Learning zk-snark process by tools

Subtitle: Programming zkp with circom

ZK-School: Beginner
(2/16, 2023)
정동현

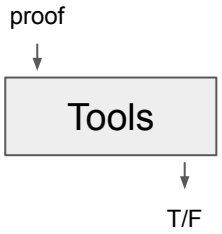
jdhyun1209@gmail.com

Today session is..

(c) Verifier V

- INPUTS: verification key vk, input $\vec{x} \in \mathbb{F}_r^n$, and proof π
- OUTPUTS: decision bit

1. Compute $\mathsf{vk}_{\vec{x}} := \mathsf{vk}_{\mathsf{IC},0} + \sum_{i=1}^{n} x_i \mathsf{vk}_{\mathsf{IC},i} \in \mathbb{G}_1$.
2. Check validity of knowledge commitments for A, B, C:
$$e(\pi_{\mathsf{A}}, \mathsf{vk}_{\mathsf{A}}) = e(\pi'_{\mathsf{A}}, \mathcal{P}_2)\,, e(\mathsf{vk}_{\mathsf{B}}, \pi_{\mathsf{B}}) = e(\pi'_{\mathsf{B}}, \mathcal{P}_2)\,,$$
$$e(\pi_{\mathsf{C}}, \mathsf{vk}_{\mathsf{C}}) = e(\pi'_{\mathsf{C}}, \mathcal{P}_2)\,.$$
3. Check same coefficients were used:
$$e(\pi_{\mathsf{K}}, \mathsf{vk}_{\gamma}) = e(\mathsf{vk}_{\vec{x}} + \pi_{\mathsf{A}} + \pi_{\mathsf{C}}, \mathsf{vk}^2_{\beta\gamma}) \cdot e(\mathsf{vk}^1_{\beta\gamma}, \pi_{\mathsf{B}})\ .$$
4. Check QAP divisibility:
$$e(\mathsf{vk}_{\vec{x}} + \pi_{\mathsf{A}}, \pi_{\mathsf{B}}) = e(\pi_{\mathsf{H}}, \mathsf{vk}_{\mathsf{Z}}) \cdot e(\pi_{\mathsf{C}}, \mathcal{P}_2)\ .$$
5. Accept if and only if all the above checks succeeded.



But, we have to face the math to go further..

Session Goals

- Circom과 snarkjs로 ZK-snark의 전체적인 프로세스 소개
  - Circuit 작성(Circom)
  - 증명생성 및 검증(snarkjs)

So,

- 툴링이 되어있는 시스템을 통해 (zk-snark)proof systems을 몰라도 큰그림을 살펴볼 수 있도록
- 추후 이론을 공부할때 이정표가 되도록(결국은 수학 필요)

Outline

1. What is zk-snark?
2. Circom
   a. R1CS https://www.youtube.com/watch?v=puxRj0NeqIo
   b. building circuit
3. snarkjs

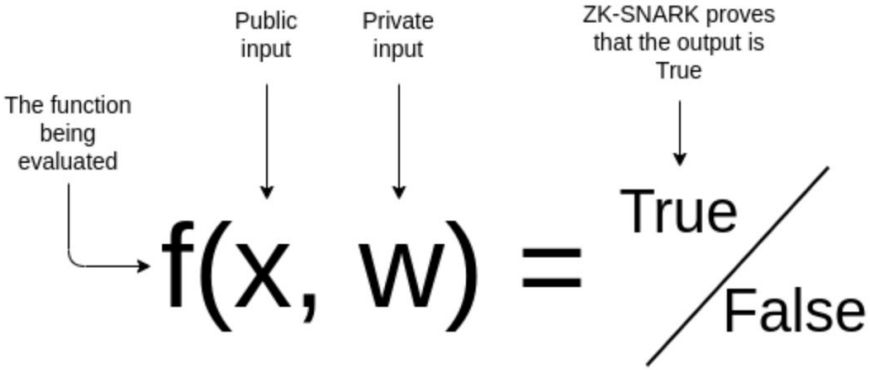
What is zk-snark?

ZK-SNARK : Zero Knowledge Succint Non-interactive ARgument of Knowledge

S(succinct):
증명되는 개념이 복잡해도 proof가 작아서 검증이 빠르게
N(non-interactive):
prover(증명자)와 verifier(검증자) 사이에 interaction이 필요X
ARK(Argument of Knowledge):
“지식증명”, (prover가 실제로 뭔가 알고 있다는 것을 증명하고자 함)

What is zk-snark?

Computational Problem: F(x,w) ex). Hash(x,w), x^3+5x+3, anything..



What is zk-snark?

Computational Problem -> R1CS 형식으로

기본적으로 유한체안에서 (+,*)연산으로만 이루어져 있는 Arithmetic circuit으로 바꿈
(R1CS는 Arithmetic circuit 표현 방법 중에 하나.)

Example

$f(x) = (x1 + x2) * x3 + 5$ hiding input: x1, x2, x3

Example

$f(x) = (x1 + x2) * x3 + 5$ hiding input: x1, x2, x3

$y1 = x1 + x2$

$y2 = y1 * x3$

$out = y2 + 5$

hiding input: x1, x2, x3

intermediate value: y1, y2

public output: out

Example

$f(x) = (x1 + x2) * x3 + 5$ hiding input: x1, x2, x3

$y1 = x1 + x2$

$y2 = y1 * x3$

$out = y2 + 5$

[?, out, x1, x2, x3, y1, y2]

Example

y1 = x1 + x2

y2 = y1 * x3

out = y2 + 5

s = [?, out, x1

(a . s) * (b . s

Dot product

$$\langle r, s \rangle = \begin{bmatrix} 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}$$

$$= 2 \cdot 1 + 0 \cdot (-1) + 1 \cdot 2$$

$$= 4$$

Example

$y1 = x1 + x2 \rightarrow \langle a1 . s\rangle * \langle b1 . s\rangle - \langle c1 . s\rangle = 0$

1 이 나와야됨

$y2 = y1 * x3 \rightarrow \langle a2 . s\rangle * \langle b2 . s\rangle - \langle c2 . s\rangle = 0$

$out = y2 + 5 \rightarrow \langle a3 . s\rangle * \langle b3 . s\rangle - \langle c3 . s\rangle = 0$

$s = [?, out, x1, x2, x3, y1, y2]$

$(a . s) * (b . s) - (c . s) = 0$

Example

y1 = x1 + x2 → <a1 . s> * <b1 . s> - <c1 . s> = 0

y1 = x1 + x2 → 1*(x1 + x2) - y1 = 0

[1, 0, 0, 0, 0, 0, 0] . s = 1

y2 = y1 * x3 → <a2 . s> * <b2 . s> - <c2 . s> = 0

out = y2 + 5 → <a3 . s> * <b3 . s> - <c3 . s> = 0

s = [?, out, x1, x2, x3, y1, y2]

Example

$y1 = x1 + x2 \rightarrow \langle a1 . s\rangle * \langle b1 . s\rangle - \langle c1 . s\rangle = 0$

$y1 = x1 + x2 \rightarrow 1*(x1 + x2) - y1 = 0$

$[1, 0, 0, 0, 0, 0, 0] . s = 1$

$y2 = y1 * x3 \rightarrow \langle a2 . s\rangle * \langle b2 . s\rangle - \langle c2 . s\rangle = 0$

$out = y2 + 5 \rightarrow \langle a3 . s\rangle * \langle b3 . s\rangle - \langle c3 . s\rangle = 0$

$s = [1, out, x1, x2, x3, y1, y2]$

Example

$y1 = x1 + x2 \quad \rightarrow \quad \langle a1 \,.\, s\rangle * \langle b1 \,.\, s\rangle - \langle c1 \,.\, s\rangle = 0$

$y1 = x1 + x2 \quad \rightarrow \quad 1*(x1 + x2) - y1 = 0$

$a1 = [1, 0, 0, 0, 0, 0, 0]$

$[0, 0, 1, 1, 0, 0, 0] \,.\, s = x1 + x2$

$y2 = y1 * x3 \quad \rightarrow \quad \langle a2 \,.\, s\rangle * \langle b2 \,.\, s\rangle - \langle c2 \,.\, s\rangle = 0$

$out = y2 + 5 \quad \rightarrow \quad \langle a3 \,.\, s\rangle * \langle b3 \,.\, s\rangle - \langle c3 \,.\, s\rangle = 0$

$s = [1, out, x1, x2, x3, y1, y2]$

Example

y1 = x1 + x2 → <a1 . s> * <b1 . s> - <c1 . s> = 0

y1 = x1 + x2 → 1*(x1 + x2) - y1 = 0

a1 = [1, 0, 0, 0, 0, 0, 0]

b1 = [0, 0, 1, 1, 0, 0, 0]

[0, 0, 0, 0, 0, 1, 0] . s = y1

y2 = y1 * x3 → <a2 . s> * <b2 . s> - <c2 . s> = 0

out = y2 + 5 → <a3 . s> * <b3 . s> - <c3 . s> = 0

s = [1, out, x1, x2, x3, y1, y2]

Example

y1 = x1 + x2 → <a1 . s> * <b1 . s> - <c1 . s> = 0

y1 = x1 + x2 → 1*(x1 + x2) - y1 = 0

a1 = [1, 0, 0, 0, 0, 0, 0]

b1 = [0, 0, 1, 1, 0, 0, 0]

c1 = [0, 0, 0, 0, 0, 1, 0]

y2 = y1 * x3 → <a2 . s> * <b2 . s> - <c2 . s> = 0

out = y2 + 5 → <a3 . s> * <b3 . s> - <c3 . s> = 0

s = [1, out, x1, x2, x3, y1, y2]

Example

y1 = x1 + x2 → <a1 . s> * <b1 . s> - <c1 . s> = 0

a1 = [1, 0, 0, 0, 0, 0, 0]

b1 = [0, 0, 1, 1, 0, 0, 0]

c1 = [0, 0, 0, 0, 0, 1, 0]

y2 = y1 * x3 → <a2 . s> * <b2 . s> - <c2 . s> = 0

out = y2 + 5 → <a3 . s> * <b3 . s> - <c3 . s> = 0

s = [1, out, x1, x2, x3, y1, y2]

Example

y1 = x1 + x2 → <a1 . s> * <b1 . s> - <c1 . s> = 0

a1 = [1, 0, 0, 0, 0, 0, 0]

b1 = [0, 0, 1, 1, 0, 0, 0]

c1 = [0, 0, 0, 0, 0, 1, 0]

y2 = y1 * x3 → <a2 . s> * <b2 . s> - <c2 . s> = 0

y2 = y1 * x3 → y1 * x3 - y2 = 0

a2 = [0, 0, 0, 0, 0, 1, 0]

b2 = [0, 0, 0, 0, 1, 0, 0]

c2 = [0, 0, 0, 0, 0, 0, 1]

out = y2 + 5 → <a3 . s> * <b3 . s> - <c3 . s> = 0

s = [1, out, x1, x2, x3, y1, y2]

Example

y1 = x1 + x2 → <a1 . s> * <b1 . s> - <c1 . s> = 0

a1 = [1, 0, 0, 0, 0, 0, 0]
b1 = [0, 0, 1, 1, 0, 0, 0]
c1 = [0, 0, 0, 0, 0, 1, 0]

y2 = y1 * x3 → <a2 . s> * <b2 . s> - <c2 . s> = 0

a2 = [0, 0, 0, 0, 0, 1, 0]
b2 = [0, 0, 0, 0, 1, 0, 0]
c2 = [0, 0, 0, 0, 0, 0, 1]

out = y2 + 5 → <a3 . s> * <b3 . s> - <c3 . s> = 0

a3 = [1, 0, 0, 0, 0, 0 ,0]
b3 = [5, 0, 0, 0, 0, 0, 1]
c3 = [0, 1, 0, 0, 0, 0, 0]

s = [1, out, x1, x2, x3, y1, y2]

Example

y1 = x1 + x2 → <a1 . s> * <b1 . s> - <c1 . s> = 0

a1 = [1, 0, 0, 0, 0, 0, 0]
b1 = [0, 0, 1, 1, 0, 0, 0]
c1 = [0, 0, 0, 0, 0, 1, 0]

y2 = y1 * x3 → <a2 . s> * <b2 . s> - <c2 . s> = 0

a2 = [0, 0, 0, 0, 0, 1, 0]
b2 = [0, 0, 0, 0, 1, 0, 0]
c2 = [0, 0, 0, 0, 0, 0, 1]

out = y2 + 5 → <a3 . s> * <b3 . s> - <c3 . s> = 0

a3 = [1, 0, 0, 0, 0, 0 ,0]
b3 = [5, 0, 0, 0, 0, 0, 1]
c3 = [0, 1, 0, 0, 0, 0, 0]

$$a_i \cdot \vec{s} + b_i \cdot \vec{s} - c_i \cdot \vec{s} = 0$$

s = [1, out, x1, x2, x3, y1, y2]

Example

$$\left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ out \\ x1 \\ x2 \\ x3 \\ y1 \\ y2 \end{bmatrix}\right) \circ \left(\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ out \\ x1 \\ x2 \\ x3 \\ y1 \\ y2 \end{bmatrix}\right) - \left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ out \\ x1 \\ x2 \\ x3 \\ y1 \\ y2 \end{bmatrix}\right) = 0$$

R1CS(Rank-1 Constraint System) Form

Example

Constraints Matrices

$$\left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ out \\ x1 \\ x2 \\ x3 \\ y1 \\ y2 \end{bmatrix}\right) \circ \left(\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ out \\ x1 \\ x2 \\ x3 \\ y1 \\ y2 \end{bmatrix}\right) - \left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ out \\ x1 \\ x2 \\ x3 \\ y1 \\ y2 \end{bmatrix}\right) = 0$$

R1CS(Rank-1 Constraint System) Form

Example

Solution vector == witness vector

$$\left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ out \\ x1 \\ x2 \\ x3 \\ y1 \\ y2 \end{bmatrix}\right) \circ \left(\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ out \\ x1 \\ x2 \\ x3 \\ y1 \\ y2 \end{bmatrix}\right) - \left(\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 \\ out \\ x1 \\ x2 \\ x3 \\ y1 \\ y2 \end{bmatrix}\right) = 0$$

R1CS(Rank-1 Constraint System) Form

Example

Solution vector == witness vector

with public inputs = x

private inputs = w

$$\left(\begin{bmatrix}1&0&0&0&0&0&0\\0&0&0&0&0&1&0\\1&0&0&0&0&0&0\end{bmatrix} * \begin{bmatrix}1\\out\\x1\\x2\\x3\\y1\\y2\end{bmatrix}\right) \circ \left(\begin{bmatrix}0&0&1&1&0&0&0\\0&0&0&0&1&0&0\end{bmatrix} * \begin{bmatrix}1\\out\\x1\\x2\\x3\\y1\\y2\end{bmatrix}\right) - \left(\begin{bmatrix}0&0&0&0&0&1&0\\0&0&0&0&0&0&1\\0&1&0&0&0&0&0\end{bmatrix} * \begin{bmatrix}1\\out\\x1\\x2\\x3\\y1\\y2\end{bmatrix}\right) = 0$$

R1CS(Rank-1 Constraint System) Form

Example

Solution vector == witness vector

with public inputs = x

private inputs = w

$$\left(\begin{bmatrix}1&0&0&0&0&0&0\\0&0&0&0&0&1&0\\1&0&0&0&0&0&0\end{bmatrix} * \begin{bmatrix}1\\out\\x1\\x2\\x3\\y1\\y2\end{bmatrix}\right) \circ \left(\begin{bmatrix}0&0&1&1&0&0&0\\0&0&0&0&1&0&0\\ & & & & & & 1\end{bmatrix} * \begin{bmatrix}1\\out\\x1\\x2\\x3\\y1\\y2\end{bmatrix}\right) - \left(\begin{bmatrix}0&0&0&0&0&1&0\\0&0&0&0&0&0&1\\0&1&0&0&0&0&0\end{bmatrix} * \begin{bmatrix}1\\out\\x1\\x2\\x3\\y1\\y2\end{bmatrix}\right) = 0$$

Prover가 해야할 것 : 위를 만족하는 witness vector를 알고 있음을 증명

Circom

zkrepl - Online play ground for zk circom circuits.
https://zkrepl.dev/

$f(x) = (x1 + x2) * x3 + 5$ hiding input: x1, x2, x3

$y1 = x1 + x2$

$y2 = y1 * x3$

$out = y2 + 5$

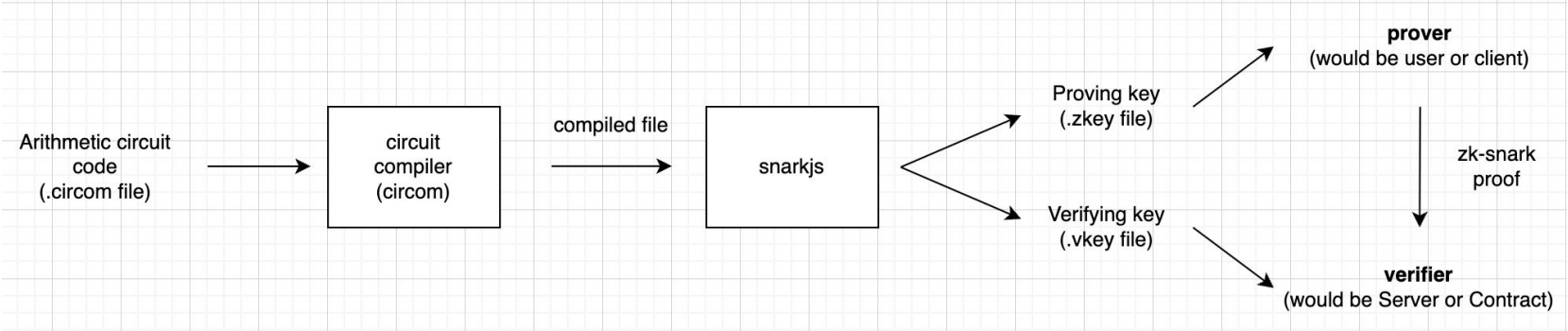
Circom

Circom docs:
https://docs.circom.io/

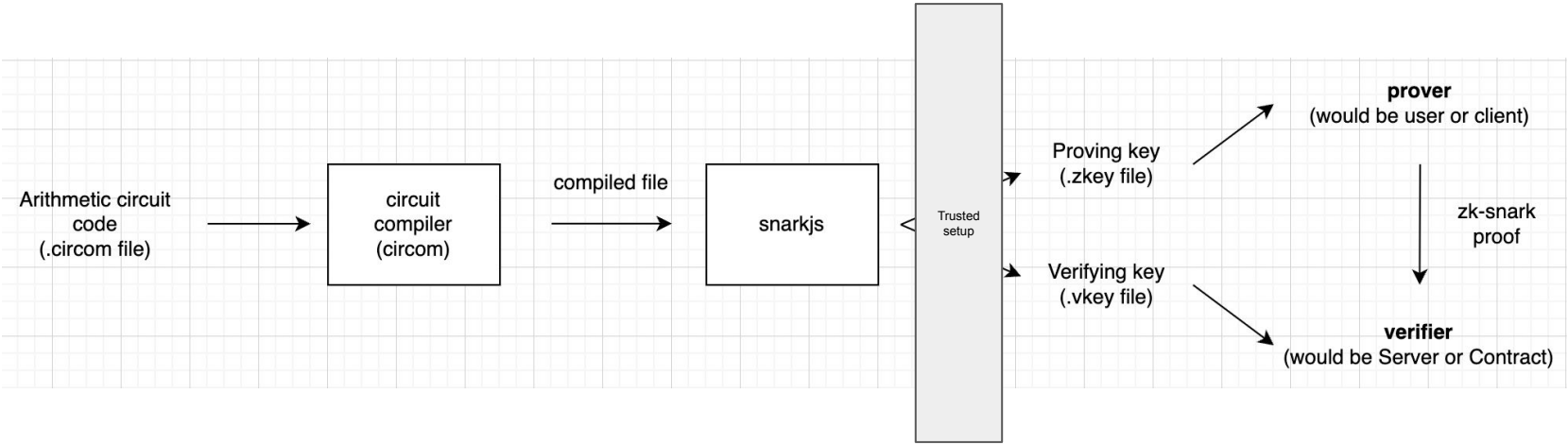
Other useful tools
zkrepl - Online play ground for zk circom circuits.
https://zkrepl.dev/
hardhat-circom - Hardhat plugin to integrate circom and snarkjs into build process.
https://github.com/projectsophon/hardhat-circom#readme
circomlib - Repository contains a library of circuit templates.
https://github.com/iden3/circomlib

Circuit clear!
Next step is..

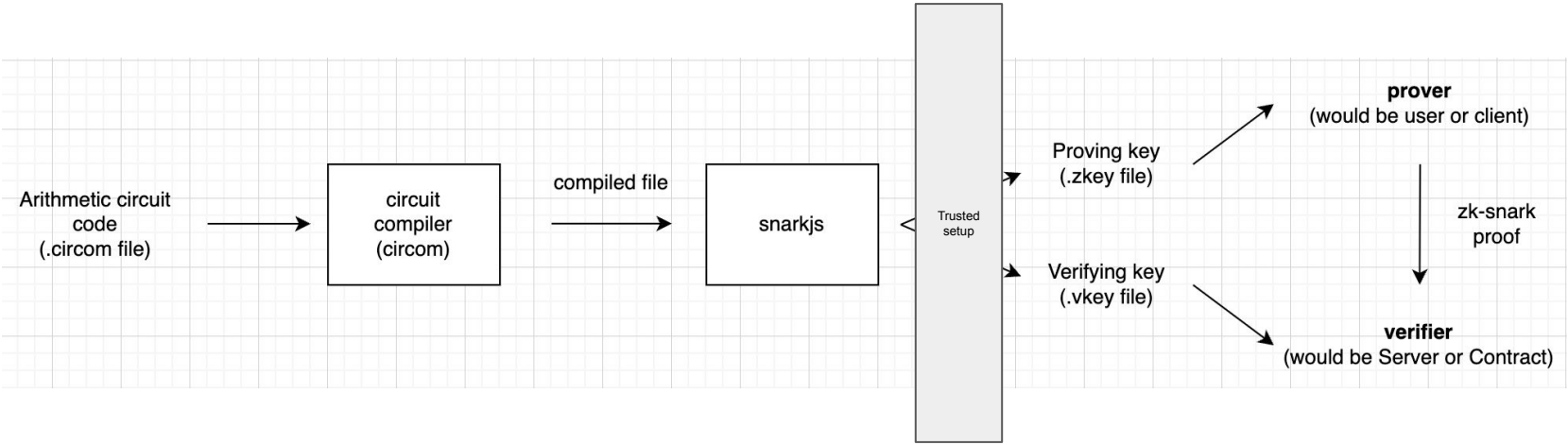
snarkjs


Arithmetic circuit code (.circom file)
circuit compiler (circom)
compiled file
snarkjs
Proving key (.zkey file)
Verifying key (.vkey file)
prover (would be user or client)
zk-snark proof
verifier (would be Server or Contract)

snarkjs

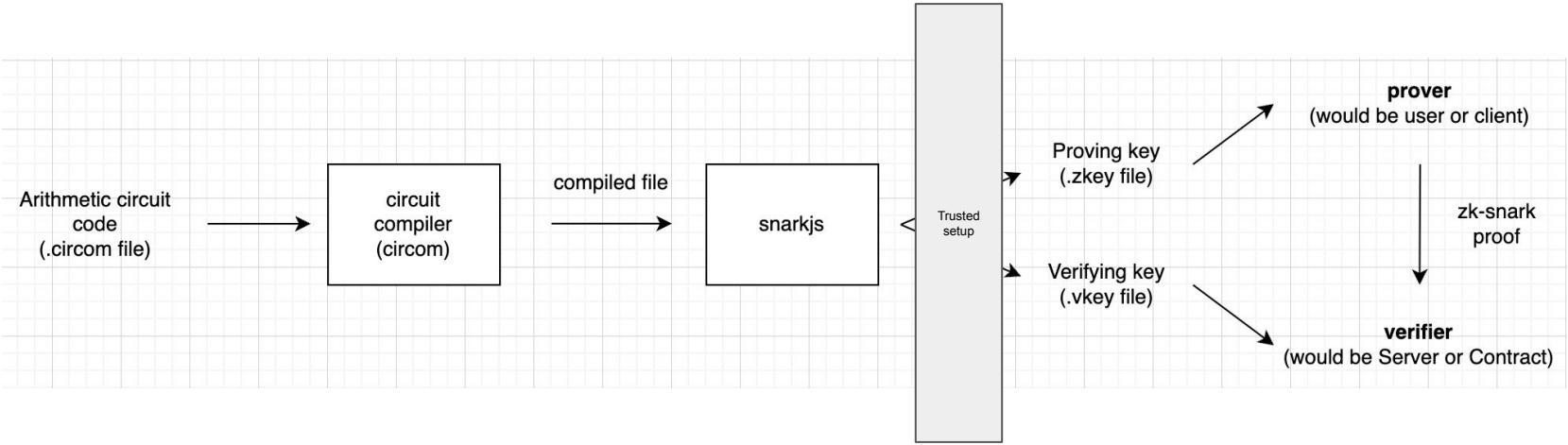

Arithmetic circuit code (.circom file)
circuit compiler (circom)
compiled file
snarkjs
Trusted setup
Proving key (.zkey file)
Verifying key (.vkey file)
prover (would be user or client)
zk-snark proof
verifier (would be Server or Contract)

snarkjs



Groth16: circuit별로 trusted setup 필요. (CRS)
Plonk: universial하게 trusted setup 필요. (SRS)

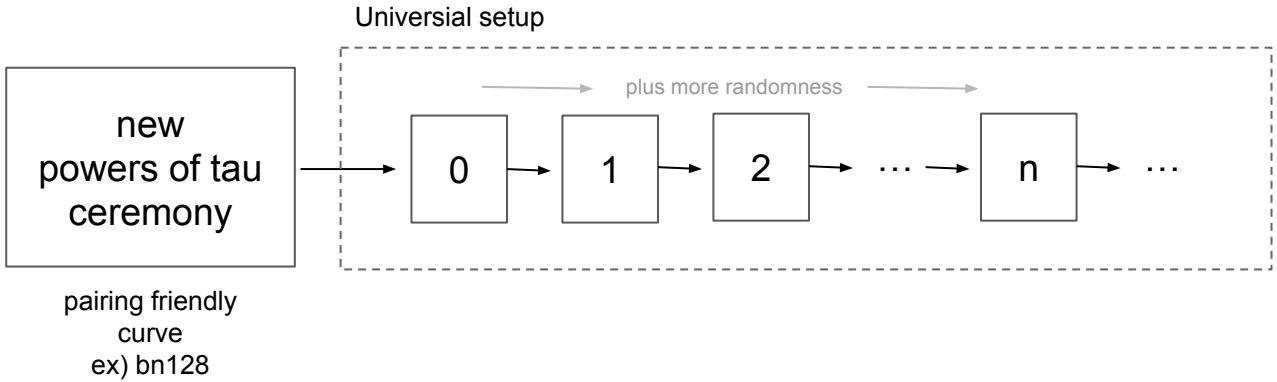
snarkjs



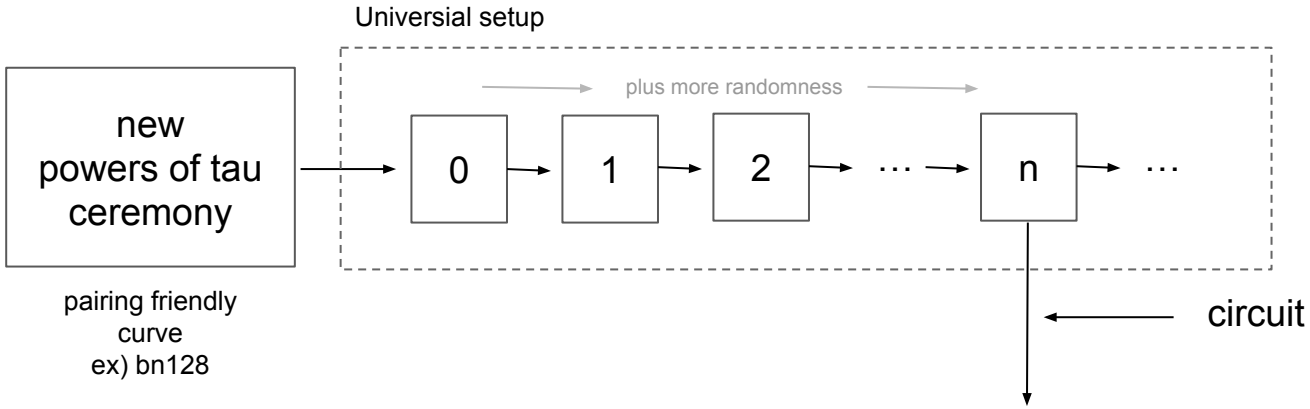
Groth16: circuit별로 trusted setup 필요. (CRS)
Plonk: universial하게 trusted setup 필요. (SRS)

New!! “FFlonk”
https://twitter.com/jbaylina/status/1624116186861404188

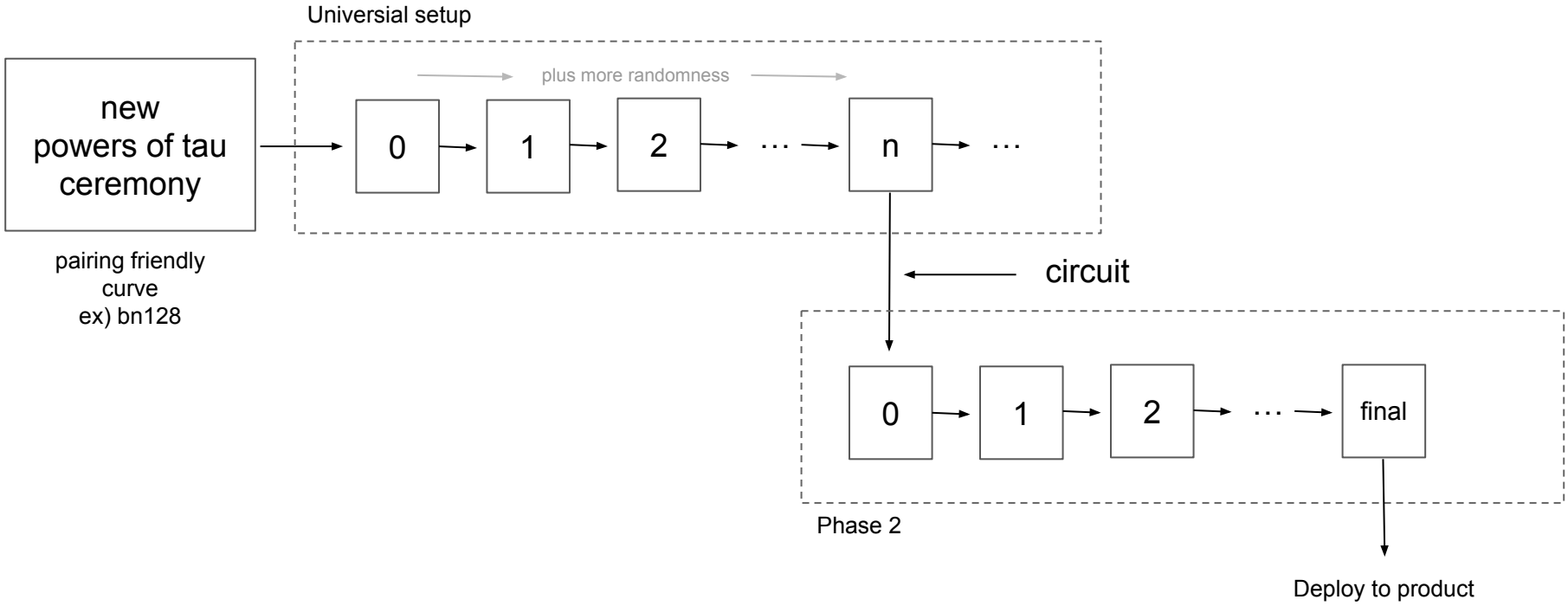
Powers of tau



Powers of tau



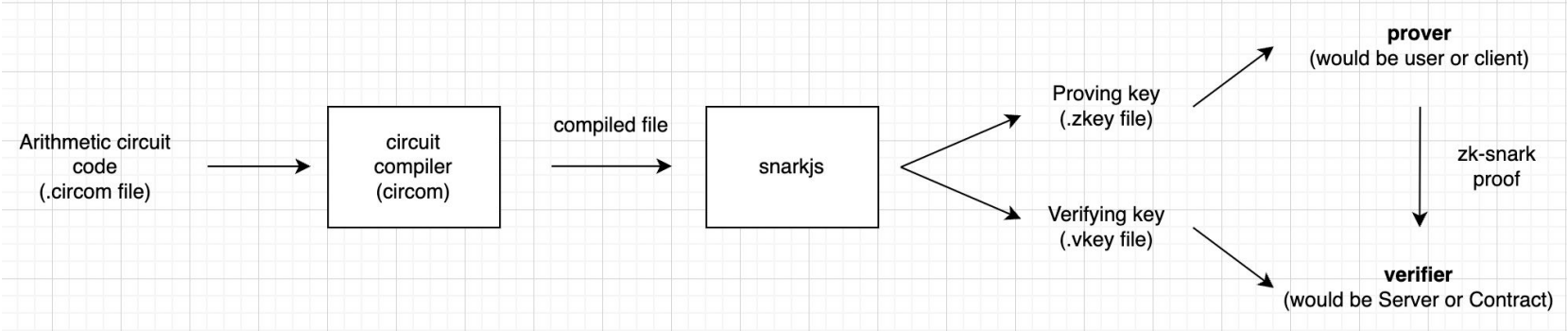
Powers of tau



snarkjs

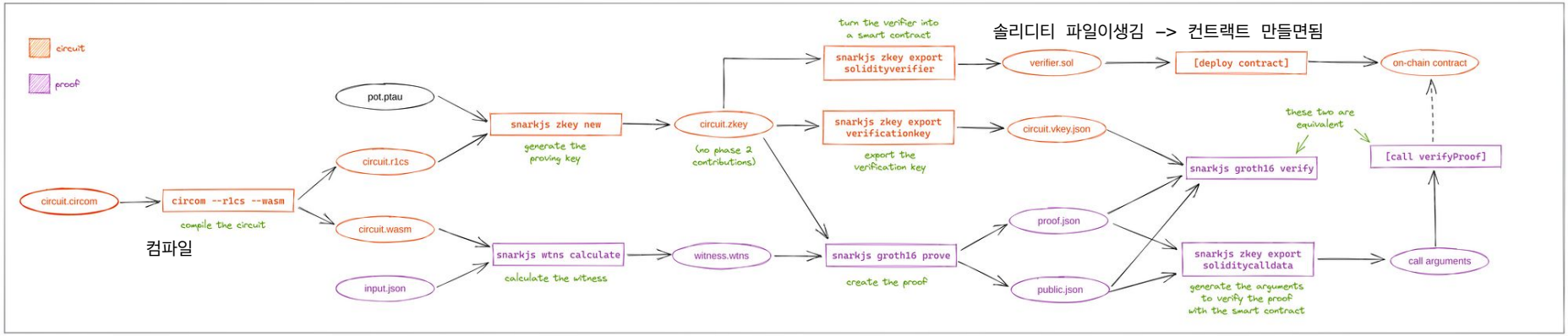
Key point!

Proving key와 witness로
proof 생성



Verifying key와 proof로
T/F 검증

snarkjs



- made by fvictorio