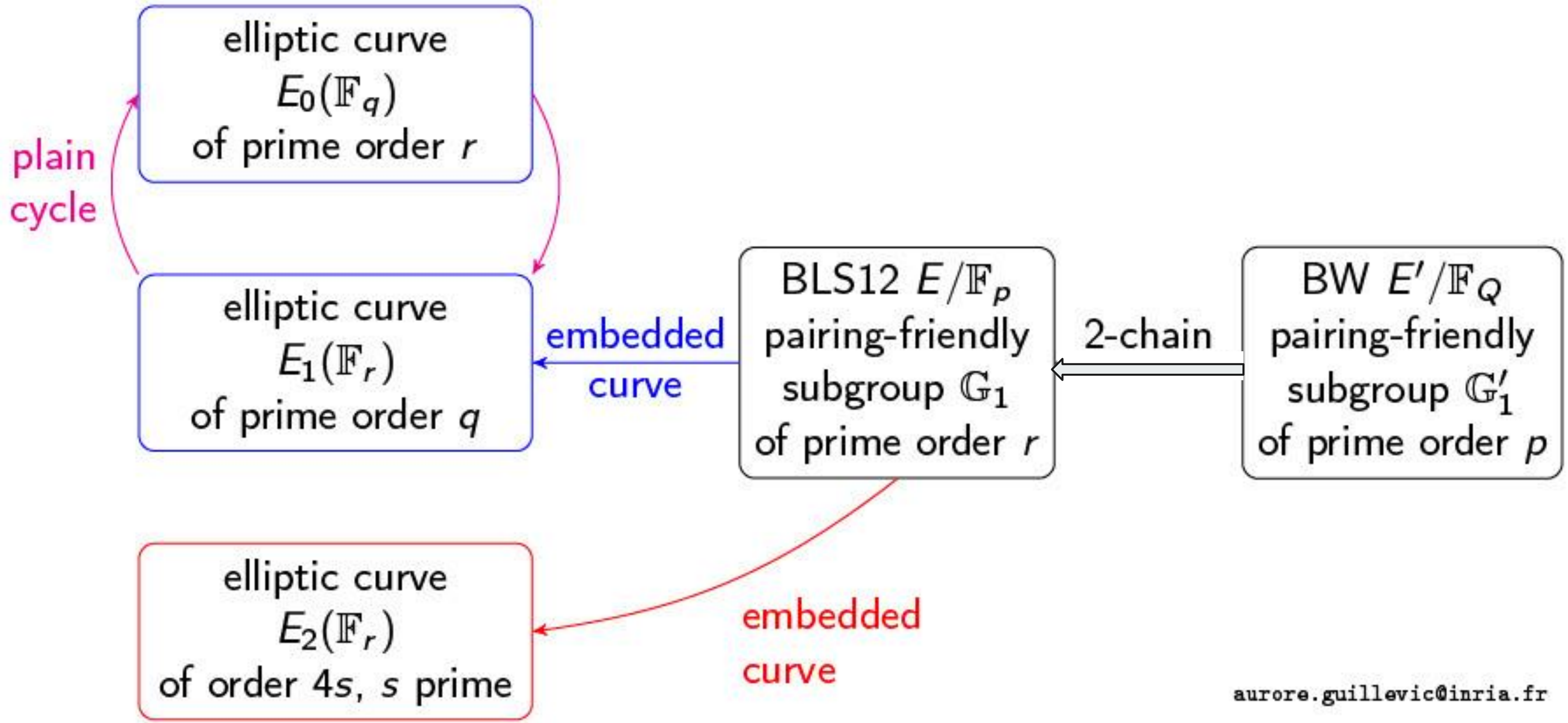
Conclusion

Flip and prove aggregation:

- massive reduction in prover time compared to SnarkPack
- no need for expensive arithmetization of verifier circuit inside the prover
- no need for (half) cycles of elliptic curves

Committed Relaxed Groth16:

- Can be potentially used as an alternative for proving the last step of Nova



aurore.guillevic@inria.fr


THANK
YOU