The theory behind zk-SNARKs

Kyosuke Tsubaki

Introduction

Kyosuke Tsubaki

Senior, Department of Computer Science, Tokyo University of Science

After interning at Gaudiy approximately one year ago, I currently work as an engineer developing the front and back ends of community applications.

Background challenges

Gaudiy services as examples of how zk-SNARKs are used

Providing IP-unique community platforms

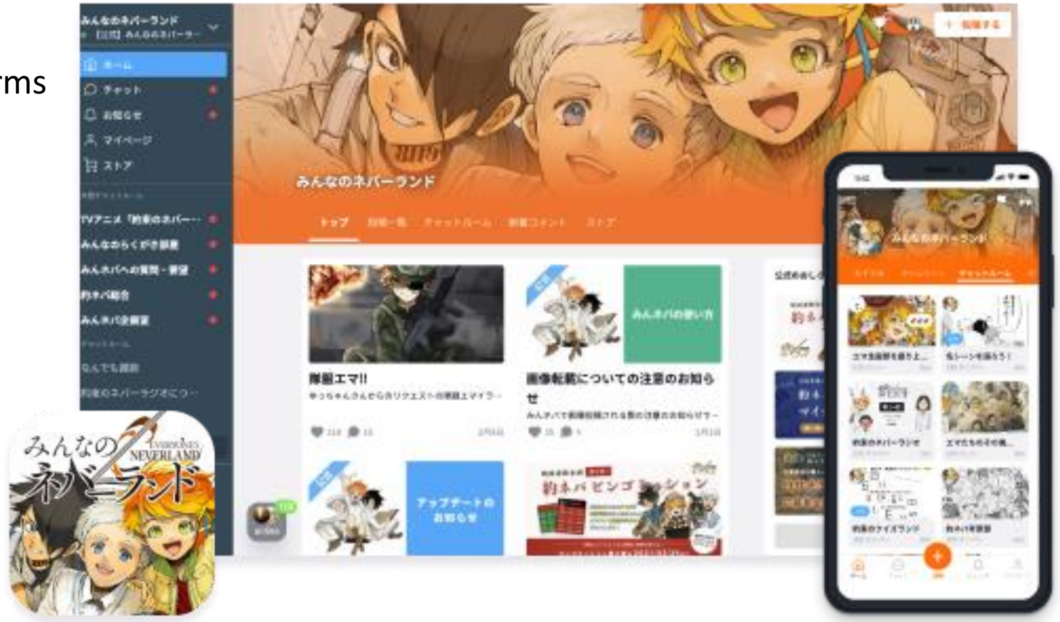
A fan community that uses blockchain to create a token economy

Provision of IP-based community platforms

Supports a variety of entertainment genres such as manga, anime, idols, and games

Packaging blockchain for society

Use magic links to provide concepts such as private keys and wallets that are difficult for end users to understand in a comprehensible form



Highly customizable and multifunctional

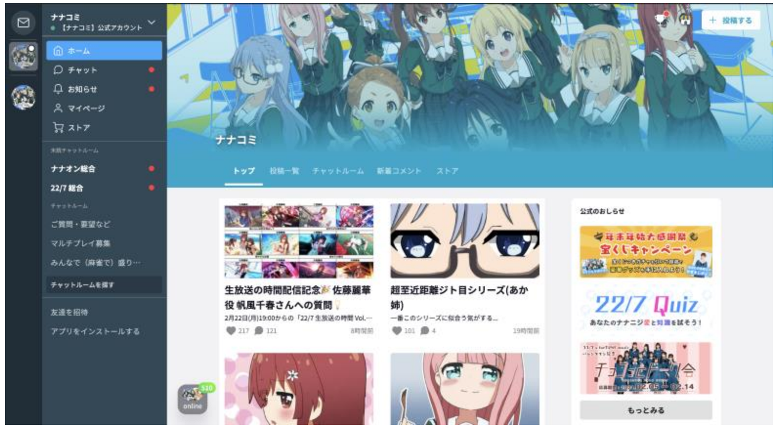
Use blockchain technology to develop payment systems, NFT digital trading cards, and other functionality.

Promote fan engagement and digital transformation from one’s own community

Example services

Use a blockchain technology called DID (decentralized identifier documents) to link different services (communities, games, fan clubs, etc.)

Communities



Ticketing

TICKET
forTUNE music
フォーチュンミュージック

Fan clubs

22/7
FANCLUB

Games

22/7
ナナブンノニジュウニ
音楽の時間

Digital Idol Group 22/7 economic bloc

Privacy issues

Communities are connected to external platforms via wallets

Problem:
The public nature of blockchains leads to privacy issues

Wants to check if other party has some information

Providing the information as-is will reveal unnecessary or confidential information

Game

Fan club

zk-SNARKs are a privacy-enhancing technology that allow for blockchain verification that guarantees the integrity and accuracy of confidential information.

What are zk-SNARKs?

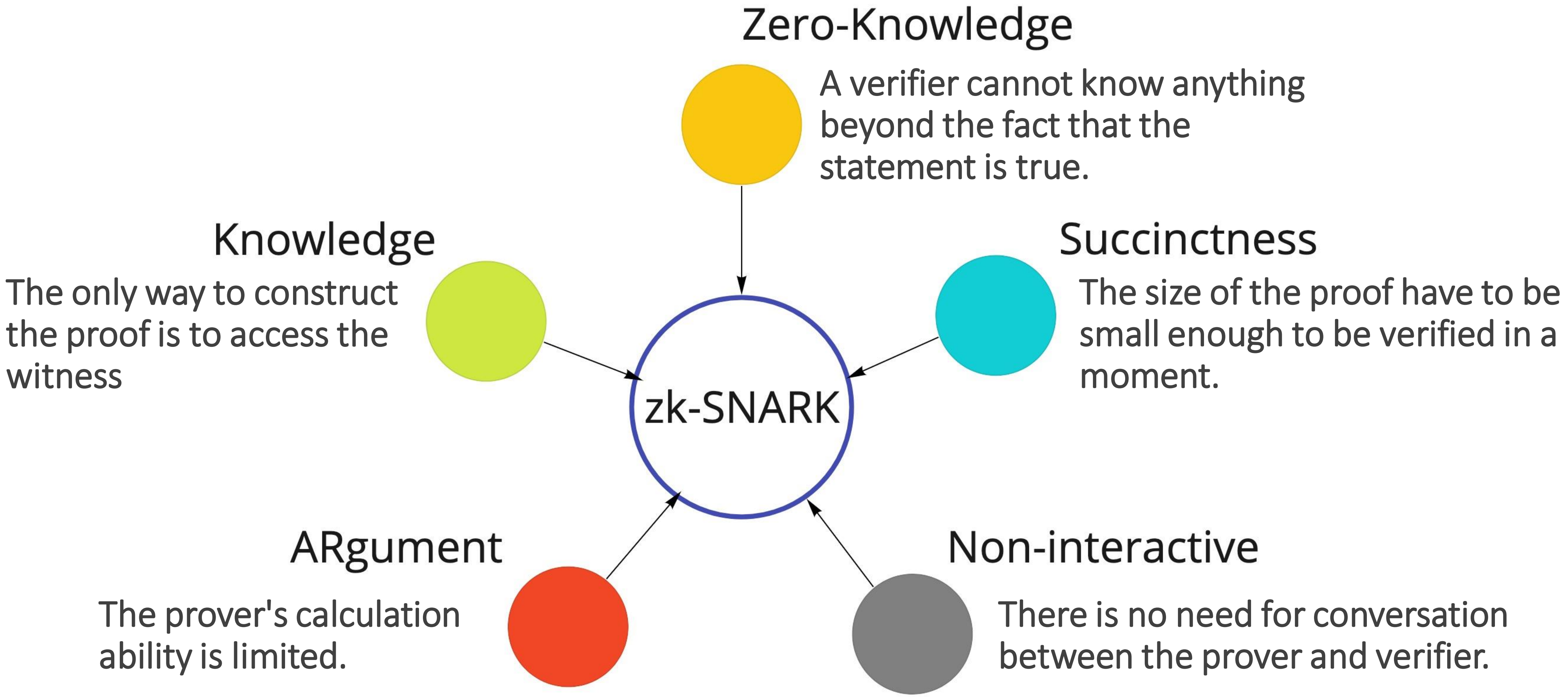
zk-SNARKs (Zero-Knowledge Succinct Non-interactive Arguments of Knowledge) are systems that enable zero-knowledge proofs that require no interaction.

They current uses include:

- Preserving the privacy of blockchain transactions
- Solving Ethereum’s scaling issues

zk-SNARKs are based on the "Pinocchio" protocol, a "system for efficiently verifying general computations while relying only on cryptographic assumptions." zk-SNARKs are succinct while also being highly secure.

Properties of zk-SNARKs

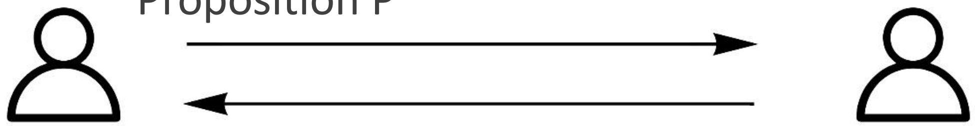


Zero-knowledge proofs: the most important concept needed to explain zk-SNARKs

What are zero-knowledge proofs?

Zero-knowledge proofs allow a prover to prove to a verifier that a certain piece of information is correct without revealing any other information other than that truth.

Proposition P
Age 20 or above



This will reveal personal information. One would prefer to prove only that one is age 20 or above without revealing age or other information (zero-knowledge proof).

Properties of zero-knowledge proofs

- Completeness
  - If a prover’s proposition is true, then a verifier is guaranteed to be able to confirm this truth.
- Soundness
  - If the prover's proposition is false, the verifier will be able to detect this falsehood with a high probability.
- Zero-knowledge
  - If the prover's proposition is true and the verifier attempts to steal knowledge from the prover, the verifier will be able to acquire no knowledge other than "the proposition is true."

Required knowledge before explaining how zk-SNARKs are structured

1. NP complexity class
2. Language L in NP

Complexity classes

- Class P

The set of all decision problems that can be solved in polynomial time

(Polynomial time means that calculations can be made in a realistic length of time.)

- Class NP

The set of all decision problems that can be verified in polynomial time

> Given some evidence w, one can decide in polynomial time whether evidence w is correct.

However, it may take exponential time to find a solution to the problem.

(NP problem examples: traveling salesman problem, Three Color Problem)

Zero-knowledge proofs of "Language L in NP"

The proposition that the prover seeks to prove: "For language L in NP and a statement x, $x \epsilon L$ holds."

- Example of a language L in NP

  "A set of correctly generated RSA public keys"

- Example of statement x

  "A specific public key"

- Evidence of evidence w

  "The prime factors of a public key" (private key)

Evidence w makes it easy to confirm whether a public key is included in the RSA public key set. However, it is difficult to obtain the prime factors from a public key.

The zk-SNARK process

- The function $f(x,w)$ outputs 1 only if the input is correct, and 0 outputs 0 if it is incorrect
- The prover wants to prove to the verifier that the prover has w such that $f(x,w)$=1 without revealing w itself.
- The prover creates a proof π that proves "I have the correct proof w" and sends this to the verifier.
- The verifier receives and verifies proof π, accepting it if it is true and rejecting it if it is not.

Non-interactive zero-knowledge proofs can be constructed by placing a "trusted third party" between the prover and verifier, allowing for verification that is efficient and secure.

The structure of zk-SNARKs

- Generator

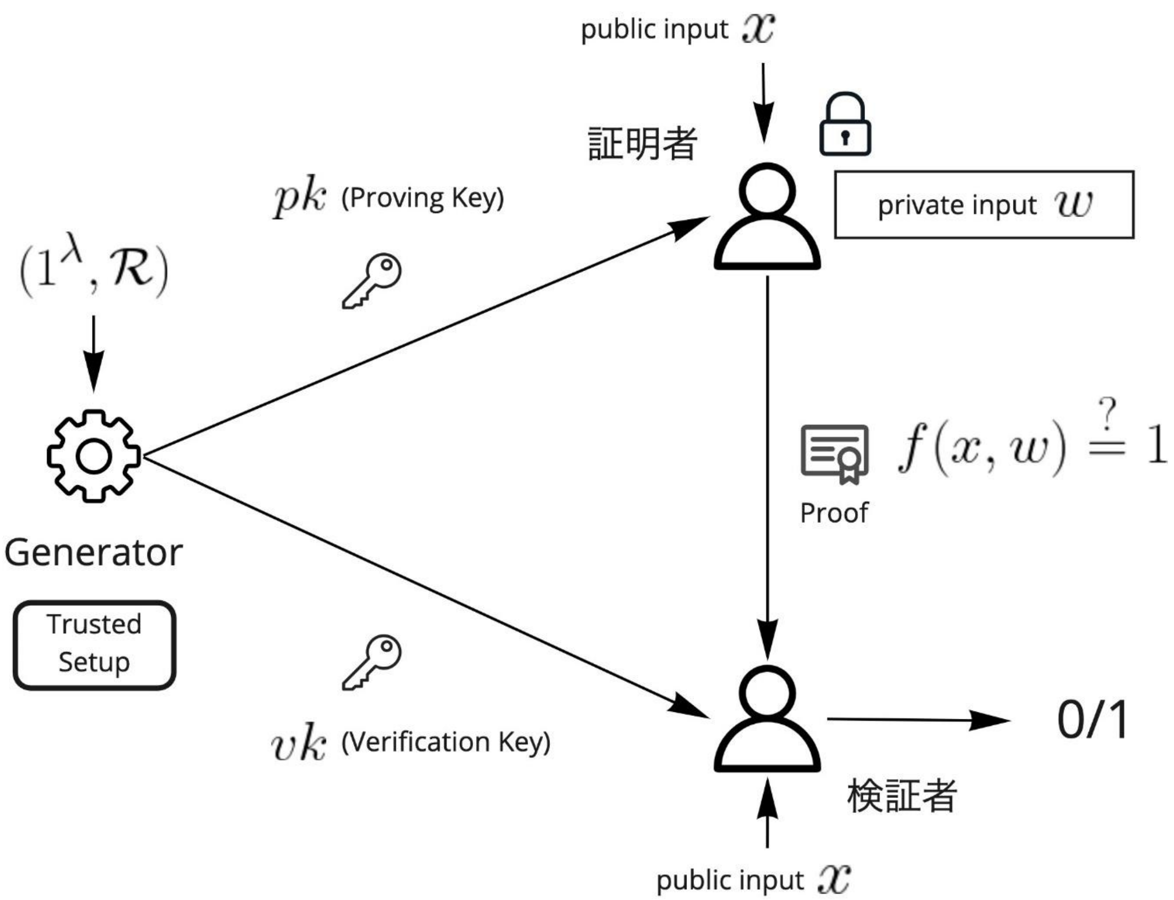
A trusted third party. In a setup phase, this party generates a CRS (common reference string) and respectively sends pk and vk to the prover and verifier

- Prover

Receives pk and x. Uses w to create proof π and sends it to the verifier.

- Verifier

Receives vk and proof π. Accepts if the proof is correct. Rejects otherwise.



Replace "has correct evidence w" with "knows a certain polynomial"

Converting to algebraic properties

"Has evidence w that $f(x, w) = 1$" → "Knows polynomial $h(x)$ that satisfies $h(x)t(x) = p(x)$"

- The function f can be expressed as an arithmetic circuit constructed from addition and multiplication gates.
- From this arithmetic circuit, one obtains a set of three polynomials $\{v_0(x), \dots, v_m(x)\}, \{w_0(x), \dots, w_m(x)\}, \{y_0(x), \dots, y_m(x)\}$ which represent the target polynomial $t(x)$ and the function f. (These are called left, right, and output polynomials.)
- By passing input into this circuit, one can acquire $c_1, \dots, c_m$. These coefficients represent the input and output of the circuit and the output values of each gate (intermediate values).
- A linear combination of $c_1, \dots, c_m$ and the set of three polynomials can be used to create a polynomial $p(x)$ in the following form.

$$p(x) = \left(v_0(x) + \sum_{k=1}^{m} c_k \cdot v_k(x)\right) \cdot \left(w_0(x) + \sum_{k=1}^{m} c_k \cdot w_k(x)\right) - \left(y_0(x) + \sum_{k=1}^{m} c_k \cdot y_k(x)\right)$$

- If the circuit input is "correct" $p(x)$ evenly divides $t(x)$ and there exists $h(x)$ such that $h(x)t(x) = p(x)$.
- If the input is "incorrect," even division is not possible and no $h(x)$ that satisfies the above exists.

How these polynomials are represented to the prover and verifier

Polynomial verification

Verify whether the sent polynomial is correct

$$h(x)t(x) \overset{?}{=} p(x)$$

A linear combination of the set three polynomials and $c_1, \dots, c_m$ obtained from the arithmetic circuit

$$p(x) = \left(v_0(x) + \sum_{k=1}^{m} c_k \cdot v_k(x)\right) \cdot \left(w_0(x) + \sum_{k=1}^{m} c_k \cdot w_k(x)\right) - \left(y_0(x) + \sum_{k=1}^{m} c_k \cdot y_k(x)\right)$$

$$p(x) = \big(v_0(x) + v(x)\big) \cdot \big(w_0(x) + w(x)\big) - \big(y_0(x) + y(x)\big)$$

$t(x), v_0(x), w_0(x), y_0(x)$

Verifier

Proof π= $h(x), v(x), w(x), y(x)$

Proof of possession of $h(x)$ such that $h(x)t(x) = p(x)$

Prover

Obtained by passing input to circuit

$c_1, \dots c_m$

Values known only by prover

Making polynomials zero-knowledge

Encrypt the polynomials

One-way function E

$$E(x) = g^x$$

Property:

Producing g^x from x is simple. However, finding x from g^x is difficult (discrete logarithm problem)

Use one-way function $E(x)$ to encrypt $h(x), v(x), w(x), y(x)$ and send them to the verifier.

As determining the details of these polynomials would be difficult, the verifier is unable to steal the values of $c_1, \ldots c_m$. Knowing $c_1, \ldots c_m$ would mean knowing the input that satisfies the circuit.

Proof $\pi = E\big(h(x)\big), E\big(v(x)\big), E\big(w(x)\big), E\big(y(x)\big)$

Perform verification on encrypted polynomials

Verification by pairing

For example, the prover can prove knowledge of $(a_1 a_2 a_3)$ that satisfy $x_1 x_2 {+} {x_3}^2 = 0$ without revealing their values to the verifier.

$$e(g,g)^{a_1 a_2 + {a_3}^2} = \mathrm{e}(g^{a_1}, g^{a_2}) \cdot e(g^{a_3}, g^{a_3}) \stackrel{?}{=} e(g,g)^0$$

When a map e is bilinear, that map is called a pairing.

Pairing allows one to achieve such technologies as ID−based encryption, searchable encryption, and functional encryption.

Polynomial verification by pairing

$$
\begin{aligned}
e\left(g^{h(x)}, g^{t(x)}\right) &= e(g,g)^{h(x)t(x)} \\
&= e(g,g)^{(v_0+v(x))\cdot(w_0+w(x))-(y_0+y(x))} \\
&= e(g^{v_0(x)} g^{v(x)}, g^{w_0(x)} g^{w(x)}) / e(g^{y_0(x)} g^{y(x)}, g)
\end{aligned}
$$

Determining the particulars of the polynomials is difficult, but one can verify whether $h(x)t(x) = p(x)$ holds.

$t(x), v_0(x), w_0(x), y_0(x)$

Proof $\pi = g^{h(x)}, g^{v(x)}, g^{w(x)}, g^{y(x)}$

Verifier

Prover

Achieve non-interactive zero-knowledge proofs by placing a "trusted third party" between the prover and verifier

Non-interactive zero-knowledge proofs

pk

$\{g^{v_0(x)}, \dots, g^{v(x)}\}, \{g^{w_0(x)}, \dots, g^{w(x)}\}, \{g^{y_0(x)}, \dots, g^{y(x)}\}$

vk

$g^{t(x)}, g^{v_0(x)}, g^{w_0(x)}, g^{y_0(x)}$

$$
\begin{aligned}
g^{v(x)} &= g^{v_0(x)+c_1 \cdot v_1(x)+\cdots+c_m . v_m(x)} \\
&= g^{v_0(x)} \cdot g^{c_1 \cdot v_1(x)} \cdot \dots \cdot g^{c_m . v_m(x)} \\
&= (g^{v_0(x)})^1 \cdot (g^{v_1(x)})^{c_1} \cdot \dots \cdot (g^{v_m(x)})^{c_m}
\end{aligned}
$$

$$
e\left(g^{h(x)}, g^{t(x)}\right) = e(g^{v_0(x)} g^{v(x)}, g^{w_0(x)} g^{w(x)}) / e(g^{y_0(x)} g^{y(x)}, g)
$$

Generator

vk

pk

Obtained by passing input into arithmetic circuit

$c_1, \dots, c_m$

private input w

Proof $\pi = g^{h(x)}, g^{v(x)}, g^{w(x)}, g^{y(x)}$

$0/1$

Verifier

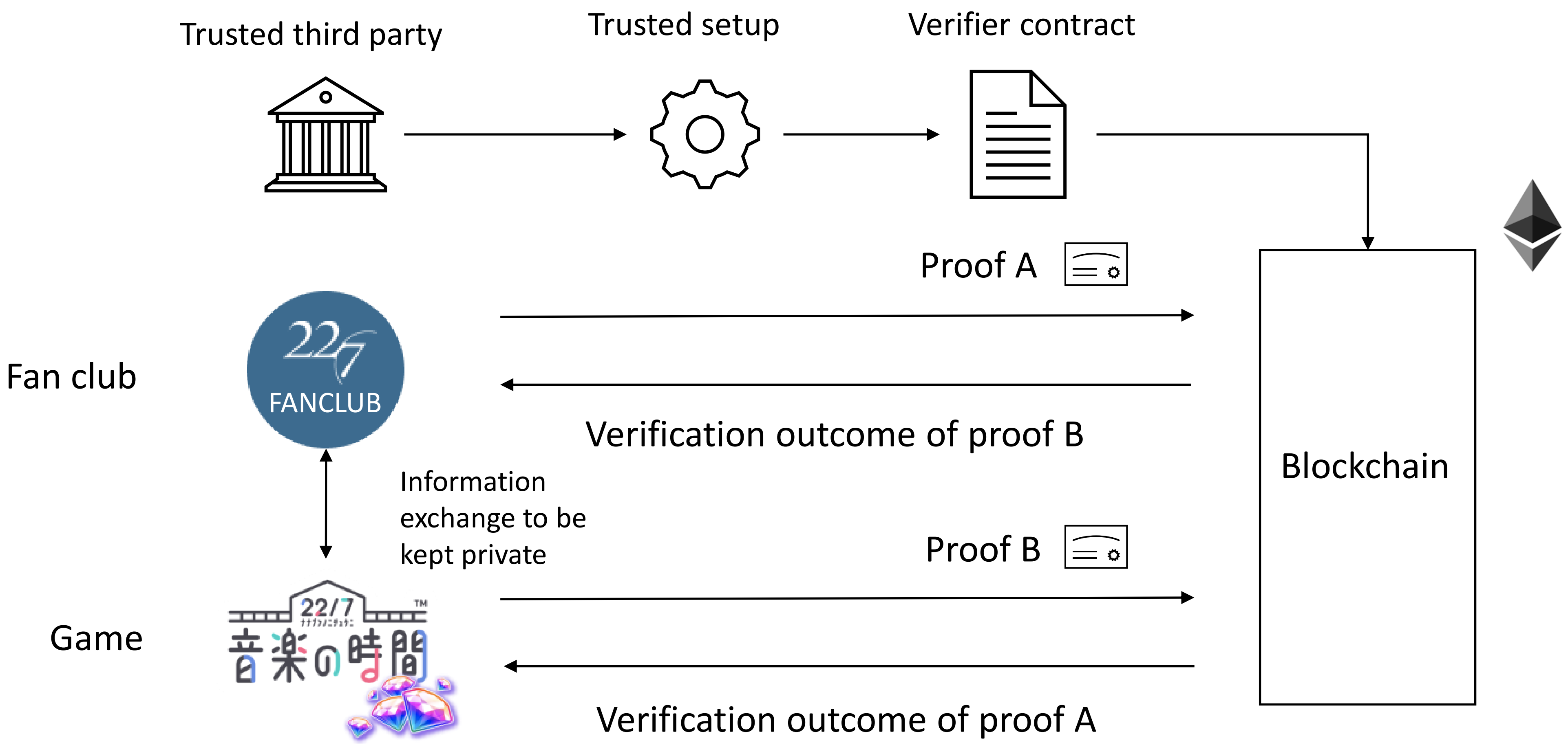
Prover

public input x

public input x

Solving privacy issues

Problems zk-SNARKs can solve



Conclusion

We looked briefly at how zk-SNARKs work.

Things we did not cover:

- A prover's ability to perform a calculation in polynomial time does not guarantee that the calculation result will be sent.

  Fraudulent activity is more difficult to achieve than with the Pinocchio protocol's q-PKE assumption.

- More efficient polynomial verification

  The Schwartz-Zippel lemma allows one to easily detect invalid polynomials.

- Complete zero-knowledge proofs

  Randomized proofs