



ZK MPC: Bring public auditability into MPC

Yusuke Nakae & Task Ohmori

Yoi Inc.

Session Outline

1. Introduction
2. Collaborative zk-SNARKs (Public Auditable MPC)
3. Secure consistency of secret inputs
4. Bitwise Operation Support
5. Applications & Demo
6. Future prospects
7. Q&A



Section 1

Introduction

Team & Project Introduction

- **About Yoi Inc.**
 - Japanese Fintech company
 - Helps entrepreneurs and businesses grow by increasing asset liquidity with innovative financial solutions
- **Our Expertise**
 - Research and Development in ZKP & MPC
 - Focus on secure and efficient computation
- **Achievements**
 - Received two ESP grants from the Ethereum Foundation

Team member Introduction



Yusuke Nakae
X, Telegram: @sheagrief
Researcher & Developer



Task Ohmori
X, Telegram: @taskooh
PM & Co-Founder CTO



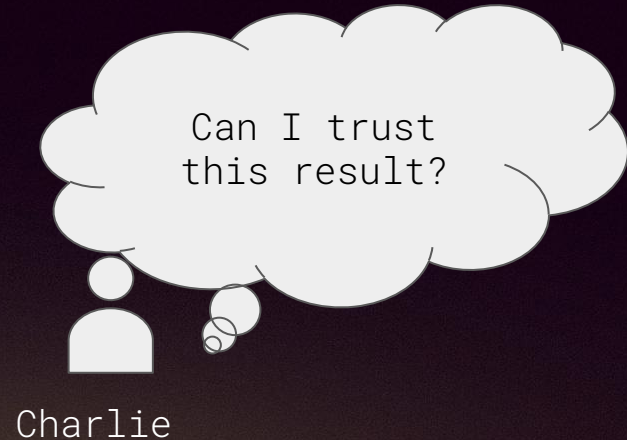
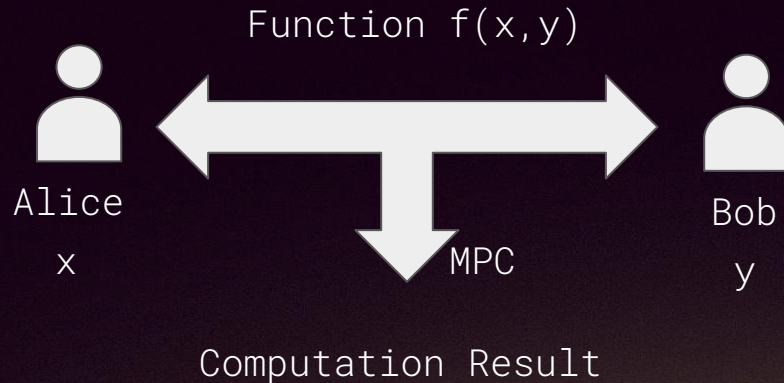
Masaharu Uno
X: @Masa_Ukov
Telegram: @masauno
Co-Founder CEO

The Importance of Secure Computation and MPC

- **Secure Computation in a Data-Driven World**
 - Protects sensitive data (e.g., healthcare, financial)
 - Enables useful analysis without exposing information
- **Multi-Party Computation (MPC)**
 - Jointly computes functions while keeping inputs private
 - Shares only final results, ensuring secure collaboration
 - MPC addresses the growing need for privacy-preserving technologies

Third-party verification for MPC is essential

- When using a dishonest majority MPC protocol such as **SPDZ**, MPC participants can verify that the calculation results are computed correctly. However, they cannot directly prove its correctness to a third party.
- Ex. blockchain, finance, medical data analysis



Overview

- **Public Auditable SPDZ (MPC protocol).**
 - Introduce collaborative zkSNARKs
- **Secure consistency of Secret input**
 - Enables third-party verification of input constraints using Pedersen Commitment.
- **Bitwise Support for Public Auditable SPDZ**
 - SPDZ is well-suited for **arithmetic operations** but less effective for **bitwise operations**.
 - Support comparisons, conditional branching, bit decomposition and so on, expanding range of possible applications.
- **Application & Demo**
 - Game Master (GM)-free werewolf game using MPC.



Section 2

Collaborative zk-SNARKs (Public Auditable MPC)

MPC protocol, SPDZ

- The basics of MPC
 - Secret sharing
 - $\langle x \rangle$: the share of x by multi party.
 - Additive secret sharing, Shamir secret sharing, etc...
 - Examples: Addition
 - $\langle x+y \rangle = \langle x \rangle + \langle y \rangle$
- SPDZ is one of the MPC protocols, proposed in 2011
 - For dishonest majority (by MAC value)
 - Suitable for arithmetic operation
 - Additive secret sharing
 - There are two phases, preprocessing (setup & multiplication triple generation etc) and online (calculation).

Publicly Auditable MPC

- In ordinary MPC, a third party can't verify the correctness of MPC calculation.
 - It is only for the "secure" computation.
- Is there notion of multi party computation which can be verified by third party?

-> Publicly auditable MPC! ([Baum, Damgard, Orlandi, 2014])

Introducing the Collaborative zk-SNARKs

- Collaborative zk-SNARKs [Ozdemir, Boneh 2021]
 - Idea of proof generation by multi prover
 - Sharing KZG commitment
- What is KZG commitment?
 - Prover P wants to show for verifier V: The value of the polynomial f at the point x is y `` $f(x)=y$ ''.
 - Commit
 - P computes $c=f(\alpha)G$ and sends V
 - Prove
 - P computes polynomial $q(X)$ from $(f(X)-f(x))/(X-x)$ and sends proof $\pi=q(\alpha)G$ to V.
 - Verify
 - V checks pairing $e(\pi, \alpha H - xH) = e(c - yG, H)$.

Idea of proof generation

ZKP proof consists of polynomial commitment.

This commitment is shared by secret sharing scheme.

The prove generation scheme is divided into multi provers.

- In some zk-SNARKs (e.g. Marlin, PLONK), polynomial commitment is used for the proof generation.
- The proof is verified by evaluating that commitment through pairing.

Idea of proof generation 2

- Idea of collaborative proof

In “Collaborative proof” (Ozdemir, Boneh, 2021), proof is generated by MPC. Specifically, it is based on the linearity of polynomial commitments as follows.

- Claim

Let coefficients of poly f be f_0, f_1, \dots, f_d . Shared coeffs are $[f_0], [f_1], \dots, [f_d] \rightarrow$ shared polynomial $f(i)$

Each party P_i generates shared proof $\pi(i)$ as above. Then, shared proof is share of total proof $\pi = \sum \pi(i)$

$$\begin{aligned}\sum_i \pi^{(i)} &= \sum_i q^{(i)}(\alpha) G = \sum_i \frac{f^{(i)}(\alpha) - f^{(i)}(x)}{\alpha - x} G \\ &= \frac{f(\alpha) - f(x)}{\alpha - x} G \\ &= \pi\end{aligned}$$

Issues: challenges with MPC on Blockchain

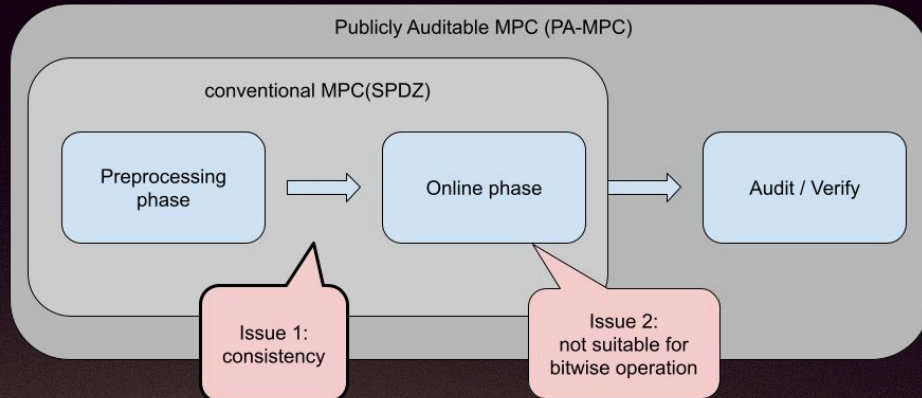
- For blockchain use, it requires that consistency of secret input and committed value.



We want to develop techniques in this region!

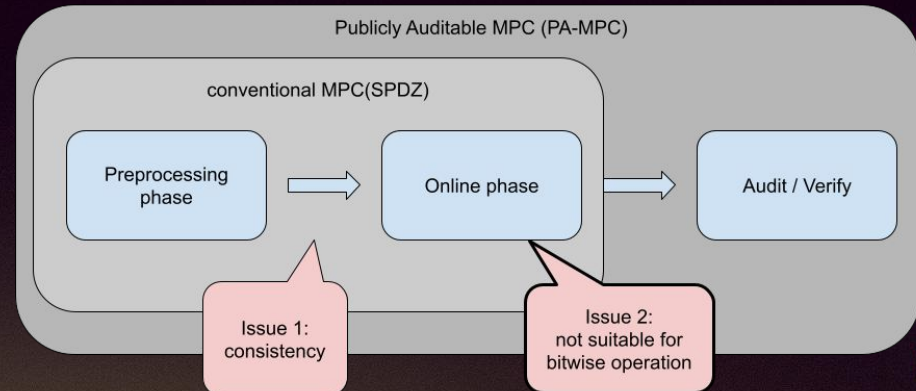
Issue 1: Consistency of Secret Inputs

- SPDZ operates in two phases: preprocessing and online.
 - This two-phase structure is designed for efficiency.
- A malicious party may use different values for MPC calculation and proof generation.
- **Ensuring consistency of secret inputs across phases is essential.**



Issue 2: Requirements for the general-purpose computations

- SPDZ based, MPC is suitable for arithmetic operations like, addition multiplication.
 - But need for bit-oriented calculation.
- This needs links with the constraint generation.
- To support such operations, arise some technical problems.
 - Need for using two finite fields.





Section 3

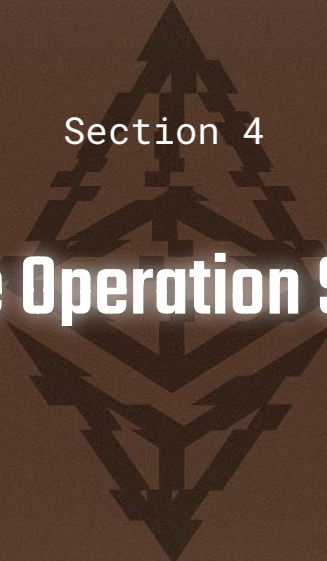
Secure consistency of secret inputs

Secure consistency of secret inputs : Solution of adding zkp constraints

- In MPC, input shares are generated using data prepared by the data holder during preprocessing. Therefore, the input value must be committed before creating the input shares.
- The relationship between the committed input and the actual input value is added as a proof constraint. This verifies that the data used in computation matches the data used in the proof, ensuring consistency of the input values.

Secure consistency of secret inputs : Implementation

- **Adoption of Pedersen Commitment**
 - We use Pedersen Commitment to secure and verify secret inputs.
 - The constraints is about 7000 in the case of 256 bit fields for one input.
- **Constraints on Input Values**
 - We implemented constraints, such as range limits, help ensure input values (e.g., training data) fall within valid boundaries.
 - For inputs with strict constraints—such as outputs defined by specific requirements or pre-signed data—commitment-based verification may offer relative advantages in maintaining input integrity.



Section 4

Bitwise Operation Support

Bitwise Operation Support: Solution

- Recall; Why are bitwise operations difficult in SPDZ?
 - Bitwise operations (e.g., bit decomposition and comparison) are significantly more costly than addition or multiplication.
 - Using larger finite fields in MPC increases the constraints needed for these operations, and limitations in the finite field used for ZKP further add to the computational cost.
- Support bitwise operations in ZK MPC
 - Both in calculation & constraint generation(ZKP).
 - Equality Zero Test, Bit Decomposition, Comparison, If function

Bitwise Operation Support: Implementation contents

- Adding subprotocol
 - These can be executed on linear sharing scheme-based MPC, like SPDZ.
 - We implemented EqualityZeroTest($[x=0]$), Comparison($[x < c]$), [Nishide, Ohta 2007]
 - Since bit decomposition is basically devilishly heavy, the method is adopted in a way that does not involve bit decomposition.
- Share conversion protocol
 - Between on two different finite fields.
 - Since the Pedersen Commitment uses different finite fields, it is necessary to have a protocol for conversion between each share.
 - [KIMHC 2018]
- Introducing above protocols into constraint generation in zkp.

Bitwise Operation Support: algorithm details

- Sub protocol example: Equality Zero Test ($[x==0]$)
 - Generate random share $[r]$
with bitwise share $[r]_B = \{ [r_0], [r_1], \dots [r_n] \}$
 - Open $[x+r]$, denote as c .
 - Calc bit decomposition of c , denote $\{c_0, c_1, \dots c_n\}$
 - Calc by MPC $a_i = r_i$, if $c_i = 1$, $a_i = 1 - r_i$ if $c_i=0$.
 - **$a_i=1$ if and only if $c_i = r_i$!**
 - Output $\text{AND}_{\{i=0\}^n} a_i$, the share means 1 if $x==0$;
otherwise, this share means 0.

Bitwise Operation Support: effect for constraints

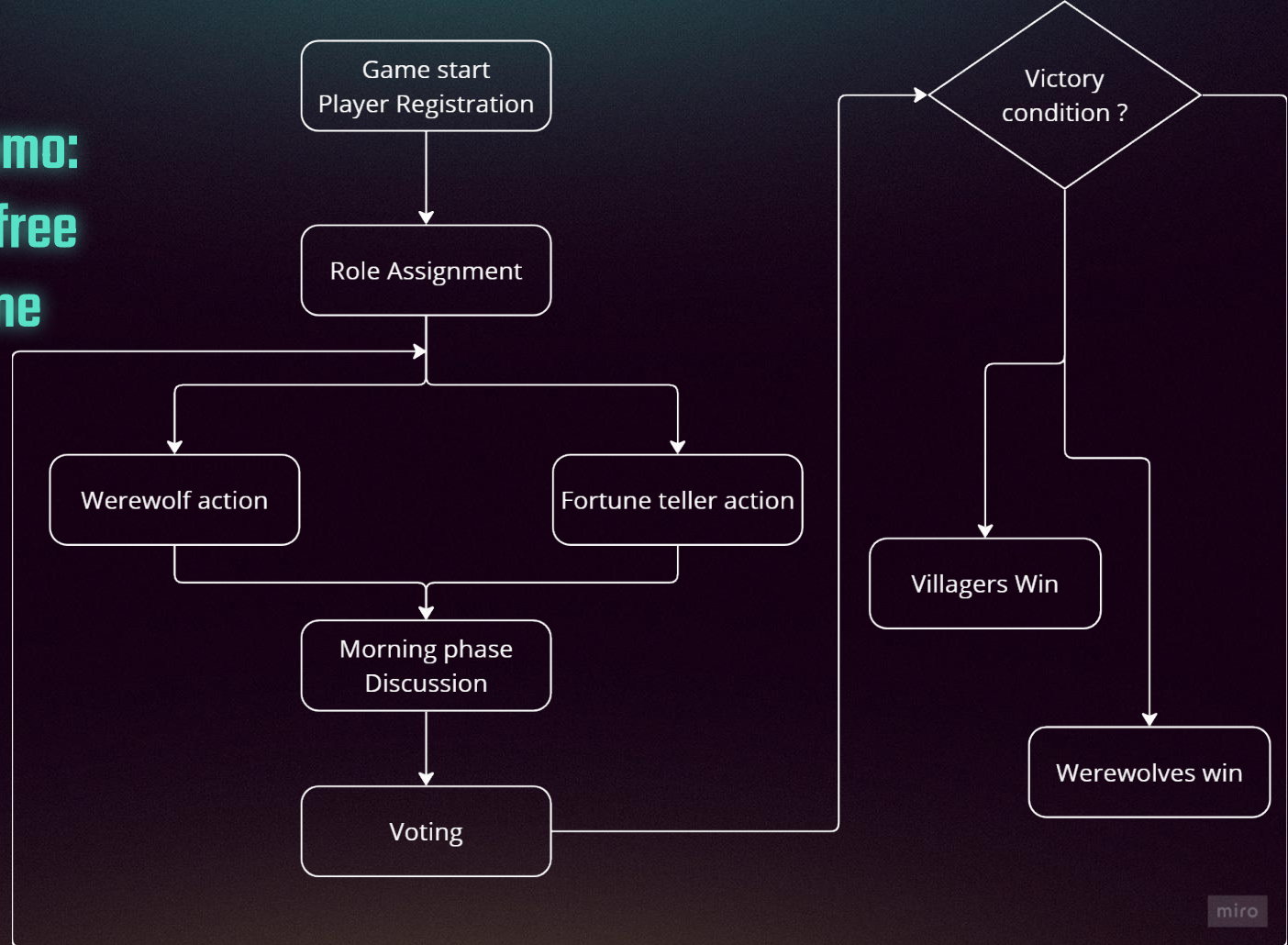
- When generating the constraint, there is no need for a constraint in the MPC calculation portion of each share.
- Even if the complete constraint is not described for each share, the overall value will be correct if the constraint is valid for the value when restored from the share.



Section 5

Applications & Demo

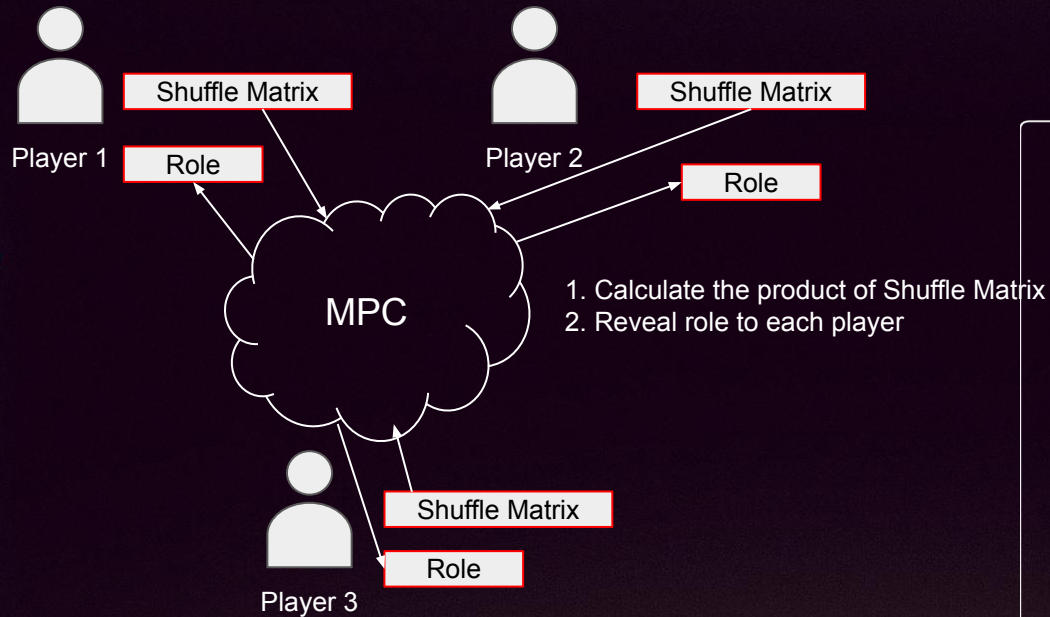
Application Demo: Game Master-free Werewolf game



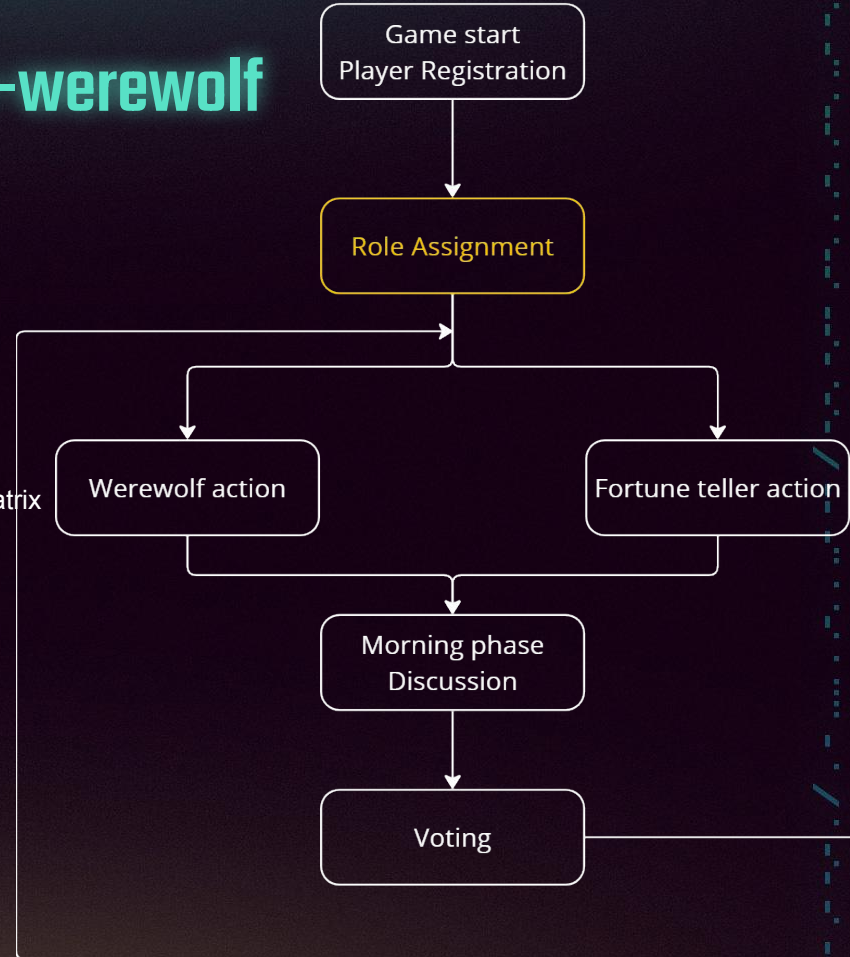
Collaborative Features we have implemented

- **Role Assignment**
 - Roles are assigned randomly, and each player should remain consistent in their role throughout the game.
- **Anonymous Voting**
 - The number of votes for each player remains hidden; only the player with the most votes is revealed.
- **Werewolf Attack**
- **Fortune Teller Check**
 - The identity of the Fortune Teller is kept secret. They may check only one player per round, with the result visible only to them.
- **Victory Condition**
 - Each player's role is hidden. The remaining number of werewolves is not revealed; only the outcome of the condition is provided.

Application Demo: Architecture of zk-werewolf



[Werewolf] Role Assignment





Section 6

Future prospects

Future Prospects

- **Potential Applications of ZK MPC on Blockchain**
 - smart contracts and digital identity verification.
 - smart contracts and financial transactions.
- **Another Application regions:**
 - Machine Learning (like federated learning)
 - Medical & Financial data use case
- **Benefits of ZK MPC:**
 - Emphasize the security advantages, such as the elimination of data leakage risks.



Thank you for listening!

Yusuke Nakae

X,Telegram: @sheagrief

Task Ohmori

CTO Yoi Inc.

X,Telegram: @taskooh

Q&A

- For Those Interested in Learning More:

[Github Repository](#)



[ETHReserch Post](#)

