Ph.D. Dissertation Defense

Anonymous Cyber Threat Intelligence Sharing on Blockchain

Committee Members

Dr. Yoohwan Kim - Advisory Committee Chairperson, Professor - Computer Science Department

Dr. Ju-Yeon Jo - Advisory Committee Co -Chairperson, Professor - Computer Science Department

Dr. Laxmi Gewali - Professor - Computer Science Department

Dr. Zuobin Xiong – Assistant Professor - Computer Science Department

Dr. Tina Vo - Assistant Professor – College of Education - Teaching and Learning Department

Chol Hyun Park

Department of Computer Science

Feb.28.2025

UNLV

Agenda

I Recap

II Introduction

III Application of Zero-Knowledge Proof Method

IV Implementation

V Result

VI Discussion

VII Conclusion and Future Works

I. Recap

1. Previous Works
2. Current Works

Current Works

Description	Dec 2024	Jan 2025	Feb 2025	Mar 2025

Description	Mar 2025	Apr 2025	May 2025
Interactions between ZK-SNARK, smart contract, and IPFS defined. Submit Dissertation			
Prototype to convert log/STIX data to CADL structure. Write Conference/Journal			
Implement and test ZKP + Ring Signature. Submit Conference/Journal			
Develop a user-friendly web interface for ZKP requests and CTI access. Graduate			
Measuring Performance			
Dissertation Defense			

II. Introduction

1. Cyber Threats
2. Problems/Challenges
3. Current Industrial Solution
4. Related Research
5. Proposed Solution

Rising Cyber Threats

Cyber Threats

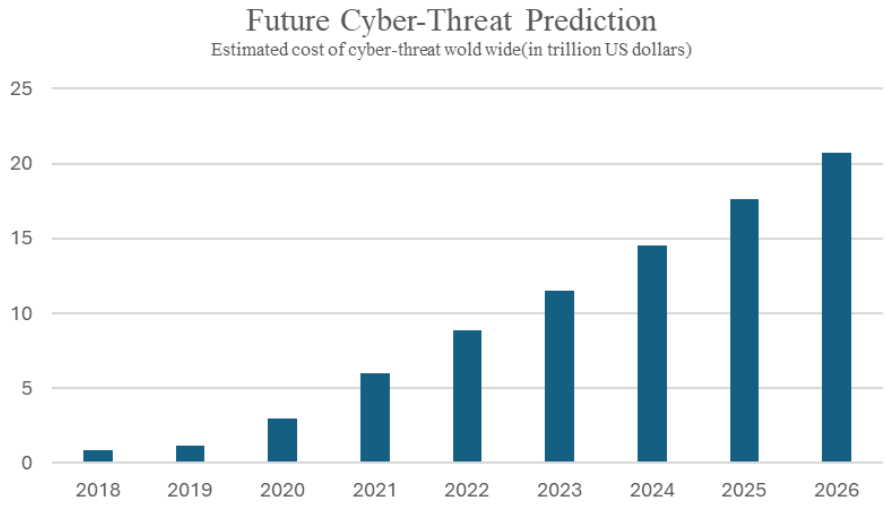
What is Cyber Threat?

A cyber threat is a potential malicious event that could compromise a system, while threat int elligence involves analyzing these threats to anticipate and counter them. Unlike attacks (act ual exploitation attempts) or vulnerabilities (system weaknesses), threat intelligence focuses on understanding and preparing for evolving adversarial capabilities.

01 Global Increase in Cyber Threats
Rising attacks target organizations of all sizes.

02 Importance of CTI Sharing
Organizations lack resources to counter all threats.
Quick sharing reduces risk.

03 Limitation of Existing Sharing Systems
Sharing sensitive data poses privacy risks and potential misuse concerns.
Current technologies fail to ensure both privacy and security in CTI sharing.



Problems / Challenges

- ✓ Balancing Transparency & Privacy
- Trust Issues in Collaborative Networks
- Lack of Strategic Perspective
- Insufficient Context
- ✓ Inconsistent Documentation
- Subjectivity in Reporting
- Evolving Tactics
- ✓ Reporting Hesitancy
- Limited System Details

Cyber Threat Intelligence Sharing

Government Initiatives

- CISA (US): National cybersecurity coordination
- NCSC (UK): Public-private sector guidance.
- ENISA (EU): EU-wide intelligence exchange.
- ISAC: Industry-specific Information Sharing and Analysis Centers.

How Is CTI Shared?

- Sharing Mechanism: Through secure web platforms and APIs (e.g., STIX/TAXII)[11].
- Access:
  - Public: Basic threat information may be open.
  - Restricted: Detailed intelligence often requires registration or is limited to authorized entities (e.g., government agencies, ISAC members).
- Time Delay:
  - Real-Time: Automated platforms like CISA's AIS share data almost immediately.
  - Reviewed Sharing: Sensitive data may require a manual review, taking hours to a day.

Challenges For Academic Research

Why is CTI Sharing Research Challenging?

- CTI sharing requires balancing data confidentiality, anonymity, trust, and scalability, which demands interdisciplinary solutions combining cryptography, networking, and security policies.
- Sharing cyber threat intelligence may involve sensitive data, raising concerns over compliance with data protection laws and ethical considerations.

Why is There Limited Academic Research?

- Most academic research focuses on cryptographic techniques (e.g., ZKP, Ring Signature) rather than their direct application to CTI sharing.
- To date, there is little to no research directly addressing the challenges of CTI sharing in a comprehensive and scalable manner.

Related Research

Previous Research

Applying ZKP and Blockchain to Enable Secure, Anonymous, and Efficient Data Sharing Across Various Domains

No existing research specifically focuses on applying ZKP to anonymize and verify CTI data sharing, leaving an open gap to develop a solution tailored for CTI ecosystems.

01 **Authentication Scheme Based on Non-Interactive Zero-Knowledge Proof for Mobile Health**

Focuses on authenticating sensitive health data transmissions without revealing patient information [1].

02 **Privacy-Preserving Authentication Scheme for Connected Electric Vehicles Using Blockchain and Zero Knowledge Proofs**

Shows how combining blockchain and ZKP can provide anonymous authentication, removing reliance on central authorities [2].

03 **Research on the Security Sharing Method of Power Grid Dispatching Control Data Based on Alliance Blockchain and Zero-Knowledge Proof Technology**

Integrates alliance blockchain and ZKP to securely share sensitive industrial control data. Leverages multi-channel blockchain and ZKP to enable cross-departmental collaboration without revealing confidential control details [3].

Related Product

Current Product

Related Technologies for Privacy and Anonymity

ZK-SNARKs and Ring Signatures are widely used in privacy-focused systems, ensuring confidentiality and anonymity in large-scale applications. ZK-SNARKs validate transactions without revealing sensitive details, while Ring Signatures anonymize senders by masking their identity within a group.

01

Zerocash: Decentralized Anonymous Payments from Bitcoin

ZK-SNARKs are widely used in privacy-focused financial networks, enabling secure and confidential transactions at scale by validating data without revealing sensitive details like sender, receiver, or amounts [4].

02

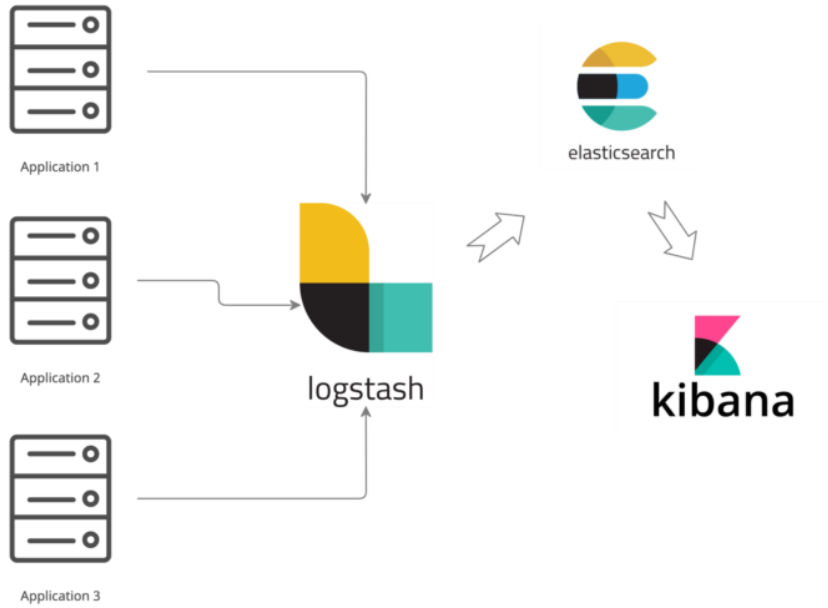
Ring Signatures for Anonymizing Senders

Ring Signatures ensure sender anonymity in decentralized payment systems by proving a valid signer exists within a group, without revealing their identity, enabling untraceable transactions [5].

Log Analysis and SIEM Tools

Splunk, ELK Stack

- These are widely used for log analysis and management [14].
- These tools helps detect patterns, identify attack paths, and pinpoint impact systems.



Limitation

- SIEM tools require structure input data to operate effectively.
- They lack the ability to provide narrative or descriptive representation of cyber-attack.

Issue with the current practice

Why Are They Not Adequate?

- Description languages (e.g., STIX) can be too technical and difficult to use.
- They may lack the necessary depth or context to fully describe the threat, such as motivation, intent, or chain of attack.
- Logs might share irrelevant or excessive data, making it harder to prioritize and respond to threats effectively.

What Is the Problem?

- While description languages provide a framework, they are not always consistent across platforms or organizations.
- Manually entered data can lead to inaccuracies or inconsistent threat descriptions.
- Sharing mechanisms may not seamlessly integrate with tools like SIEM or IDS.
- They describe technical aspects (e.g., IoCs, malware signatures) well. However, they might miss the behavioral or strategic context of the threat, such as the attacker's intent or target priorities.

III. Proposed Solution

1. Cyber Threats
2. Problems/Challenges
3. Current Industrial Solution
4. Related Research
5. Proposed Solution

Proposed Solution

Cyber Attack Description Language

- Utilize CADL (Cyber Attack Description Language) for clear, human-readable threat descriptions

Zero-Knowledge Proof

- Anonymous CTI sharing platform built on blockchain and Zero-Knowledge Proof (ZKP) technologies

Blockchain/IPFS

- Ensure the transparency and immutability of shared cyber threat intelligence [13].
- Integrate IPFS for decentralized storage

ZK-R-SNARK

- Introduce ZK-R-SNARK: Instead of performing complex aggregations, leverage ring signatures to ensure anonymity and streamline verification, thus reducing computational overhead.

Proposed approach 1: Use of new threat definition language

Cyber Attack Description Language

What is CADL?

- A structured, human-read-friendly language designed for describing cyber attack.
- Combine the strengths existing CTI standard (e.g., STIX) and SIEM tools.
- Simplifies complex threat intelligence into easy-to-read, well-organized format.

Why CADL?

- Convert log file from major SIEM tools and JSON files from STIX into a unified, user-friendly format.
- Enhances understanding for analysts, enabling faster and more effective response.

Structure of CADL

Header				
Report Id			Data Analyst	Date/Time
Attack Details				
Name	Type	Severity Level	Discovered by	Affected System
Behavioral Description				
System			Network	
Sysmon Data				
Event Id	Process	File	Registry	Network
Suricata Alerts				
Alert Id	Signature		IP	Timestamp
Technical Details				
Hash			Code	Trigger
Detection and Mitigation				
Method				
Additional Finding				
Related Incident			Indication of Compromise	

Proposed approach 2: Use of Blockchain

Distributed Ledger

- Data is replicated across nodes globally instead of centralized servers.

Immutability

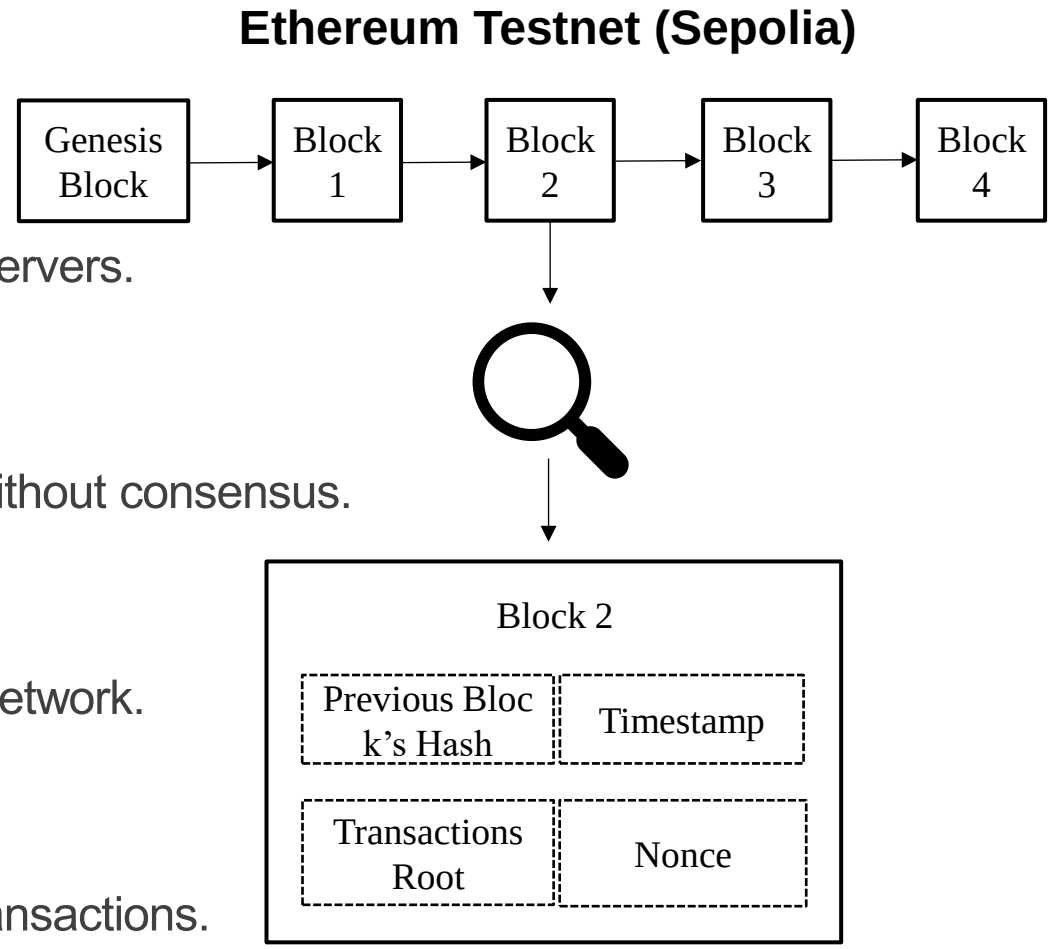
- Once a transaction is recorded in a block, it cannot be altered without consensus.

Transparency

- Transactions are visible and verifiable by all participants in the network.

Security

- Cryptographic techniques ensure the integrity and security of transactions.



Proposed approach 3: Use of ZKP for solving anonymity

Need for Anonymity

- Business and individuals require privacy to protect sensitive information, prevent unwanted surveillance.
- User data must be shielded from public scrutiny to ensure safety, compliance, and trust.

Blockchain Alone Doesn't Guarantee Anonymity

- Blockchain provides transparency and immutability, but all transaction details are visible on the ledger.
- Observers can link addresses and transaction patterns over time, potentially deanonymizing participants.

Why Zero-Knowledge Proofs?

- ZKPs allow the verification of transaction validity or compliance with certain conditions—without exposing the underlying details.
- By integrating ZKPs, participants can hide who they are and what they're doing, while still proving adherence to the system's rules.
- This balances the need for trust and verifiability with the equally important requirement for privacy and confidentiality.

Proposed approach 3: Use of ZKP for solving anonymity

Key Terminology and Concepts

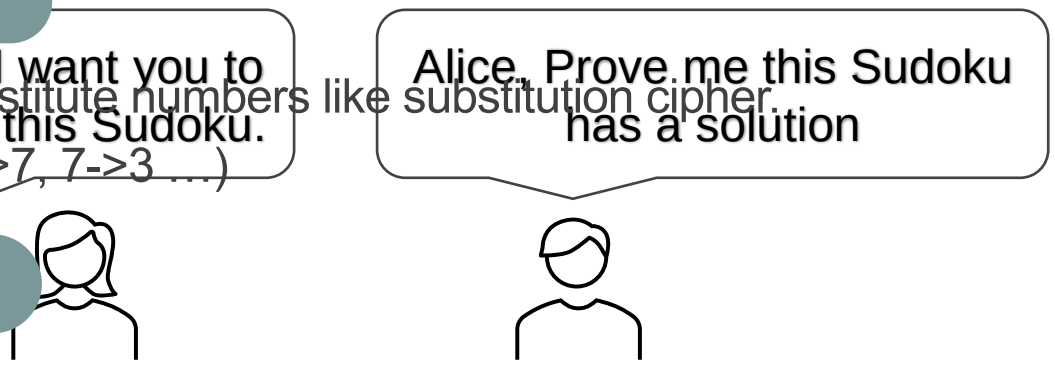
- ZKP
  - ZKP is a cryptographic protocol that allows the prover to prove the verifier that they possess specific knowledge or meet certain conditions, without revealing any underlying information about the knowledge itself [6].
- ZK-SNARK
  - Ensures the validity of Ring Signature proofs while maintaining privacy.
- Ring Signature
  - Provides anonymity by ensuring that a message is signed by one participant in a group without revealing which one. Enables anonymous and secure CTI sharing [8].
- Circuit
  - Serves as the mathematical foundation to implement Ring Signature operations, such as proving group membership and message signing within the Ring.
- ZK-R-SNARK (Newly Proposed)
  - ZK-R-SNARK is a newly proposed framework that combines the principles of ZK-SNARKs and Ring Signatures to provide enhanced anonymity and efficiency for secure and anonymous operations.

III. Application of Zero-Knowledge Proof method

1. Zero Knowledge Proof
2. Ring Signature
3. ZK-R-SNARK

Scenario: Sudoku Puzzle

Step 1



- Alice substitute numbers like substitution cipher.
  (1->9, 2->7, 7->3 …)

Step 2

- Alice hide solution, and ready to get question from Bob
- Hided solution is presented to Bob.

Alice want to make Bob believe..

Step 3

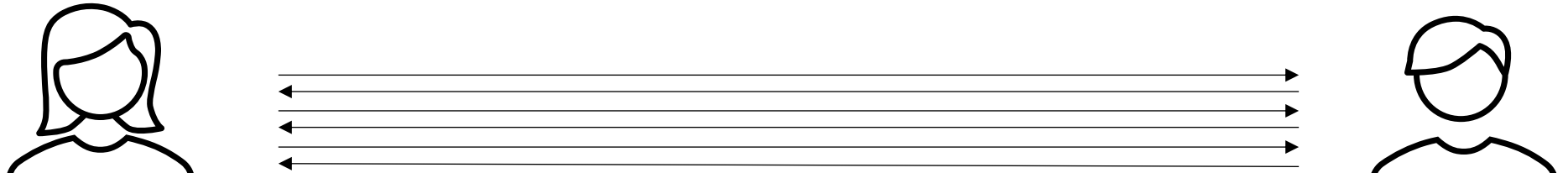
- Bob randomly pick a unit(row, column, 3x3 square), and ask Alice to show the solution of unit. Bob verifies there are number 1-9.
- Alice permute solution again and do step2 to step3.

Bob should not learn "too much"

- The sudoku puzzle has a solution.

			9	7			8	4
			2	8	4	3	1	5
						7	9	2
			7			9	5	3
	2			3	6	4	7	8
					9	2	6	1
3	8		6	2				9
	4					1	2	7
2		9		5				6

How To Convince Bob?



Alice

Bob

- Repeat the step2 to step3 n times, if answer that Alice shows always contains 1-9, Bob knows that the chance she achieve this by luck without valid solution is most at (27/28)n.
- For n=100 the chance is ~2%,
- For n=200 the chance is ~0.06%

ZK-SNARK Core Concepts

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge

- ZK-SNARK provides a way to prove that a statement is true without revealing any underlying secrets [7].

- This set of slides will introduce the foundational elements of ZK-SNARKs, guiding you through the core building blocks and the overall proof pipeline.

- Note: For deeper explanations and mathematical details, please refer paper or backup slides.

ZK-SNARK Algorithm flow

- Circuit → R1CS → QAP → CRS → Witness → Proof Generation → Proof Verification

ZK-SNARK Circuit

Circuit

- ZK-SNARK proof begins with a computational problem expressed as a circuit, consisting of a series of gates and wires.
- The circuit represents the logic and constraints of the statement you wish to prove—such as arithmetic operations, comparisons, and branching decisions—under a well-defined structure.
- By translating your problem into a circuit, you create a clear framework for verifying correct computation without revealing secret inputs.

ZK-SNARK Algorithm Flow

Circuit → R1CS

- Transforming a logical circuit into arithmetic constraints.

R1CS → QAP

- Converting matrix constraints into polynomials.

QAP → CRS

- Generating a common reference string (CRS) through a trusted setup

CRS → Witness → Proof

- The prover generates a proof using the witness and CRS.

Proof → Verification

- The verifier efficiently checks the proof's validity

Overview of Ring Signature

Ring Signature

- A Ring Signature is a cryptographic method that allows a member of a group to sign a message on behalf of the group while ensuring anonymity. The signature ensures that it is computationally infeasible to determine which member of the group signed the message, providing both privacy and authenticity.

- The verifier can confirm that the signature was produced by one of the group members but cannot identify exactly which one.

Circuit Structure

- Template Parameters:
  - n: Number of participants (ring members)
  - P: A prime number used for modular arithmetic
- Signals Defined:
  - pks[n]: Public keys of the ring members
  - c[n]: Array of challenges
  - s[n]: Array of responses

Example of Ring Signature Circuit

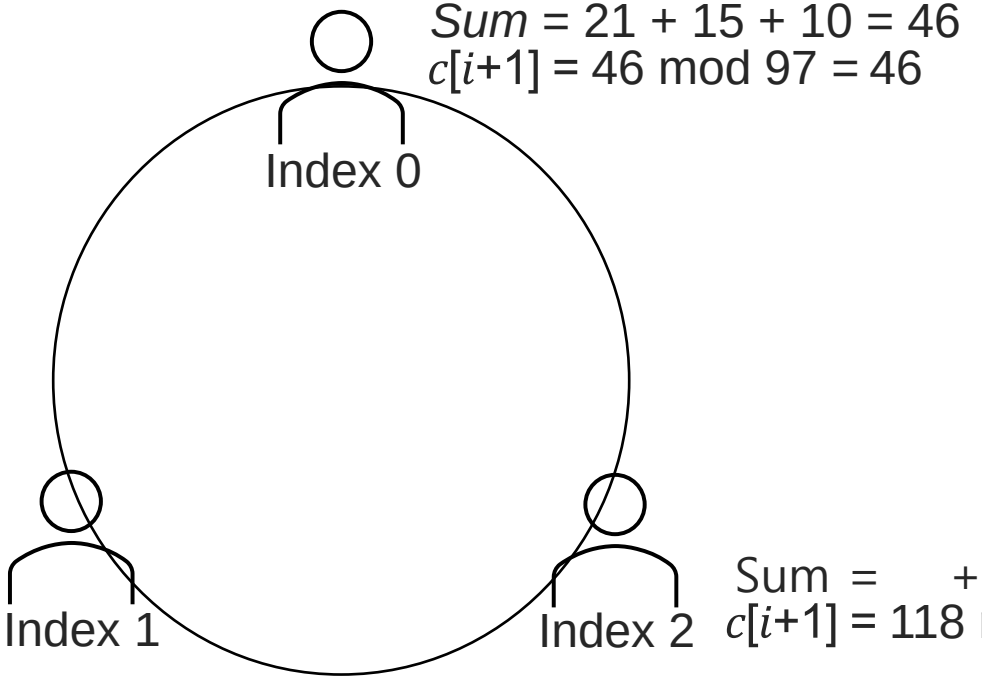
Example input Value and Computation

To enforce a cyclic (ring) relationship through modular arithmetic.

The last challenge is linked back to the first, ensuring the “ring” structure:

$$c[i+1]=(c[i]+pks[i]+s[i]) \bmod P$$

- **i = [0, 1, 2]**
- c[i] = [21,46, 19]
- **pks[i] = [15,30,10]**
- **s[i] = [10, 40 , 89]**
- P = 97

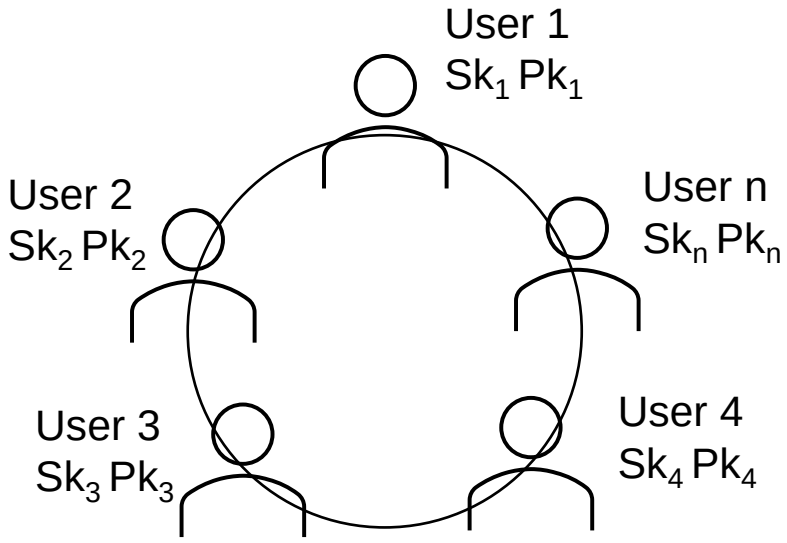


Integrating Ring Signature into ZK-SNARK Witness

Witness with Ring Signature

- **The ZK-SNARK witness now includes Ring Signature components:**
  - Secret key (SK): The private key corresponding to one of the group's public keys.
  - Index (idx): The position of the signer in the group.
  - Ring Signature values: $\{s_i\}$,c_0,Key Image, and intermediate values.

- **These elements allow the circuit to verify:**
  - The signature is valid for the group.
  - The signer remains anonymous among the group members.
  - Double-spending is prevented using Nullifier (Key Image hash).



Constructing the Witness with Ring Signature

Ring Signature Validation in the Circuit

- **Key Image Generation**: The circuit ensures the Key Image is correctly generated using the secret key: Key Image=$H(SK \cdot G)$

- **Nullifier Generation**: The Nullifier is derived from the Key Image to prevent secret key reuse: Nullifier=H(Key Image)

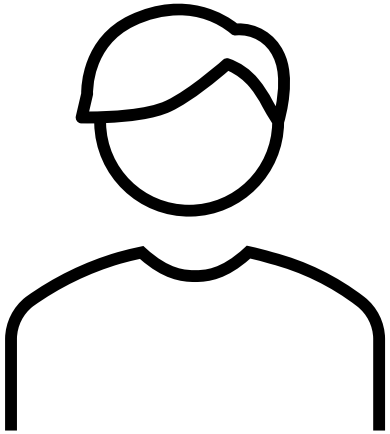
- **Ring Signature Verification**: The circuit verifies the Ring Signature chain by checking the values:c_i+1=$H(g^s{}_i \cdot \text{pk}_i{}^c)$
  - The witness provides s_i and c_0 for each step.
  - Preventing Double-Spending: The circuit compares the Nullifier with previously used Nullifiers to confirm:
  - Nullifier ∉ Used Nullifier List

Application of ZK-R-SNARK to CTI Sharing

1. Create Random Values, where $r \in z_p$ $R = g^r$
2. Create Ring Signature, where

$$c_i = H(D, P_i, R) \oplus g^{sk}_i$$

$$\sigma = (R, c_1, ..., c_n)$$

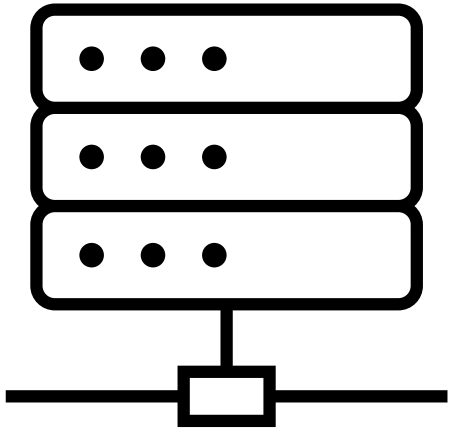


Prover

Send Proof = (σ, D)

$c_i = ?H(D, P_i, R) \oplus g^{sk}_i$

1. $\forall P_i \in \{P_1, ..., P_n\}$,
2. $c_i == H(D, P_i, R) \oplus g^{sk}_i$
3. Return Valid or Not



Validator

ZK-R-SNARK

Benefits of Integrating Ring Signature

- **Anonymity**: The circuit validates that the signer is part of the group without revealing the specific signer.
- **Double-Spending Prevention**: The Nullifier ensures that the same private key cannot be used multiple times [15].
- **Efficiency**: Ring Signature verification is embedded within the ZK-SNARK circuit, enabling compressed proofs and efficient verification on the blockchain.

Key Differentiator

- Unlike prior methods that expose contributors to identity inference risks, our solution integrates advanced cryptographic techniques (ZKP + Ring Signatures) to offer:
  - **Strong Data Credibility**: CTI entries are verifiable and reliable.
  - **Formal Anonymity Guarantee**: Contributor identities remain private, even during extended blockchain usage.

ZK-R-SNARK

Aspect	General Witness Generator	Proposed Witness generator
Purpose	Simple condition verification	Anonymity, Ring signature Validation, Nullifier
Private input	Single secret value (x)	Secret Key (SK) and index (idx)
Intermediate Values	None	P, s_i, c_0
Computation Complexity	Low	High
Anonymity	none	Hide singer identity among a group

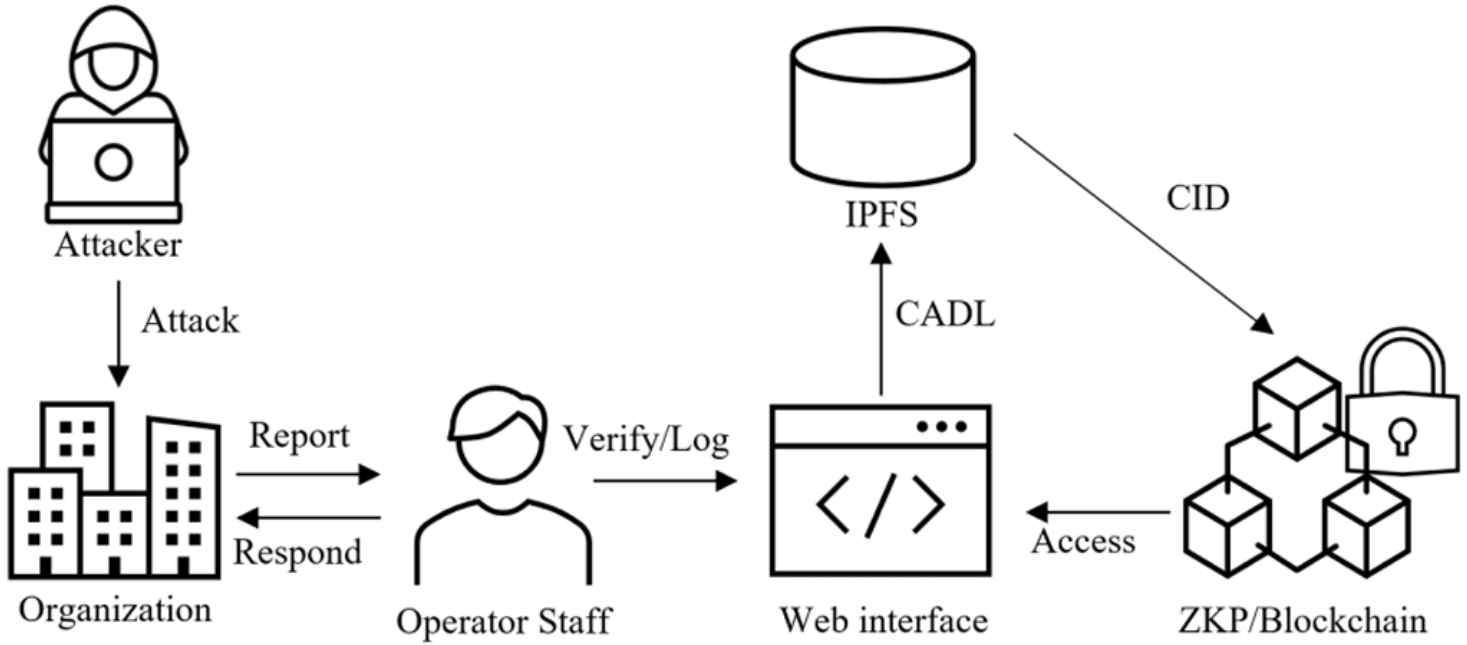
IV. Implementation

1. Overview
2. Circom
3. Smart Contract
4. Web Interface

Implementation Overview

Purpose of Implementation

- To build and validate a working prototype of the proposed anonymous CTI sharing model, demonstrating that the theoretical framework can operate in practice. It aims to prove that organizations can share threat intelligence on blockchain with privacy preserved (using ZKP) and no central authority, thus increasing trust and participation



Circom (ZKP Implementation)

Why Circom was Chosen

- **Circom is a high-level circuit definition language tailored for creating arithmetic circuits for zk-SNARKs. It allows the implementation of complex cryptographic logic (like our anonymous credential proofs) in a concise way.**

- **Key advantages:**
  - Has a strong ecosystem and integrates with SnarkJS, a JavaScript library, enabling end-to-end workflow from circuit design to proof generation and verification. This made it convenient to generate the proofs and the corresponding Solidity verifier contract for Ethereum.
  - Circom's compatibility with Ethereum (via SnarkJS output) and its efficiency in generating succinct R1CS (Rank-1 Constraint System) representations were ideal for our use-case, where on-chain verification and performance are important considerations.

Smart Contract Development

Role of S

- The smart co
  - on the blocko
  - control policie
- This ensures
  - to blockchain

Smart C

- zk-SNARK V
  - We integ
    - our cont
    - SNARK
  - Under th
    - performs
    - a submis
    - verificati
    - details, thanks to the ZKP.

```
function verifyProof(uint[2] calldata _pA, uint[2][2] calldata _pB, uint[2] calldata _pC, uint[<%=IC.length-1%>] calldata _pubSignals) public view returns (bool) {
    assembly {
        function checkField(v) {
            if iszero(lt(v, q)) {
                mstore(0, 0)
                return(0, 0x20)
            }
        }

        // G1 function to multiply a G1 value(x,y) to value in an address
        function g1_mulAccC(pR, x, y, s) {
            let success
            let mIn := mload(0x40)
            mstore(mIn, x)
            mstore(add(mIn, 32), y)
            mstore(add(mIn, 64), s)

            success := staticcall(sub(gas(), 2000), 7, mIn, 96, mIn, 64)

            if iszero(success) {
                mstore(0, 0)
                return(0, 0x20)
            }

            mstore(add(mIn, 64), mload(pR))
            mstore(add(mIn, 96), mload(add(pR, 32)))

            success := staticcall(sub(gas(), 2000), 6, mIn, 128, pR, 64)

            if iszero(success) {
                mstore(0, 0)
                return(0, 0x20)
            }
        }

        function checkPairing(pA, pB, pC, pubSignals, pMem) -> isOk {
            let _pPairing := add(pMem, pPairing)
            let _pVk := add(pMem, pVk)

            mstore(_pVk, IC0x)
            mstore(add(_pVk, 32), IC0y)
```

Web Interface

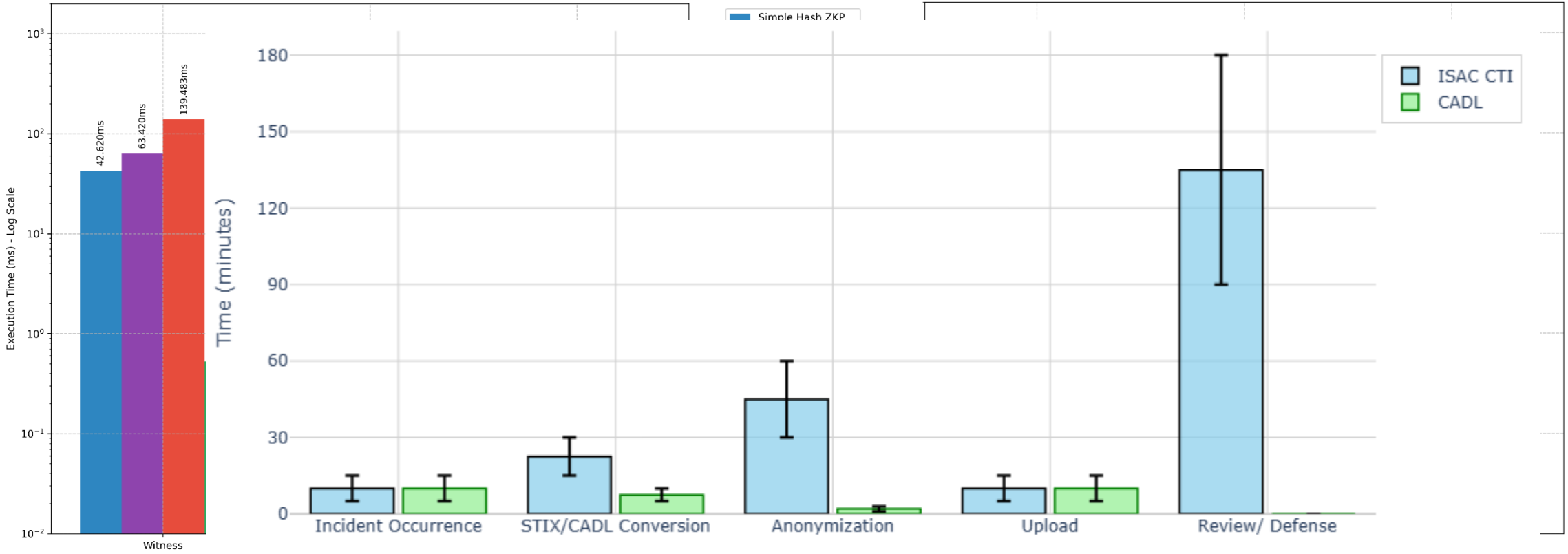
Front-End

- We developed a web-based interface to make the system accessible to end-users (analysts or cybersecurity professionals in member organizations). This interface is essentially a dApp (decentralized application) that communicates with the blockchain and IPFS in the background.
  - **Dashboard** showing the latest shared threat intelligence entries (with metadata like timestamp, threat type, etc.), which are fetched by reading blockchain events and then retrieving the actual report content from IPFS using the CID.
  - **Submission form** allowing authorized users to input a new CTI report (or select a prepared report file). When the user submits, the front-end code triggers the Circom proof generation process (this could be done client-side or via a back-end service; in our implementation, we experimented with generating proofs in the browser for small circuits). After obtaining the proof and CID (after uploading the report to IPFS), the interface uses Web3 functionalities to send the data to the smart contract.
  - **Status notifications** are provided (e.g., “Proof generated successfully”, “Transaction submitted to blockchain, awaiting confirmation...”), since blockchain transactions aren’t instant. Once the transaction is confirmed, the new CTI entry appears on all users’ dashboards.

V. Result

1. Performance Comparison by Programming Language
2. Task-wise Time Comparison
3. Demo

Performance Comparison

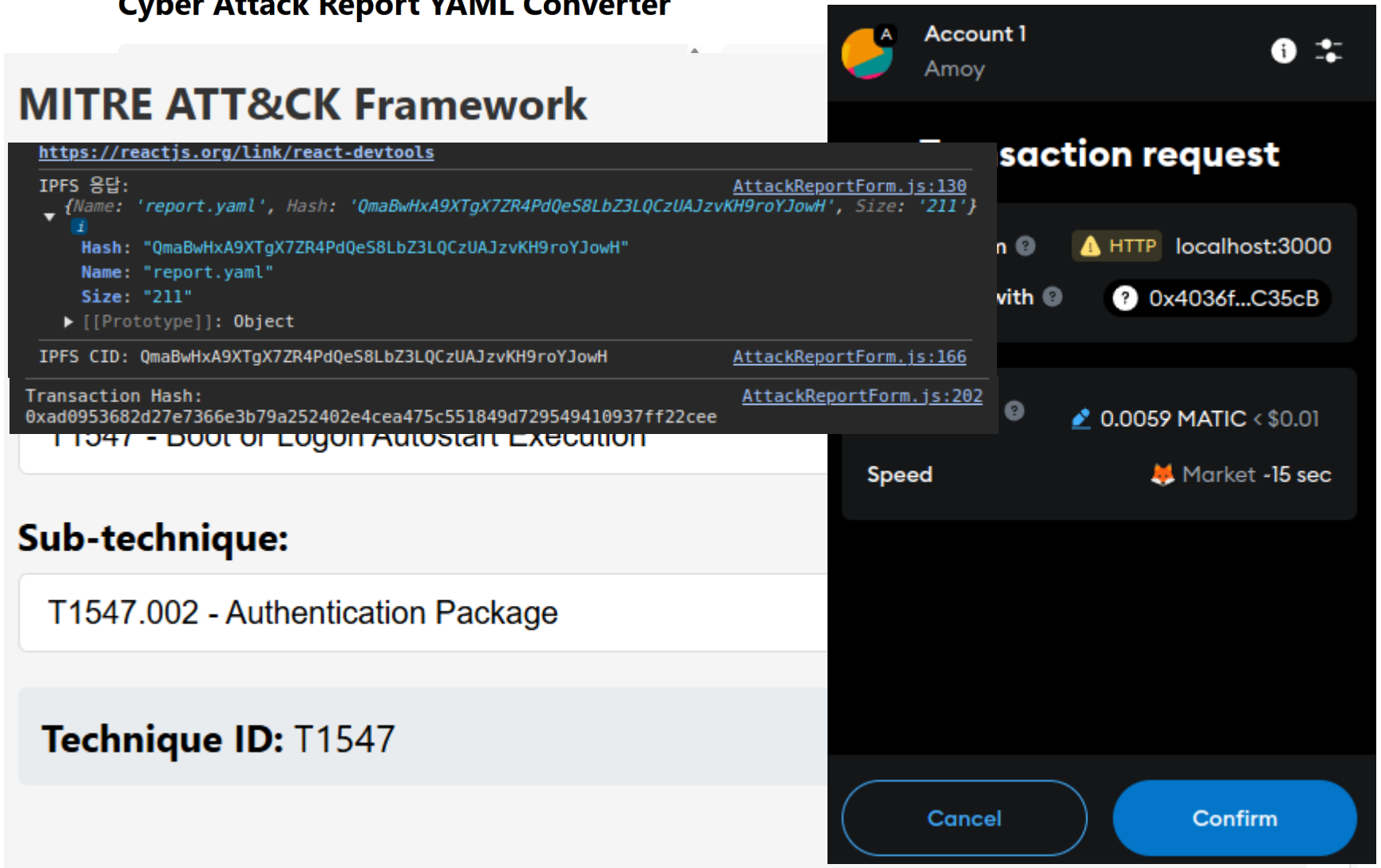


JavaScript

C++

Demo

Cyber Attack Report YAML Converter



VI. Discussion

1. Security Analysis
2. Limitation

Security Analysis: Enhancing CTI Sharing Security

- **Group-based Anonymity:**
  - Cryptographically signs CTI submissions by an unspecified ring member.
  - Breaks direct address linkability; adversaries can't correlate multiple submissions.
- **Metadata Protection:**
  - Reduces exposure of transaction metadata, hindering machine-learning based forensics.
- **Dynamic Access Control:**
  - Only authorized group members can produce a valid ZKP without revealing identity.
- **Resistance to Sybil Attacks:**
  - Predefined, cryptographically validated group membership prevents mass fraudulent submissions.
- **Enhanced Censorship Resistance:**
  - Decentralized storage (IPFS + blockchain references) secures data against selective suppression.

Limitations – Governance & Trust Management

Security Analysis: Enhancing CTI Sharing Security

- **Dynamic Membership Challenges:**
  - Static nature of ring membership makes it hard to update or manage group changes.

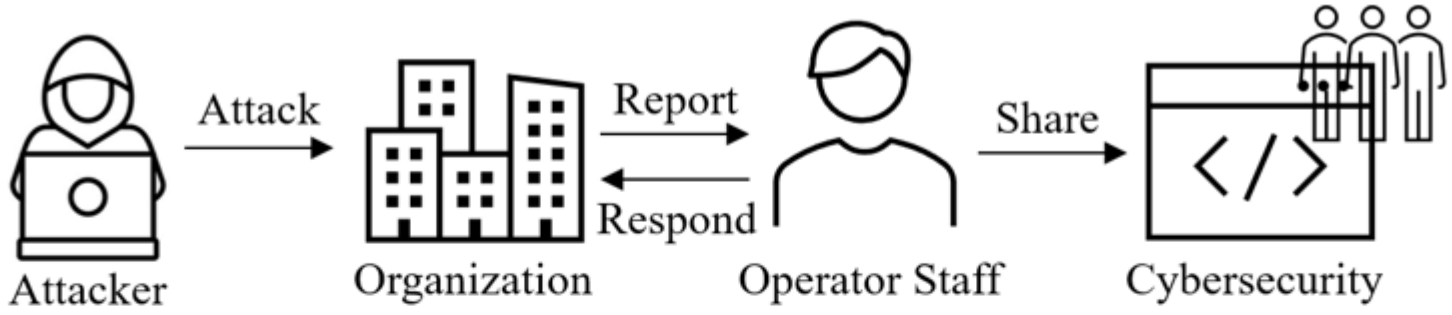
- **Quality Assurance:**
  - Decentralized model lacks straightforward methods to filter low-quality or false reports.
  - Risks include false, outdated, or misleading intelligence without central oversight.

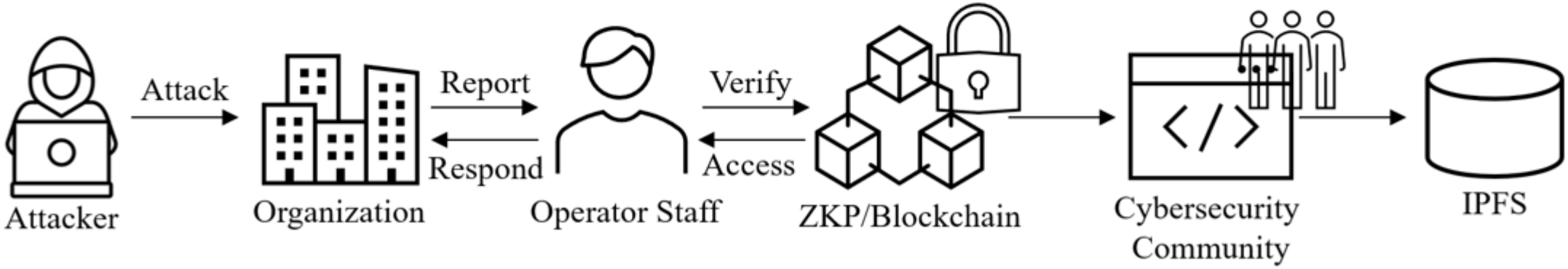
- **Economic Incentives:**
  - Traditional systems use identifiable addresses for reputation and rewards.Anonymity makes designing effective incentive models more complex.

VII. Conclusion and Future Work

1. Conclusion
2. Future Work

Putting It All Together





- ZKP ensures data validity, while ring signatures maintain contributor anonymity, even if the public key is exposed.

What Makes This Research Unique

First to Address CTI Sharing with Cryptographic Guarantees

- Our research combines zk-SNARK and Ring Signature technologies to ensure anonymity and trust in CTI sharing.

Focus on Anonymity and Scalability

- This research specifically addresses the barriers of privacy and scalability that have hindered academic exploration in this field.

A Unique Direction

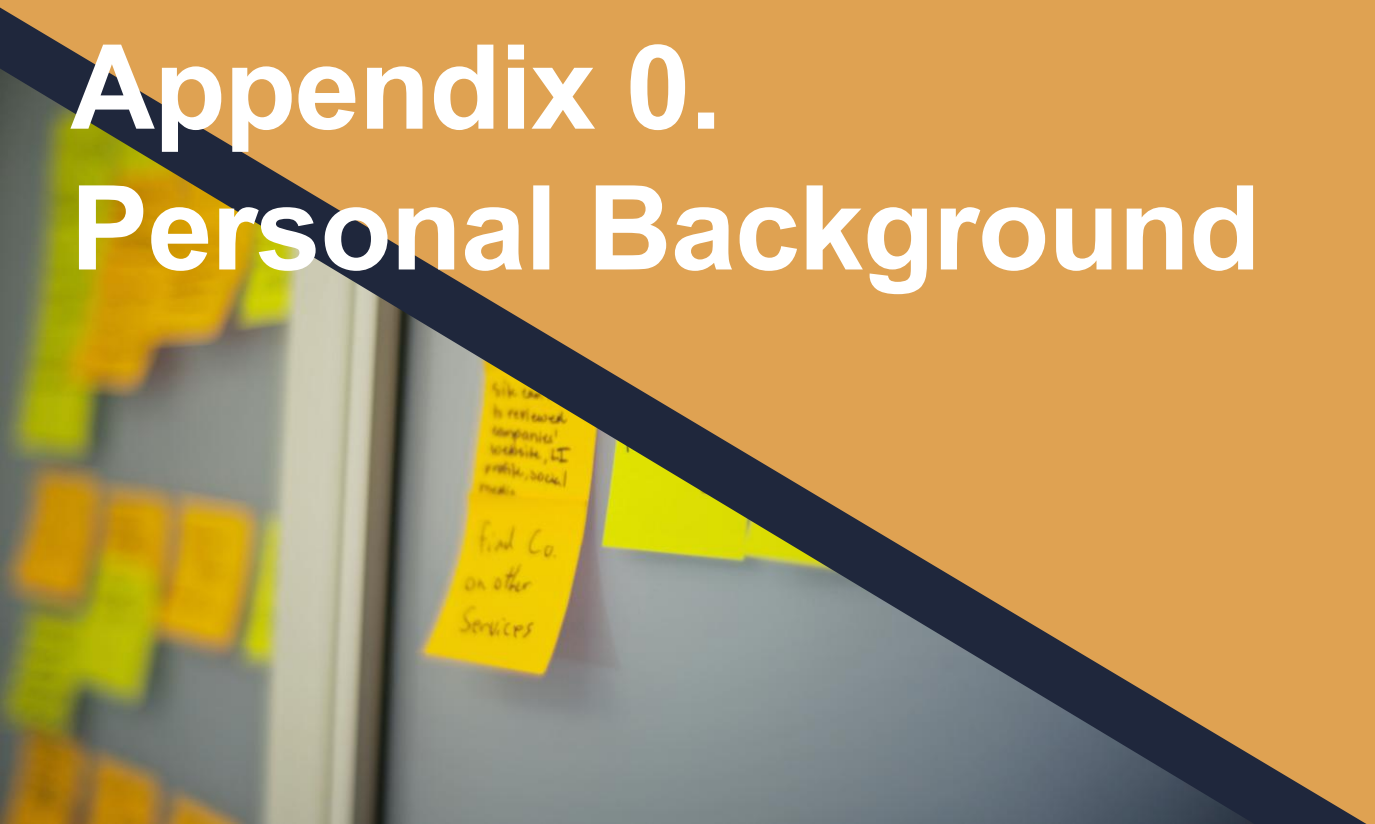
- While prior work focuses on cryptographic solutions, this is one of the first studies directly applying these technologies to solve real-world CTI sharing challenges.

References

1. E. M. Abdelmaboud, S. B. [illegible], "Authentication Scheme Based on Non-Interactive Zero-Knowledge Proof for Mobile Health," 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), Hainan, China, 2022, pp. 1690-1696, doi: 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00256. [illegible] non-interactive zero-knowledge proof;mHealth;lightweight devices},
2. D. Gabay, K. Akkaya and M. Cebe, "Privacy-Preserving Authentication Scheme for Connected Electric Vehicles Using Blockchain and Zero Knowledge Proofs," in *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5760-5772, June 2020, doi: 10.1109/TVT.2020.2977361.
3. Z. Zhang et al., "Research on the security sharing method of power grid dispatching control data based on Alliance Blockchain and Zero-Knowledge Proof technology," *2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Chongqing, China, 2022, pp. 1645-1649, doi: 10.1109/IMCEC55388.2022.10019813.
4. E. Ben-Sasson et al., "Zerocash: Decentralized Anonymous Payments from Bitcoin," 2014 IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 2014, pp. 459-474, doi: 10.1109/SP.2014.36.
5. F. Pitu and N. C. Gaitan, "Survey of security, performance, and profitability of Monero: a browser-based cryptocurrency," *2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, Tenerife, Canary Islands, Spain, 2023, pp. 1-6, doi: 10.1109/ICECCME57830.2023.10253316.
6. Goldwasser, Shafi, et al. "The Knowledge Complexity of Interactive Proof Systems." SIAM Journal on Computing, vol. 18, no. 1, Feb. 1989, pp. 186–208, https://doi.org/10.1137/0218012.
7. E. Ben-Sasson, A. Chiesa, E. Tromer and M. Virza, "Succinct non-interactive zero knowledge for a von Neumann architecture", Proc. 23rd USENIX Secur. Symp., pp. 781-796, 2014.
8. X. Li, Y. Mei, J. Gong, F. Xiang and Z. Sun, "A Blockchain Privacy Protection Scheme Based on Ring Signature," in *IEEE Access*, vol. 8, pp. 76765-76772, 2020, doi: 10.1109/ACCESS.2020.2987831
9. J. Lu, R. Qi and J. Shen, "A Novel Dynamic Group Signature with Membership Privacy," *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, Aizuwakamatsu, Fukushima, Japan, 2021, pp. 1-5, doi: 10.1109/DSC49826.2021.9346238.
10. J. Zhang, Y. Zhao, J. Wu and B. Chen, "LVPDA: A Lightweight and Verifiable Privacy-Preserving Data Aggregation Scheme for Edge-Enabled IoT," in IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4016-4027, May 2020, doi: 10.1109/JIOT.2020.2978286.
11. V. Mavroeidis and S. Bromander, "Cyber Threat Intelligence Model: An Evaluation of Taxonomies, Sharing Standards, and Ontologies within Cyber Threat Intelligence," 2017 European Intelligence and Security Informatics Conference (EISIC), Athens, Greece, 2017, pp. 91-98, doi: 10.1109/EISIC.2017.20.
12. Ernest F. Brickell, Jan Camenisch, Liqun Chen, "Direct anonymous attestation," pp. 132-145, 2004.
13. Nakamoto, Satoshi. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. Cryptography Mailing list at https://metzdowd.com.
14. A. [illegible], "Threat Modeling and Threat Intelligence System for Cloud using Splunk," *2022 10th International Symposium on Digital Forensics and Security (ISDFS)*, Istanbul, Turkey, 2022, pp. 1-6.
15. H. Wang, F. Zhang, and [illegible], "Cryptanalysis of a Generalized Ring Signature Scheme," IEEE Transactions on Dependable and Secure Computing, vol. 6, no. 2, pp. 149-151, 2009.
16. [illegible], "[illegible] deployment solution based on the peer-to-peer protocol," Applied Sciences (Multidisciplinary Digital Publishing Institute), vol. 9, no. 2, p. 296, 2019.
17. [illegible], "Research on threat intelligence detection model based on deep neural network," [illegible] vol. 12642, [illegible] 2023.
18. [illegible], "[illegible] Federated Graph Neural Network for Threat Intelligence," [illegible]
19. Edgardo Montes de Oca, Jart Armin, Angelo Consoli, "Cyber-Threat Intelligence from European-wide Sensor Network in SISSDEN," Challenges in Cyber Security and Privacy - the European Research Landscape, pp. 117-128, 2022.

Q&A

Appendix 0.
Personal Background



Chol Hyun Park

Education

- Bachelor's Degree in Computer Science, Westmont College 2018
- Ph.D. Candidate in Computer Science, UNLV 2018~Current
- Research Interest: Cybersecurity, Blockchain

Experience

- Casual Data Scientist – MSTS LLC (2023-2024)
  - Applied deep learning techniques detect patterns in seismic datasets.

Key Publication

- A Market Place Solution for Energy Transaction on Ethereum Blockchain, IEMCOM, 2019
- Detecting Cyber Threats with Limited Dataset Using GAN on SCADA System, CSCI, 2023
- A secure communication method for CANBus, CCWC, 2021

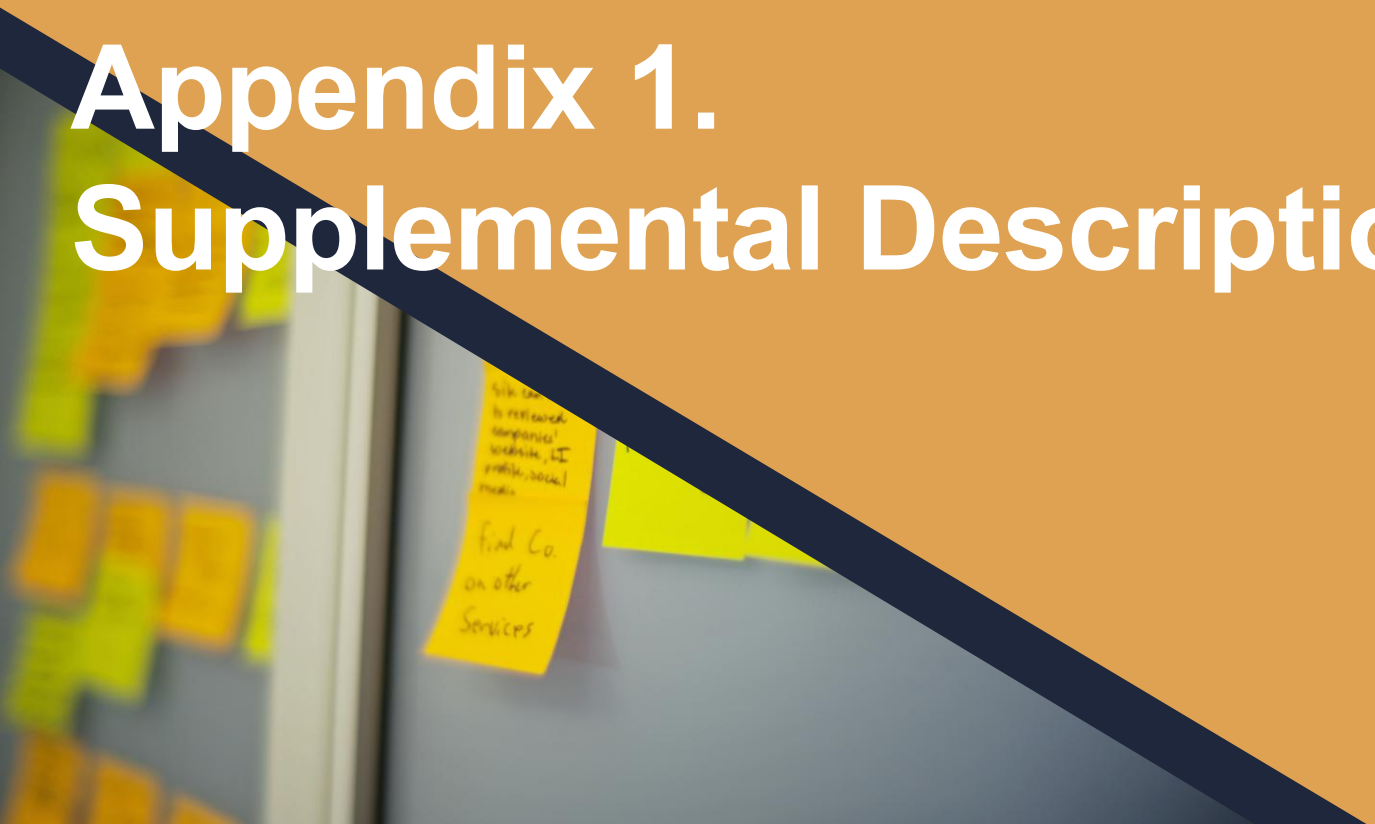
https://chpark0714.github.io/

Publications

Publications

1. C. H. Park, I. Mejia Barlongo and Y. Kim, "A Market Place Solution for Energy Transaction on Ethereum Blockchain," 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 2019
2. C. H. Park, Y.Kim, J. Jo, “A Secure Communication for CANBus” 2021 IEEE 11th Computing And Communications Workshop and Conference(CCWC), Virtual, 2021 3. B.
3. Hoffman, Alex, P. Austria, C. H. Park, Y. Kim. “Bountychain: Toward Decentralizing a Bug Bounty Program with Blockchain and Ipfs.” International Journal of Networked and Distributed Computing, vol. 9, no. 2-3, 2021, p. 86.,
4. B. Cisneros, J. Ye, C. H. Park and Y. Kim, "CoviReader: Using IOTA and QR Code Technology to Control Epidemic Diseases across the US," 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), NV, USA, 2021
5. P. Austria, C. H. Park, A. Hoffman and Y. Kim, "Performance and Cost Analysis of Sia, a Blockchain-Based Storage Platform," 2021 IEEE/ACIS 6th International Conference on Big Data, Cloud Computing, and Data Science (BCD), Zhuhai, China, 2021
6. C. H. Park, G Chmaj, H Selvaraj, “Blockchain-Based Smart Contracts Use for Photovoltaic Energy Trade Transactions.” Advances in Systems Engineering, 2021
7. P. Austria, C. H. Park, J.-Y. Jo, Y. Kim, R. Sundaresan and K. Pham, "BBR Congestion Control Analysis with Multipath TCP (MPTCP) and Asymmetrical Latency Subflow," 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2022, pp. 1065-1069, doi: 10.1109/CCWC54503.2022.9720867.
8. C. H. Park, P. Austria, Y. Kim and J.-Y. Jo, "MPTCP Performance Simulation in Multiple LEO Satellite Environment," 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA, 2022,
9. C. H. Park, JY Jo, Y Kim, “Detecting Cyber Threats with Limited Dataset Using Generative Adversarial Network on SCADA System” 2023 International Conference on Computational Science and Computational Intelligence (CSCI)
10. M Buslon, C.H. Park, Y Kim, JY Jo, “Attack Target Detection Using Machine Learning on SCADA GasPipeline Data” 2023 International Conference on Computational Science and Computational Intelligence (CSCI)

Appendix 1.
Supplemental Description of ZKP process



Zero Knowledge

Alice want to make Bob believe..

- The Sudoku puzzle has a solution.

Bob should not learn “too much”

- The sudoku solution.

Most Simple Form of ZKP

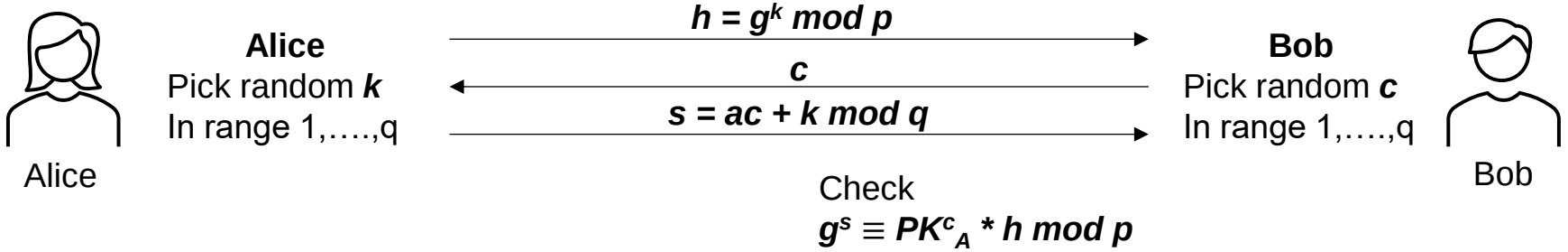
Prove Alice knows a Secret Key

- A secret key in the world of elliptic curve cryptography is a large random number.
- Let the secret key be *a*.
- The corresponding public key is $PK = g^a \bmod P$
- Math recall

$$g^{(ac)}=(g^a)^c = (PK)^c$$

$$g^{(m+k)}=g^m * g^k$$

- Given $g^a \bmod P$ is computationally infeasible to find *a* if *a* is large enough.
- Using same idea as how Alice prove Bob she shows the Sudoku solution, she ask Bob to pick a random large number *c*. She can prove to Bob she knows secret *a*, without telling Bob what a is, by presenting *s=ac+k*



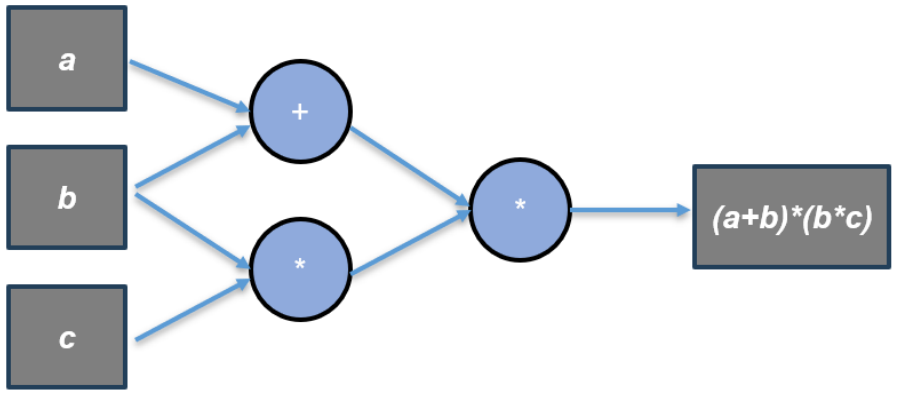
ZK-SNARK Circuit

Circuit

- ZK-SNARK proof begins with a computational problem expressed as a circuit, consisting of a series of gates and wires.

- The circuit represents the logic and constraints of the statement you wish to prove—such as arithmetic operations, comparisons, and branching decisions—under a well-defined structure.

- By translating your problem into a circuit, you create a clear framework for verifying correct computation without revealing secret inputs.



ZK-SNARK R1CS/QAP

R1CS

- R1CS transforms the circuit into a set of polynomial equations, ensuring that the computation's correctness can be captured as linear constraints over variables.
- Each constraint is in the form of a rank-1 equation:$(A \cdot z) \circ (B \cdot z) = (C \cdot z)$ where z is a vector of all variables (inputs, outputs, and intermediates).

QAP

- QAP encodes the R1CS system into a set of polynomials, ensuring that satisfying the constraints corresponds to a polynomial identity holding true at a secret evaluation point.
- Three sets of polynomials ($A(X)$, $B(X)$, $C(X)$) represent the constraints.
- A valid witness ensures that $A(X)*B(X) - C(X) = 0$ at a specific, hidden point $X = s$.

ZK-SNARK Quadratic Arithmetic Program

QAP

- QAP encodes the R1CS system into a set of polynomials, ensuring that satisfying the constraints corresponds to a polynomial identity holding true at a secret evaluation point.

- Three sets of polynomials ($A(X)$, $B(X)$, $C(X)$) represent the constraints.

- A valid witness ensures that $A(X)*B(X) - C(X) = 0$ at a specific, hidden point $X = s$.

- Converting R1CS to QAP enables the use of pairing-based cryptography and the creation of a succinct proof that can be efficiently verified.

ZK-SNARK Common Reference String / Witness

CRS

- The Common Reference String (CRS) is a set of cryptographic parameters generated once during a trusted setup phase.
- The CRS encodes information derived from the QAP and a secret evaluation point.
- It allows the prover to create and the verifier to check proofs without revealing secret values.

Witness

- The witness is the private data (secret inputs and intermediate computations) that the prover knows and uses to demonstrate that the constraints are satisfied.
- The witness provides the internal evidence that the prover performed the computation correctly.
- It never needs to be revealed to the verifier.

ZK-SNARK Proof

Generation

- Using the witness and the CRS, the prover constructs a short proof that asserts the correctness of the computation.
- The witness is used to evaluate polynomials at the secret point.
- The resulting values are combined with the CRS parameters to produce a succinct proof.

Verification

- The verifier uses the verifying key (derived from the CRS) and the public inputs to check the proof efficiently.
- The verifier performs a few pairing checks and polynomial evaluations.
- If the proof is valid, it confirms that a correct witness exists for the given statement.

ZK-SNARK

- SNARK(succinct non-interactive argument of knowledge) is one of the most popular ZKP schemes.

- Given function *f(x)*, and output *y*, using ZK-SNARK, one can generate a proof to demonstrate the knowledge of solution *s*, without showing the value of *s*.

- Given : $f(x) = y$
- Produce s, such as that $f(s) = y$

- SNARK consumes the “circuit” of the function *f(x)*, and public input *y* as input, and produce ZK proof as output.

Integrating Ring Signature into ZK-SNARK Witness

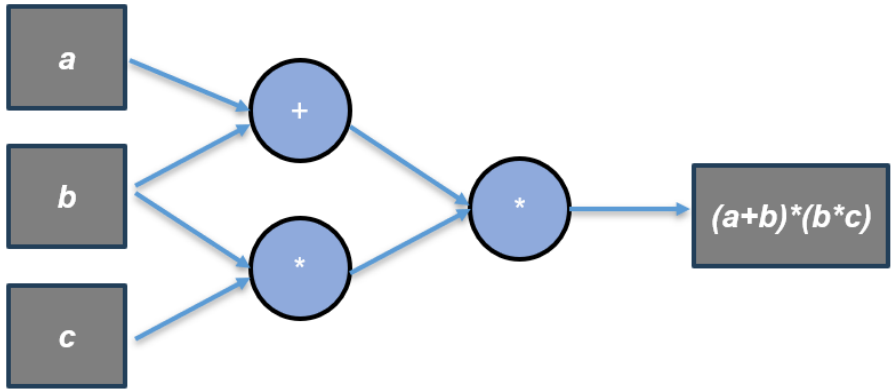
Why Include Ring Signature in the Witness?

- ZK-SNARK focuses on proof compression and verification efficiency but lacks native anonymity guarantees for group proofs.
- Problem: Without additional mechanisms, the identity of the signer or participant might still be inferred.
- Solution: Ring Signature ensures anonymity by proving "a valid signer exists within a group" without identifying which one.

Circuit

What is circuit?

- Circuit is used for model computations. Circuit is a network of gates with two input wires and one output wire.
- For example, *(a+b)*(b*c)*:



Program analysis

R1CS Representation of the Circuit

- Using Rank 1 Constraint System (R1CS), the computations can be captured in a collection of vectors.
- R1CS is a vector-based system that represents the sequence of gates in the circuit.
- The circuit that calculated $y=x3+2x+5$ can be broken down into gates:
  - *1) T1 = x * x*
  - *2) z = T1 * x*
  - *3) T2 = z + 2x*
  - *4) y = T2 + 5*

	(1,x,y,T1,z,T2)	**(1,x,y,T1,z,T2)**	**(1,x,y,T1,z,T2)**
	A	B	C
1)	[0,1,0,0,0,0]	[0,1,0,0,0,0]	[0,0,0,1,0,0]
2)	[0,0,0,1,0,0]	[0,1,0,0,0,0]	[0,0,0,0,1,0]
3)	[0,2,0,0,1,0]	[1,0,0,0,0,0]	[0,0,0,0,0,1]
4)	[5,0,0,0,0,1]	[1,0,0,0,0,0]	[0,0,1,0,0,0]

Program analysis

R1CS Representation of the Circuit

- The next step is building a “witness expression”, that captures all the variables involved in the computation gates.
- For the gates, involved in the above calculation, the witness expression is *w = (1,x,y,T1,z,T2)*
- The expression to represent the state of each of the input and output wire of each gate.
- A, B are the input wires.
- C is the output wire
  - *1) T1 = x * x*
  - *2) z = T1 * x*
  - *3) T2 = z + 2x*
  - *4) y = T2 + 5*

	(1,x,y,T1,z,T2)	**(1,x,y,T1,z,T2)**	**(1,x,y,T1,z,T2)**
	A	B	C
1)	[0,1,0,0,0,0]	[0,1,0,0,0,0]	[0,0,0,1,0,0]
2)	[0,0,0,1,0,0]	[0,1,0,0,0,0]	[0,0,0,0,1,0]
3)	[0,2,0,0,1,0]	[1,0,0,0,0,0]	[0,0,0,0,0,1]
4)	[5,0,0,0,0,1]	[1,0,0,0,0,0]	[0,0,1,0,0,0]

Verifying R1CS

- R1CS matrix can be verified using "$\circ$" operation with a witness:
  - $w \circ A * w \circ B = w \circ C$
- Using example: $P(x) = x^3+x+5$, $P(3)= 33+3+5 = 35$,
- Expressed in witness format: [1, 3, 35, 9, 27, 30]

[1, 3, 35, 9, 27, 30]	**[1, 3, 35, 9, 27, 30]**	**[1, 3, 35, 9, 27, 30]**	
A	B	C	
[0,1,0,0,0,0]	[0,1,0,0,0,0]	[0,0,0,1,0,0]	3*3=9
[0,0,0,1,0,0]	[0,1,0,0,0,0]	[0,0,0,0,1,0]	9*3=27
[0,2,0,0,1,0]	[1,0,0,0,0,0]	[0,0,0,0,0,1]	(3+27)*1=30
[5,0,0,0,0,1]	[1,0,0,0,0,0]	[0,0,1,0,0,0]	(5+30)*1=35

Quadratic Arithmetic Program

- Once the prover's computations are represented in R1CS form, the verifier need only validate each corresponding step. However, to further reduce complexity, we aim for a more succinct verification approach than directly checking every step.

- Leveraging Quadratic Arithmetic Programs (QAPs) allows us to transform R1CS outputs into polynomial expressions, thereby streamlining and accelerating the verification process.

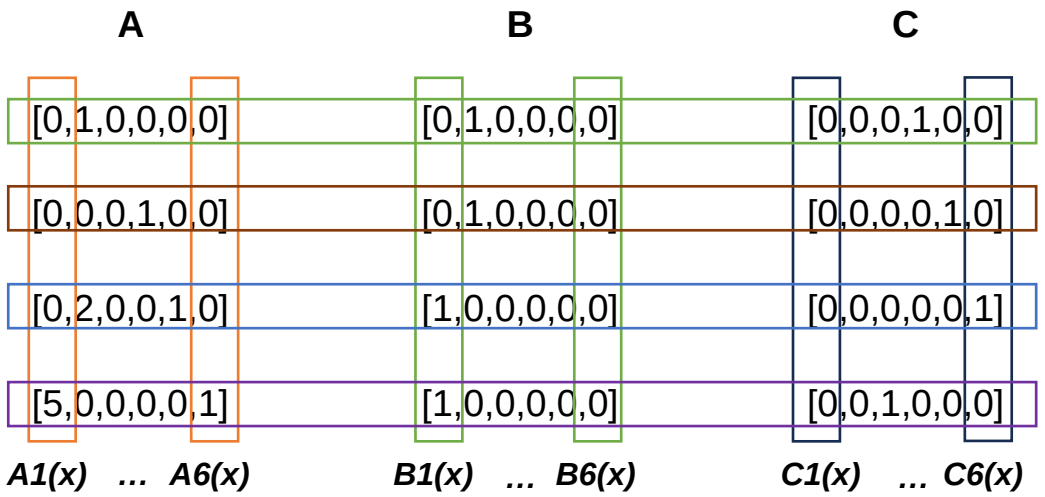
$$P(x) = c_0 + c_1x^1 + c_2x^2 + \ldots + c_dx^d$$

Polynomial Properties

- Two useful properties about polynomials:
  - One expression can embed infinite amount of information.
    - The coefficients of a polynomial can represent arbitrary information.

- Comparing the knowledge of the set of coefficients of a polynomial is easy
  - Just plug in few x values, if Alice can return the expected result, then she must know the set of coefficients.

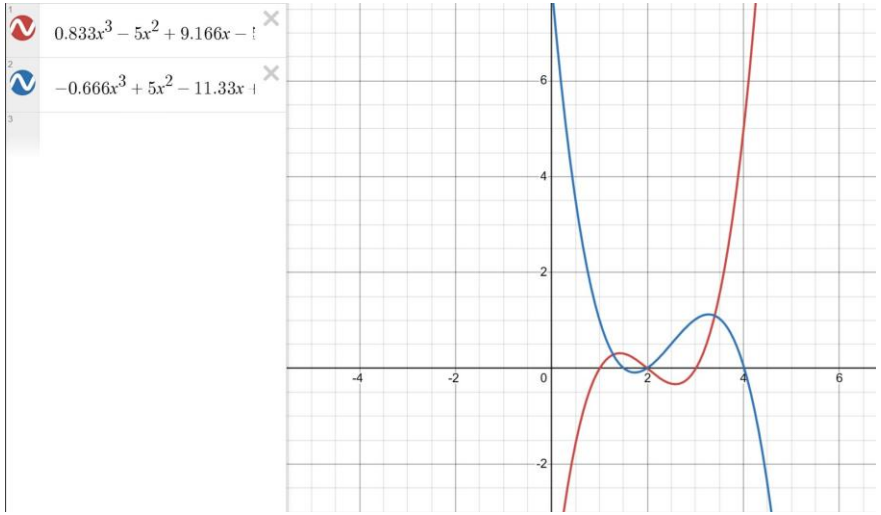
R1CS -> QAP: Representing 4 steps of the calculation



- 4 Sets of vector groups.
- 3 vectors per group.

- Convert to 3 groups of polynomials of degree-3.
- 6 polynomial per group.

$A1(x) = 0.833x^3 – 5x^2 + 9.166x – 5$
$A2(x) = -0.666x^3 – 5x^2 - 11.333x + 8$
…
$B1(x) = -0.333x^3 + 2.5x^2 - 5.166x + 3$
$B2(x) = -0.333x^3 + 2.5x^2 - 5.166x + 3$
…
$C1(x) = 0x^3 + 0x^2 + 0x+ 0$
$C2(x) = 0x^3 + 0x^2 + 0x+ 0$



R1CS -> QAP: Representing 4 steps of the calculation

R1CS

A	B	C
[0,1,0,0,0,0]	[0,1,0,0,0,0]	[0,0,0,1,0,0]
[0,0,0,1,0,0]	[0,1,0,0,0,0]	[0,0,0,0,1,0]
[0,2,0,0,1,0]	[1,0,0,0,0,0]	[0,0,0,0,0,1]
[5,0,0,0,0,1]	[1,0,0,0,0,0]	[0,0,1,0,0,0]

QAP

	x = 1	x = 2	x = 3	x = 4
A	0	0	0	5
	1	0	1	0
	0	0	0	0
	0	1	0	0
	0	0	1	0
	0	0	0	1
	⋮	⋮	⋮	⋮
C	0	0	0	0
	0	0	0	0
	0	0	0	1
	1	1	0	0
	0	0	0	0
	0	0	1	0

Verifying the computation

- Just as how we used the witness [1,3,35,9,27,30] to verify the R1CS matrix, we can use the witness the verify the polynomials.

- The polynomials resulted from the QAP transformation also satisfy the "$\circ$" operator with the witness:

$$A(x) \circ w * B(x) \circ w - C(x) \circ w = H(x) * Z(x)$$

- Z(x) is the Zero Polynomial, where it's zero at all x values, for x in [1..n] where n is the number of gates:

$$(x-1)*(x-2)*(x-3) \ldots *(x-n)$$

Verifying the computation

Checking Work

- $A(x) = -5.166x3 – 38.5x2 – 73.33x + 43$
- $B(x) = 0.666x3 – 5x2 + 10.33x -3$
- $C(x) = 2.833x3 – 24.5x2 + 71.66x – 41$
- $A(x)*B(x) – C(x) = 3.444x6 + 51.5\ x5\ 294.777x4 +805.833x3 -1063.77x2 +592.66x - 88$
- Since we have 4 gates

$$Z(x) = (x-1)(x-2)(x-3)(x-4) = x4 – 10x3 + 35x2 -50x + 24$$

- $A(x)*B(x)-C(x) = H(x)*Z(x)$
- There is a H(x) because there is no remainder in
- $A(x)*B(x)-C(x) / Z(x)$

$$-3.444x6 + 51.5\ x5\ 294.777x4 +805.833x3 -1063.77x2 +592.66x -88 / x4 – 10x3 + 35x2 -50x + 24 = -3.444x2 + 17.055x - 3.666$$

- No remainder!

Proving and Verifying

Common Knowledge

- Common knowledge shared between Alice(prover) and Bob(Verifier)
  - Circuit
  - A1(x), A2(x) …. , An(x)
  - B1(x), B2(x) …. , Bn(x)
  - C1(x), C2(x) …. , Cn(x)
  - Z(x)

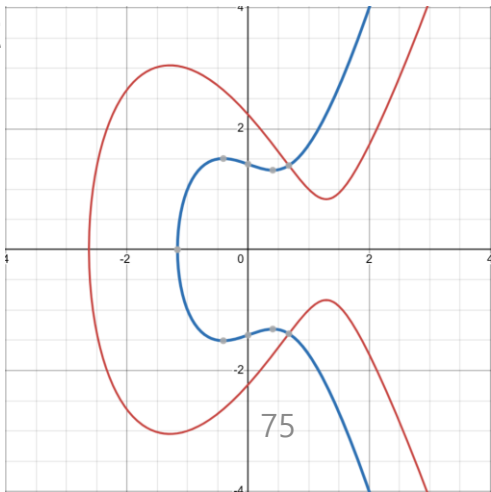
Proving and Verifying

Prover and Verifier

- Prover:
  - Perform the calculations with the circuit and record the witness w(secret)
    - Uses w and calculates:
      - $A(x) = w \circ \{A1(x), A2(x), \ldots, An(x)\}$
      - $B(x) = w \circ \{B1(x), B2(x), \ldots, Bn(x)\}$
      - $C(x) = w \circ \{C1(x), C2(x), \ldots, Cn(x)\}$
    - Then calculate $H(x) = (A(x)*B(x)-C(x)) / Z(x)$, and evaluate A(x), B(x), C(x), H(x) at point s, randomly chosen by the verifier. Share A(s), B(s), C(s), H(s) with verifier.
- Verifier:
  - Check $A(s)*B(s) - C(s) = H(s) * Z(s)$ holds. This proves all the constraints of the circuit have been met.

Problem

- With the transformation so far, the problem becomes this:
- Alice need to prove Bob she know a set of values w(witness), that is the result of correct computation according to the circuit.
- Using Alice's witness value and QAP set of polynomials, Alice calculate polynomials A(x), B(x), C(x), and H(x).
- By providing A(s), B(s), C(s), and H(s), Alice can prove that she performed the calculation correctly.
  - Problem: Alice must not know the value s; Bob must not know the polynomial.
- Bob Must Hide the Evaluation Point s, Otherwise, Alice can cheat by finding different polynomial that produce same expected y value with her invalid input, only at the evaluation point s.



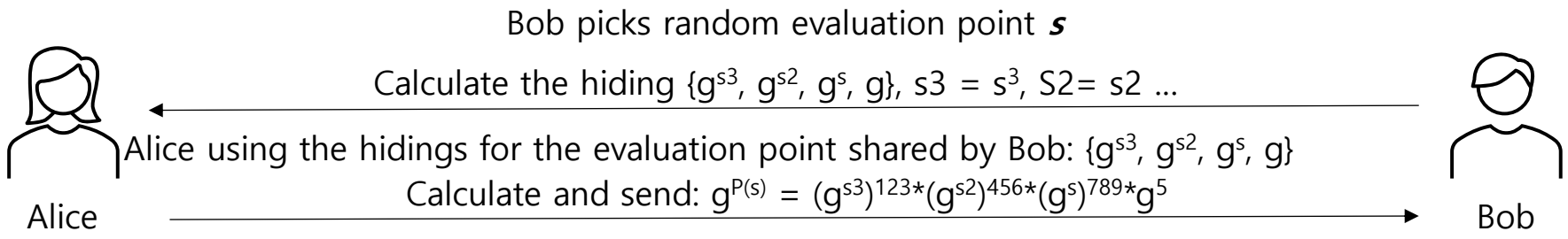
Hiding Shared Evaluation Parameter

Homomorphic Hiding

- Homomorphic Hiding is a cryptographic technique that allows certain computations on encrypted data without revealing the original plaintext.
- After performing operations (like addition or multiplication) on the encrypted data, decrypting the result yields the same outcome as if the computations were done on the plaintext directly.

Blind Evaluation of Polynomials

- Example Challenge: P(x) 123x3 + 456x2 + 789x + 5



Convert to Non-interactive Proof

Common Reference String

- Using a Common Reference String (CRS) obtained from Trusted Setup

- A CRS is a set of cryptographic parameters generated once during a trusted setup phase. Both the prover and verifier use this CRS to construct and verify zero-knowledge proofs without revealing the underlying secret information.

- Role in ZK-SNARKs:
  - Universal Parameters: The same CRS can be reused for multiple proofs, streamlining the proof process.
  - Guaranteed Soundness & Zero-Knowledge: The CRS helps maintain the core properties of ZK-SNARKs, ensuring proofs are both verifiable and privacy-preserving.

Convert to Non-interactive Proof

Trust-setup output

- Contains the proving key and verifying key
  - Proving key: hidings of A1(s), A2(s)…, and the same for B and C, s is the secret evaluation point
  - Verifying key: hiding related to the public inputs

Why ZK-SNARK?

Benefits of Integrating Ring Signature

- ZK-SNARKs produce succinct proofs, meaning the proof size is small, which saves bandwidth and storage space.
- Verification is also extremely fast, making it suitable for decentralized systems like blockchains.
- ZK-SNARKs are non-interactive, meaning that the prover and verifier do not need to engage in repeated interactions.
- This characteristic is advantageous in distributed environments, reducing latency and improving scalability.

ZKP Method	Features	Advantage	Disadvantage
ZK-SNARKs	Succinct and non-interactive proofs	- Small proof size and fast verification - Non-interactive structure is efficient	- Requires trusted setup - Complex mathematical computations
ZK-STARKs	Transparent setup, Quantum resistance	- No trust setup required	- Larger proof sizes increase storage and transmission costs
Bulletproofs	Short proofs but interactive	- Compact proof size - No trusted setup required	- Interactive structure requires repeated communication
Interactive ZKP	Proof generation involves multiple rounds of interaction	- Simpler computational process- No trusted setup required	- High network latency due to interactive nature

Appendix 2.
Proof Aggregation

Proof Aggregation

Problem

- What if hacker packet snipping between user and proxy server?

KZG Commitment

- Polynomial commitment with linearity, enabling efficient aggregation.

Goal

- Aggregate multiple witness' proof
- Maximize anonymity using KZG commitment

Motivation for KZG Commitment

Challenge

- Aggregate multiple user proofs into one enhance anonymity.
- Avoid complex hash/Merkle tree method, using polynomial commitments instead.

Feature of KZG commitment

- Commit a polynomial P(x) as $C = \sum a_i[s^iG]$.
- Supports linear operations and proof aggregation.
- Efficiently verifies evaluations at specific points.

KZG Commitment Proof

Polynomial

- $P(x)=a_0+a_1x+a_2x^2+\cdots+a_dx^d$

Commitment

- $C= \sum_{i=0}^{d} a_i(siG)$, $C \in G_1$

Proof

- To prove P(z) at z:
- $Q(x)=(P(x)-P(z))/x-z \Rightarrow \pi=Q(s)G$

- Here, $\pi \in G1$ is the proof

KZG Commitments Verification

Verification Equation

- Check_Equal(e(C−P(z)G,H),e(π,zH+G))
- Above function check left hand side and right hand side same.
- If same return true, else false.
- True means verification success.

- C: Commitment to P(x)
- H,zH+G: Points on G2 incorporating evaluation point z.
- π: Proof of correctness
- Paring C – P(z)G and π , then prove P(z)’s correctness

Aggregation Concept for Multiple Users

Witness Aggregation

- Each user provides a proof π_i for polynomial $P_i(x)$.

Linear Combination

- $P_{agg}(x) = \sum_{i=0}^{n} c_i P_i(x)$
- $C_{agg}(x) = \sum_{i=0}^{n} c_i C_i$
- $\pi_{agg} = \sum_{i=0}^{n} c_i \pi_i$
- $Pagg(z) = \sum_{i=0}^{n} c_i P_i(z)$
- Because of the linearity of KZG, the proof π_{agg} for combine P_{agg} is formed well.

Advantage of Aggregation

Mechanism

- Aggregated proof combines multiple proofs, making individual identities indistinguishable.
- Add shuffle arguments to randomize proof order

Advantages

- Aggregated proof π_{agg} is compact and hides individual proofs

Limitations of Traditional Aggregation Techniques

KZG Commitment

- Pros
  - Provides linearity and efficient polynomial commitments.
- Cons
  - Require complex trusted setup.
  - Increase proof size and computation costs.

Hash-Based Aggregation

- Pros
  - Simple and widely used.
- Cons
  - Lack of linearity, lose benefit of pairing-friendly elliptic curve.

Proof Shuffle & Selective Verification

Proof Representation

- Each proof $\pi_i = (A_i, B_i, C_i)$, $A_i \in G_1$, $B_i \in G_2$, $C_i \in G_1$
- Where G_1, G_2 are groups on BN128 elliptic curve
- Pairing e : $G_1 * G_2 \rightarrow G_T$

Verification Equation

- isEqual(e(A_i, B_i), e(C_i, γ)), $\gamma \in G_2$
- If return value true, the proof is valid.

Create proof and shuffling

Proof List

- Collected proofs are presented as
- $\pi = [\pi_1, \pi_2, \pi_3, \ldots, \pi_n]$

Shuffling the Proof list

- $\pi_{shuffled} = [\pi_{\sigma 1}, \pi_{\sigma 2}, \pi_{\sigma 3}, \ldots, \pi_{\sigma n}]$

Verify Proof

- For each $\pi_i \in \pi_{shuffled}$
- isEqual(e(A_i, Bi), e(C_i, γ))
- Stop as soon as one valid proof is found

Mathematical advantage

Anonymity via shuffling

- Proofs are permuted with σ, $\pi_{shuffled} = [\pi_{\sigma 1}, \pi_{\sigma 2}, \pi_{\sigma 3}, \ldots, \pi_{\sigma n}]$

Efficiency in verifications:

- Only evaluate pairing equation until one valid proof is found:
- $\exists_i$ such that $e(A_i,B_i)=e(C_i,\gamma)$

No Aggregation Overhead

- Avoid computationally expensive operations like: $\pi_{agg} - \sum_{i=0}^{n} c_i \pi_i$

Appendix 3.
Ring Signature

Why Ring-Signature?

Benefits of Integrating Ring Signature

- Sender-Centric Anonymity:
  - Protects the sender's identity without relying on predefined groups or central authorities.
- Flexibility in Group Selection:
  - The sender can dynamically form a group of public keys, offering greater adaptability.
- Decentralized Structure:
  - No need for trusted third parties or pre-registered groups, unlike Group Signature.

	Anonymity Scope	Flexibility	Decentralization
Blind Signature	Sender's data and identity	Limited to one signer	Central authority required
Group Signature	Sender within a defined group	Requires group pre-registration	Requires group manager
Ring Signature	Sender within a self-chosen group	No group setup needed	Fully decentralized

How does a ring signature work?

How does a ring signature work?

- A signer creates a signature that looks like it could have been created by any member of a predefined group.
- Verifiers can confirm that the signature is valid but cannot determine the exact signer.

Benefits of ring signatures

- Anonymity: Protects the identity of the signer.
- Unlinkability: Prevents linking multiple signatures to the same signer.
- Efficiency: Relatively fast to generate and verify.

Mathematical Implementation and Proof of Ring Signature

Components of Ring Signature

- Key Setup
  - Private Key: The sender's private key xi.
  - Public Keys: A set of public keys $\{P1, P2, \ldots, Pn\}$, where $Pj = gxj \bmod p$
  - Parameters:
    - g: A generator (public parameter).p
    - : A large prime, where g is a primitive root modulo p

Ring Signature Generation Process

Hash Calculation

For a given message m, compute its hash:
h=H(m)

Random Values

Generate random values sj for all $j \neq i$(keys other than the signer's key).
Leave si undefined initially, to be calculated using the private key.

Public Key Computation

For each public key Pj , calculate:$vj = gsj \cdot (Pj)cj \bmod p$ where cj is a value derived from a hash.
Create a cyclic chain of values:$cj+1 = H(vj)$
Ensure that the cyclic condition $cn+1 \equiv c1$ is satisfied.

Private Key Computation

The signer computes their si using their private key xi and a random value r: $si = r - ci \cdot xi \bmod (p-1)$

Ring Signature Generation Process

Final Output of Ring Signature

The ring signature is composed of: Signature=($c1,s1,s2,\ldots,sn$)

Verification Process for Ring Signature

Step 1: Recompute Public Values

Using the signature values $c1,s1,\ldots,sn$and public keys Pj , the verifier computes:

$$vj=gsj\cdot(Pj)cj\text{mod } p$$

Step 2: Chain Verification

Verify that the cyclic chain condition is satisfied:

$$cj+1=H(vj)$$

Finally, confirm that $cn+1\equiv c$.

Mathematical Guarantee

The cyclic structure of the signature $c1,v1,\ldots,vn$ prevents the verifier from dedu cing which private key xi was used.

If H is a cryptographically secure hash function, $cj+1$ and vj cannot be inverte d to reveal the private key xi .

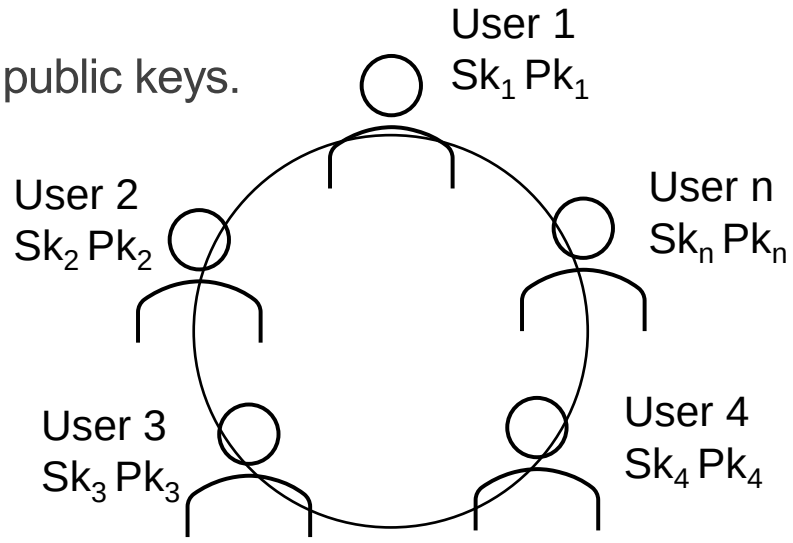
Integrating Ring Signature into ZK-SNARK Witness

Ring Signature

- A Ring Signature is a cryptographic method that allows a member of a group to sign a message on behalf of the group while ensuring anonymity. The signature ensures that it is computationally infeasible to determine which member of the group signed the message, providing both privacy and authenticity.

Witness with Ring Signature

- **The ZK-SNARK witness now includes Ring Signature components:**
  - Secret key (SK): The private key corresponding to one of the group's public keys.
  - Index (idx): The position of the signer in the group.
  - Ring Signature values: $\{s_i\}$,c_0,Key Image, and intermediate values.



- **These elements allow the circuit to verify:**
  - The signature is valid for the group.
  - The signer remains anonymous among the group members.
  - Double-spending is prevented using Nullifier (Key Image hash).

Constructing the Witness with Ring Signature

Components of the Witness

- **To integrate Ring Signature into a ZK-SNARK circuit, the witness includes both private inputs and intermediate values needed to verify the Ring Signature while maintaining anonymity.**
- **Private Inputs:**
  - Secret Key (SK):The private key corresponding to the signer's public key in the group.
  - Index (idx):Position of the signer's public key in the group.
- **Intermediate Values:**
  - Random Scalars (ri):Random values used to compute the Ring Signature components.
  - Chain Values (c_0,$\{c_i\}$):Values that link each step of the signature together.
  - Key Image: Prevents reuse of the same secret key by providing a one-way hash of the key.
    - Key Image=$H(SK \cdot G)$
    - Nullifier=H(Key Image)

Constructing the Witness with Ring Signature

Ring Signature Validation in the Circuit

- **Key Image Generation**: The circuit ensures the Key Image is correctly generated using the secret key: Key Image=$H(SK \cdot G)$

- **Nullifier Generation**: The Nullifier is derived from the Key Image to prevent secret key reuse: Nullifier=H(Key Image)

- **Ring Signature Verification**: The circuit verifies the Ring Signature chain by checking the values:c_i+1=$H(g^s{}_i \cdot \text{pk}_i{}^c)$
  - The witness provides s_i and c_0 for each step.
  - Preventing Double-Spending: The circuit compares the Nullifier with previously used Nullifiers to confirm:
  - Nullifier ∉ Used Nullifier List

Glossary of ZK-R-SNARK terms

Variable	Description	Public/Private
r	Random value generated by the prover.	Private
R	R=g r , group element derived from the random value r	Public
Sk_i	Secret key of the prover (private key).	Private
P_i	$P_i=g^{sk}{}_i$, public key of the i-th ring member.	Public
D	Data to be proven (e.g., data ID or hash).	Public
C_i	Component of the ring signature associated with P_i.	Public
σ	Ring signature composed of $R,c_1,c_2,\ldots,c_n$.	Public
H	Cryptographic hash function.	Public
g	Generator used in elliptic curve cryptography or cryptographic operations.	Public
Z_p	Finite field p where cryptographic operations are performed.	Public
Proof	Proof provided to the verifier, consisting of σ (ring signature) and D (data)	Public

Benefits of Integrating Ring Signature

- **Anonymity**: The circuit validates that the signer is part of the group without revealing the specific signer.
- **Double-Spending Prevention**: The Nullifier ensures that the same private key cannot be used multiple times.
- **Efficiency**: Ring Signature verification is embedded within the ZK-SNARK circuit, enabling compressed proofs and efficient verification on the blockchain.

Aspect	General Witness Generator	Proposed Witness generator
Purpose	Simple condition verification	Anonymity, Ring signature Validation, Nullifier
Private input	Single secret value (x)	Secret Key (SK) and index (idx)
Intermediate Values	None	P, s_i, c_0
Computation Complexity	Low	High
Anonymity	none	Hide singer identity among a group

Advantages of ZK-R-SNARK

Challenges in Prior Research

- **Identity Traceability:**
  - Public keys or blockchain wallet addresses, when repeatedly used in smart contracts, can allow observers to identify and trace contributors over time through transaction analysis.
- **Limited Anonymization:**
  - Existing solutions offer incomplete anonymity, often relying on intermediaries or weak pseudonymization.
  - Long-term usage risks exposing contributors, leading to reputational or business risks and reluctance to share CTI.

Key Differentiator

- Unlike prior methods that expose contributors to identity inference risks, our solution integrates advanced cryptographic techniques (ZKP + Ring Signatures) to offer:
  - **Strong Data Credibility**: CTI entries are verifiable and reliable.
  - **Formal Anonymity Guarantee**: Contributor identities remain private, even during extended blockchain usage.

Appendix 4. Odds

Current Industrial Solution

Existing CTI Practices and Their Shortcomings, Relying on Centralized Entities, Complex Standards, and Limited Anonymity

Current Practices

- Centralized CTI hubs, government initiatives(CISA, NCSC, ENISA), industry alliances.
- Standards like STIX/TAXII, MITRE ATT&CK framework.
- SIEM tools for detection and analysis.

Limitations

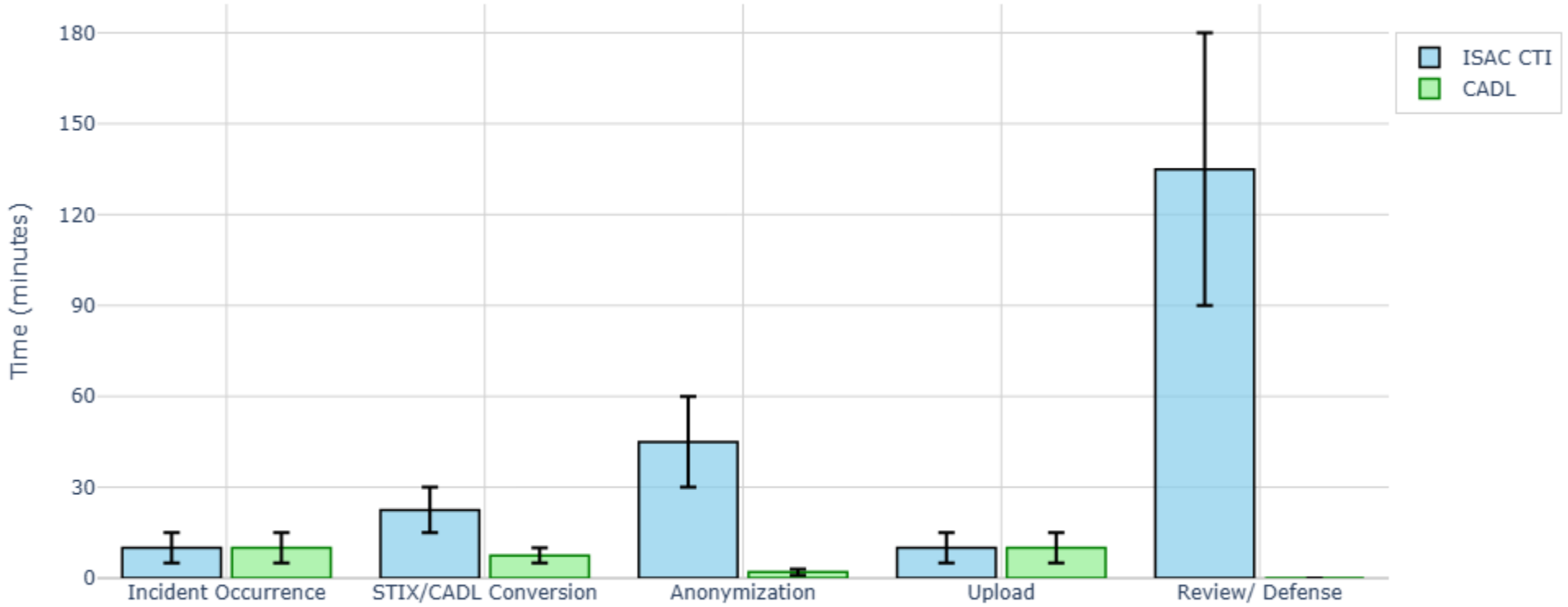
- No support for anonymity.
- Need for privacy-preserving systems
- Trust still relies on central entities.
- Complex format not easily digestible by all stakeholders.

Circom (ZKP Implementation)

Circuit Design & Constraints Setup

- **Inputs:**
  - Private Inputs: The signer's secret key (or membership credential), which remains hidden. Any necessary randomness used in generating the ring signature.
- **Constraints:**
  - **Signature Validity**:
    - The circuit enforces arithmetic constraints that validate the ring signature.
  - **Cryptographic Relation Enforcement**:
    - Staring value and end value must be equal in the ring.These constraints ensure that each public key in the ring contributes properly to the computation, yet the process does not reveal which specific key was used, maintaining anonymity.
  - **Zero-Knowledge Guarantee**:
    - If any of these conditions fail, no valid proof can be generated. This means only when all constraints are met does the circuit output a valid signal.
  - **Outputs**:
    - A single public signal that indicates whether the ring signature verification succeeded.

Task-wise Time Comparison



Smart Contract

```
function verifyProof(uint[2] calldata _pA, uint[2][2] calldata _pB, uint[2] calldata _pC, uint[<%=IC.length-1%>] calldata _pubSignals) public view returns (bool) {
    assembly {
        function checkField(v) {
            if iszero(lt(v, q)) {
                mstore(0, 0)
                return(0, 0x20)
            }
        }

        // G1 function to multiply a G1 value(x,y) to value in an address
        function g1_mulAccC(pR, x, y, s) {
            let success
            let mIn := mload(0x40)
            mstore(mIn, x)
            mstore(add(mIn, 32), y)
            mstore(add(mIn, 64), s)

            success := staticcall(sub(gas(), 2000), 7, mIn, 96, mIn, 64)

            if iszero(success) {
                mstore(0, 0)
                return(0, 0x20)
            }

            mstore(add(mIn, 64), mload(pR))
            mstore(add(mIn, 96), mload(add(pR, 32)))

            success := staticcall(sub(gas(), 2000), 6, mIn, 128, pR, 64)

            if iszero(success) {
                mstore(0, 0)
                return(0, 0x20)
            }
        }

        function checkPairing(pA, pB, pC, pubSignals, pMem) -> isOk {
            let _pPairing := add(pMem, pPairing)
            let _pVk := add(pMem, pVk)

            mstore(_pVk, IC0x)
            mstore(add(_pVk, 32), IC0y)
```

```
pragma circom 2.0.0;

template Multiplier2() {
    signal input a;
    signal input b;
    signal output c;
    c <== a*b;
}

component main = Multiplier2();
```

Summary of Research

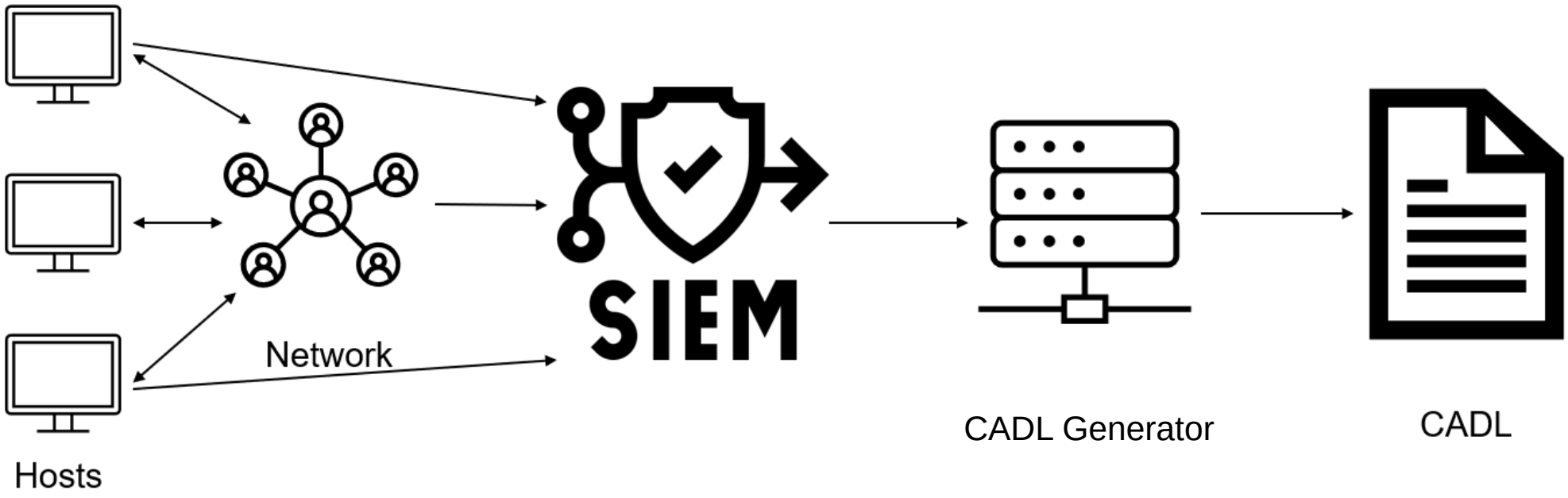
- This research introduces an anonymous CTI sharing platform by integrating ZK-SNARKs and Ring Signatures. This system addresses key challenges in the cybersecurity industry: the hesitation to share sensitive threat data due to privacy concerns and the risk of identity exposure.

- ZK-SNARKs ensure that participants can prove eligibility to share and access CTI data without revealing any private information.

- Ring Signatures allow users to anonymously contribute threat data, ensuring their identity remains untraceable within a group.

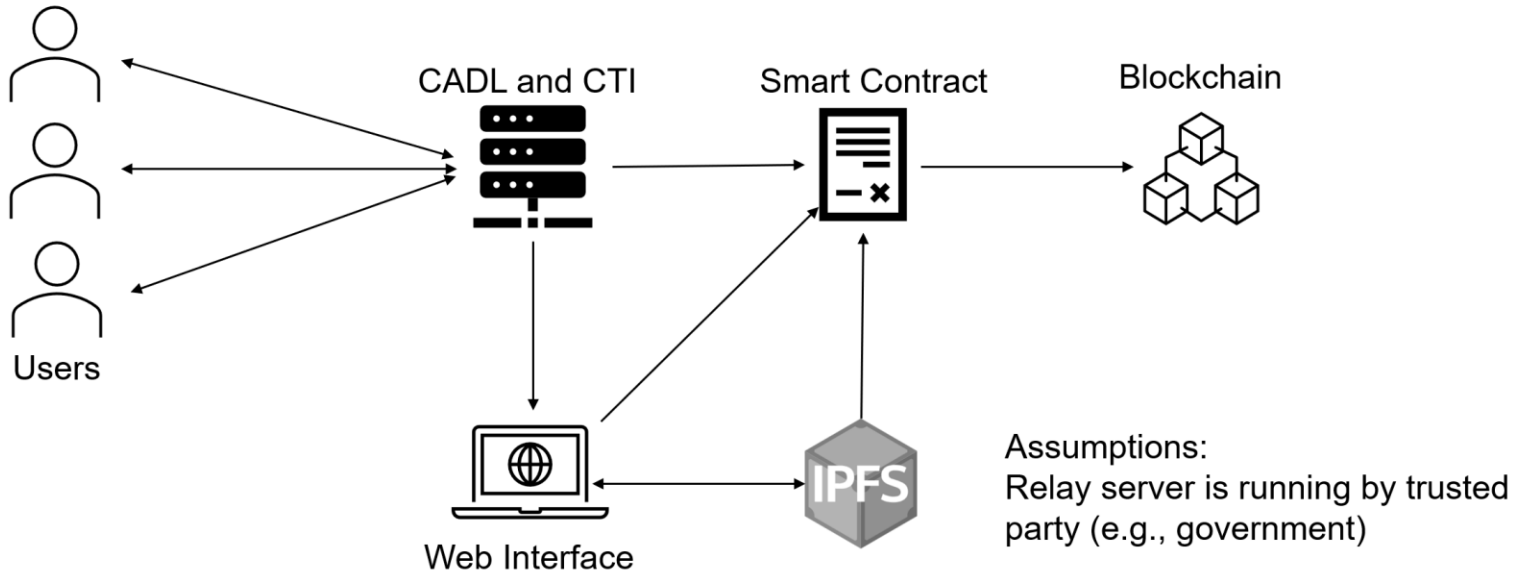
- Blockchain and Smart Contracts provide transparency and immutability, enabling secure storage of CTI metadata (e.g., IPFS CIDs) while fostering trust between participants.

CADL Collection Process : Overview



CTI Sharing on Blockchain

- User prove they are authorized to share CTI without revealing their identity.
- Participants receive a token to access new CTI.
- CTI (CADL Format) is uploaded IPFS, ensure decentralization.
- IPFS CID is stored on-chain for tamper-proof tracking.
- Witness contains ring signature function for anonymity.

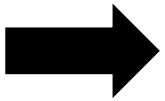


Comparison of Witness Generation Methods

➢To highlight the advancements of our proposed approach, let's compare a simple witness generation method with our Ring Signature-based witness generation:

```python
# Public Inputs: y, z
# Private Input (Witness): x
# Circuit: Check if x + y = z

def generate_witness(x, y, z):
    witness = {"x": x}  # Witness includes only the secret value x.

    # Local validation (pre-check)
    if x + y != z:
        raise Error("Invalid witness: condition does not hold.")

    return witness
```

Conventional Method



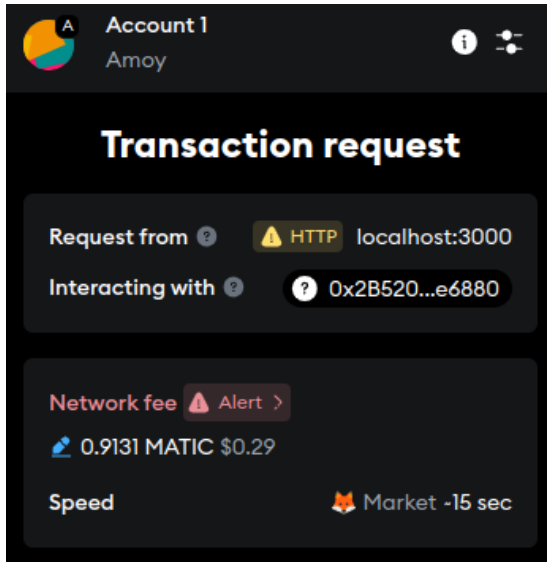
```
pragma circom 2.0.0;

template ToyRingSignature(n, P) {
    signal input pks[n];
    signal input c[n];
    signal input s[n];

    signal sumVals[n];
    signal q[n];
    signal internalC[n];

    for (let i = 0; i < n-1; i++) {
        sumVals[i] <== c[i] + pks[i] + s[i];
        sumVals[i] === q[i] * P + internalC[i];
        c[i+1] === internalC[i];
    }
    sumVals[n-1] <== c[n-1] + pks[n-1] + s[n-1];
    sumVals[n-1] === q[n-1]*P + internalC[n-1];
    internalC[n-1] === c[0];


    signal output ringCheck;
    ringCheck <== 1;
}

// n=3, P=97 예시
component main = ToyRingSignature(3, 97);
```

New Method

Gas Price

Gas

➢ **Gas** fee refers to the transaction cost required to execute operations on a blockchain network, particularly in Ethereum and EVM-compatible blockchains. Since smart contracts involve computational execution beyond simple transfers, users must pay gas fees to compensate network validators (miners or stakers) for processing transactions and securing the network.



CADL Format and Operation

What kind of parameters are there?

- Sysmon/Suricata Log Files: Automatically generated data from security monitoring tools.
- Human Input: Contextual details, such as observed attacker behavior or known vulnerabilities.

What is the format?

- Yet Another Markup Language(YAML)
- Lightweight, structured format designed for easy parsing, reading and integration.

What is the typical size?

- As a text-based format, CADL files are expected to remain lightweight and efficient for storage and transfer.

What database does it use?

- Currently, CADL uses IPFS for storage, but a graph database is being considered as future research to better analyze and represent cyber attack relationships.

Why Blockchain?

Blockchain's Advantages

- Ensures trust and tamper-resistance without a central authority
- Enhances security through transparency and decentralization

limitation

- All transaction details are publicly visible, risking exposure of personal or corporate data
- Growing need for enhanced privacy protection

Proposed Solution

- Verify transaction validity without revealing specifics
- Maintain blockchain transparency while protecting sensitive information
- Achieve both trust and anonymity for next-generation solutions

Advantages of the Proposed Solution

Privacy & Anonymity

- ZKP ensures no contributor's identity is revealed
- Ring Signature obscures individual proof origins, maximizing anonymity without relying on complex setups

Trustless & Tamper-proof

- Blockchain and IPFS ensure data integrity and immutability without central authorities

Interoperability

- CADL bridges STIX/TAXII and SIEM logs, enhancing understanding and standardization

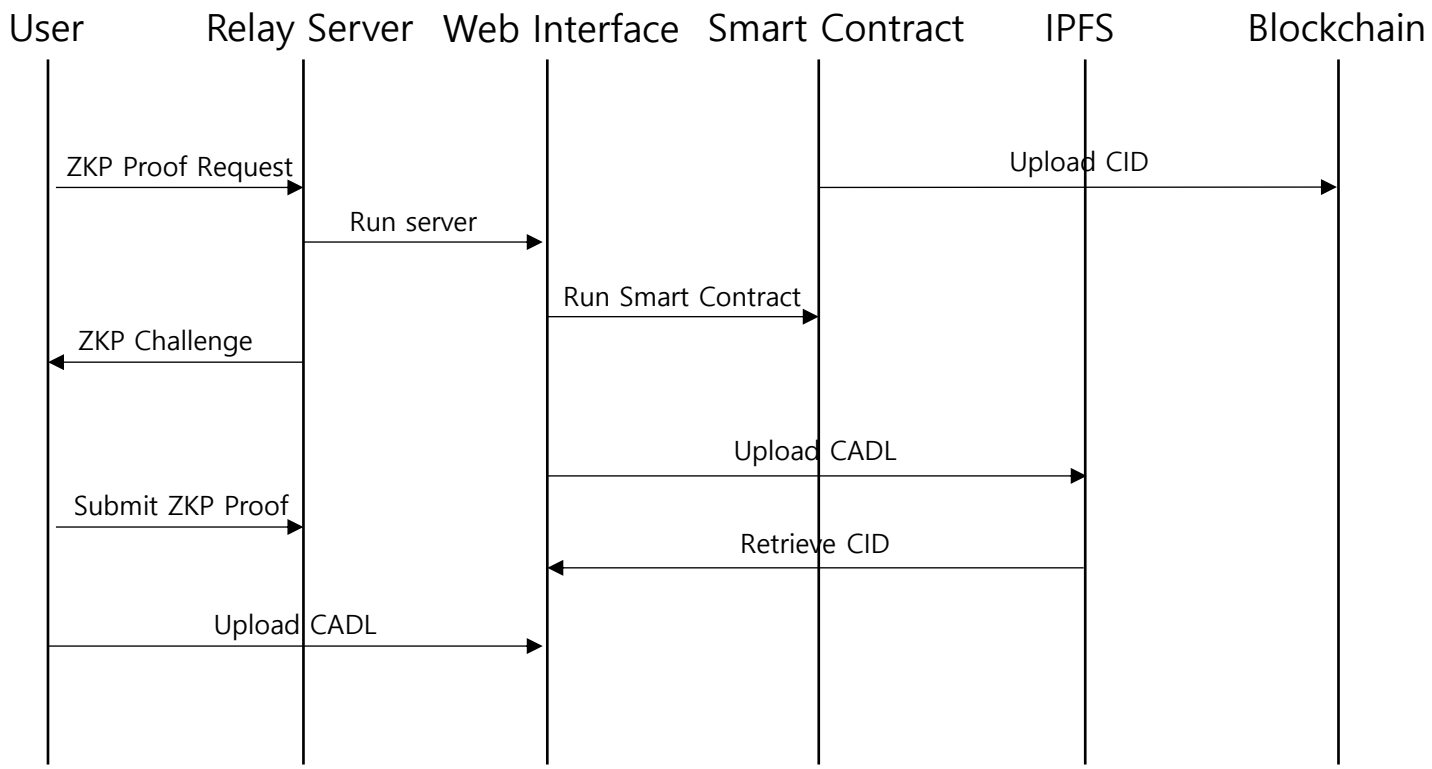
Scalability & Efficiency

- Using Ring signature avoids heavy polynomial operations, improving verification speed and resource usage at scale

Impact on Cybersecurity Industry

- **Enhanced Collaboration**: Organizations can safely share critical cyber threat information without fear of exposure or reputational damage.
- **Stronger Defense Capabilities**: Timely and anonymous threat sharing improves collective detection and response to cyber attacks.
- **Encouraged Participation**: The platform incentivizes active participation while maintaining complete anonymity, overcoming traditional barriers to information sharing.
- By addressing these challenges, our solution empowers organizations to work together securely, paving the way for a more resilient and collaborative cybersecurity ecosystem.

System Diagram



Why is CTI Sharing Necessary?

Benefits of CTI Sharing

- **Enhanced Threat Detection:**
  - By sharing threat patterns, malware signatures, and attack vectors, organizations can identify and mitigate threats faster.
- **Improved Incident Response:**
  - Collaborative efforts allow organizations to take preemptive actions against identified threats, reducing downtime and damage.
- **Stronger Defense Ecosystem:**
  - Collective intelligence strengthens overall cybersecurity resilience, making it harder for attackers to exploit similar vulnerabilities across multiple entities.

Limitations

- No support for anonymity.
- Need for privacy-preserving systems
- Trust still relies on central entities.
- Complex format not easily digestible by all stakeholders.