Zero Knowledge Proofs:

Challenges, Applications, and Real-world Deployment

NIST Workshop on Privacy Enhancing Cryptography

September 26th, 2024

Tjerand Silde & Akira Takahashi

NTNU

J.P.Morgan

AlgoCRYPT CoE

AI Research

This talk

1) Introduction to Zero Knowledge Proof (Akira)

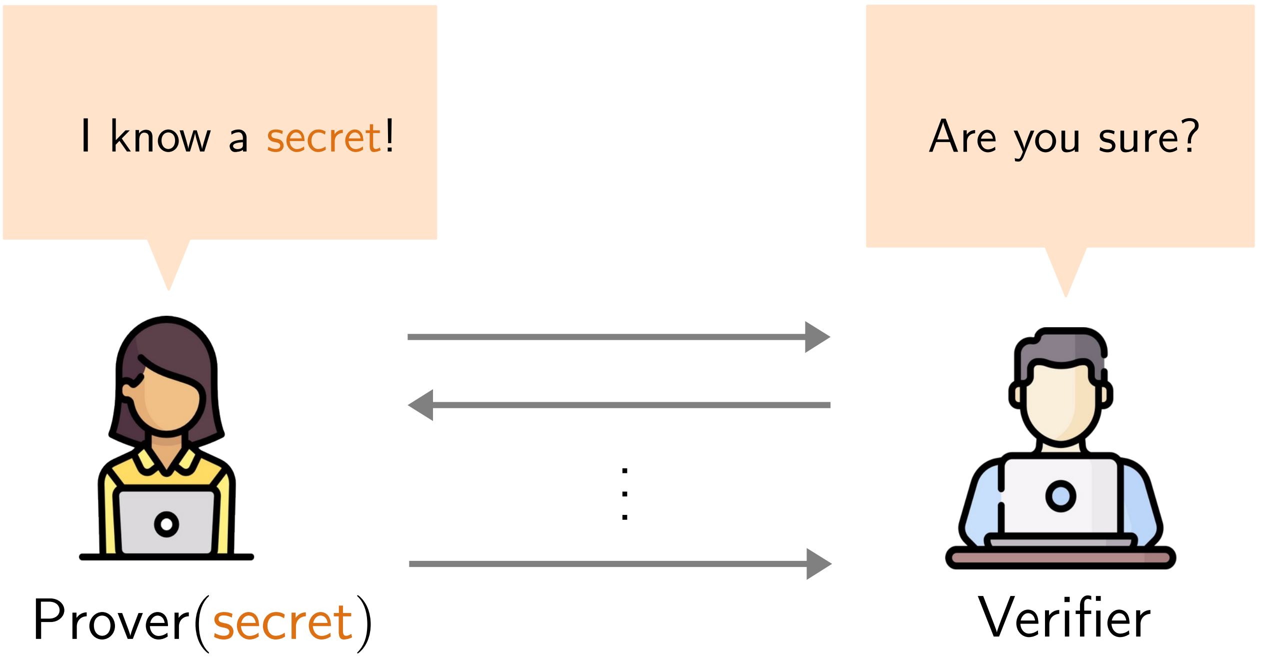
2) Technical Challenges (Akira)

3) Real-World Applications (Tjerand)

4) Insights from ZKP Workshop (Tjerand)

5) Resources and Standards (Tjerand)

What is Zero Knowledge Proof?



Basics

- ZKP is a two-party protocol, consisting of **Prover** and **Verifier**
- With ZKP, Prover can convince Verifier that she has some secret information without disclosing the secret
- Example: “I know sk corresponding to pk”
- Long history of research starting from the ‘80s [GMR85]. Lots of efficiency improvements during the last decade
  - cf. **ZK-SNARK** (Succinct Non-interactive Argument of Knowledge)

Syntax of ZKP

- x: statement (i.e. public input)
- w: witness (i.e. secret input)
- R: relation function, outputting 1 or 0
- "I know w s.t. $R(x, w) = 1$"

b: decision bit

Prover(x, w)

Verifier(x)

$b = 1$: "accept"

$b = 0$: "reject"

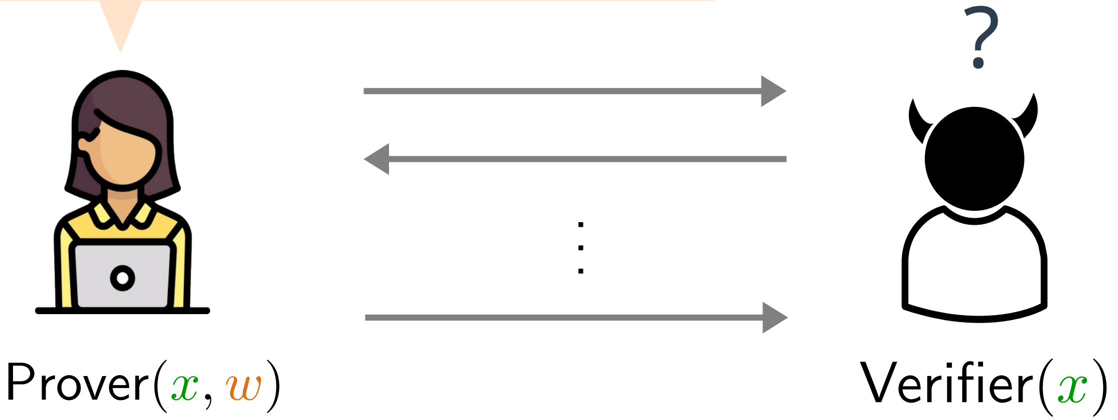
Completeness

- If Prover and Verifier honestly follow the protocol, then Verifier halts by outputting $b = 1$

Security Goals of Zero Knowledge Proof

- x: statement (i.e. public input)
- w: witness (i.e. secret input)
- R: relation function, outputting 1 or 0
- "I know w s.t. $R(x, w) = 1$"

- Tries to steal w



Zero Knowledge (ZK)

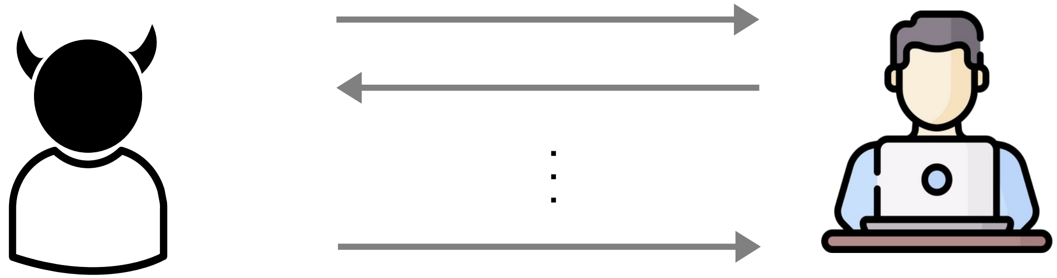
- Protecting against malicious verifier
- Verifier learns nothing about Prover's secret
- Formally, ZK is guaranteed by showing the existence of "Simulator"

Security Goals of Zero Knowledge Proof

- Maliciously interacts with Verifier
- Might use an "invalid" input

$$R(x^*, w^*) = 0$$

You lied!



Prover(x^*, w^*) Verifier(x^*)

$b = 0$: "reject"

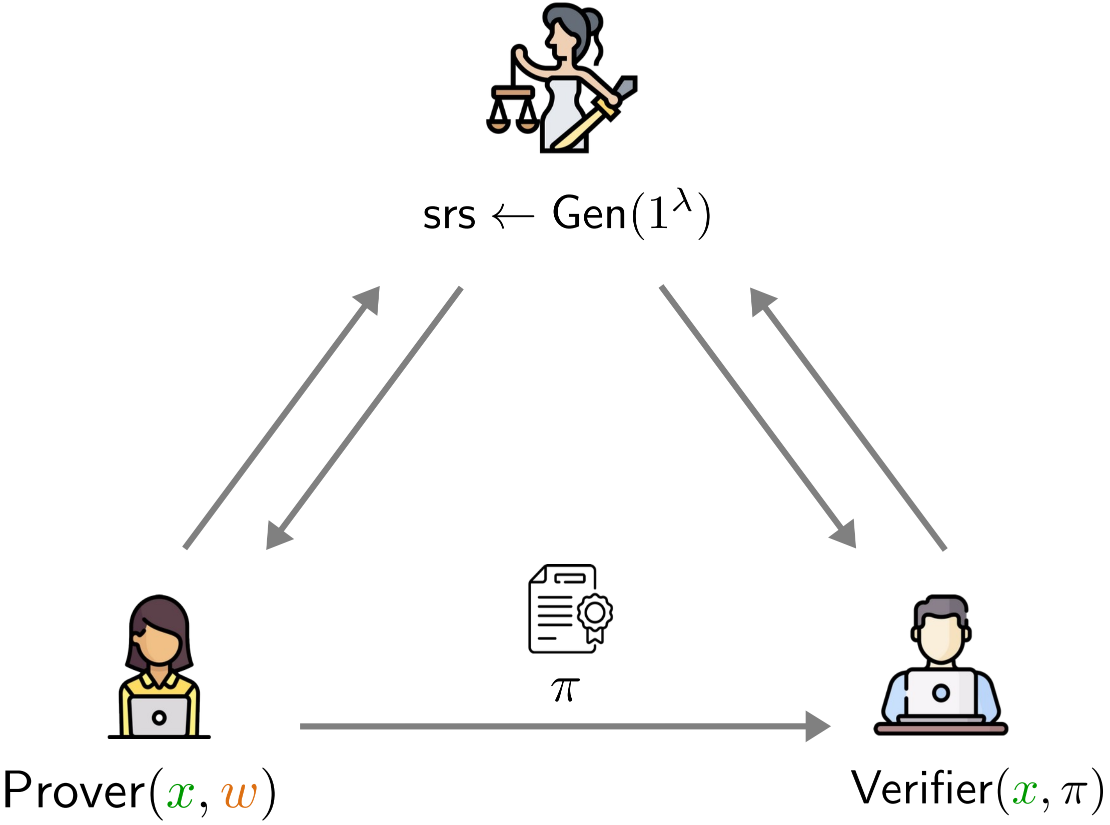
Zero Knowledge (ZK)

- Protecting against malicious verifier
- Verifier learns nothing about Prover's secret
- Formally, ZK is guaranteed by showing the existence of "Simulator"

Knowledge Soundness (KSND)

- Protecting against malicious prover
- If Prover uses an invalid secret, then Verifier catches it with high probability
- Formally, knowledge soundness is guaranteed by showing the existence of "Knowledge Extractor"

Non-interactive Zero Knowledge Proof (NIZK)



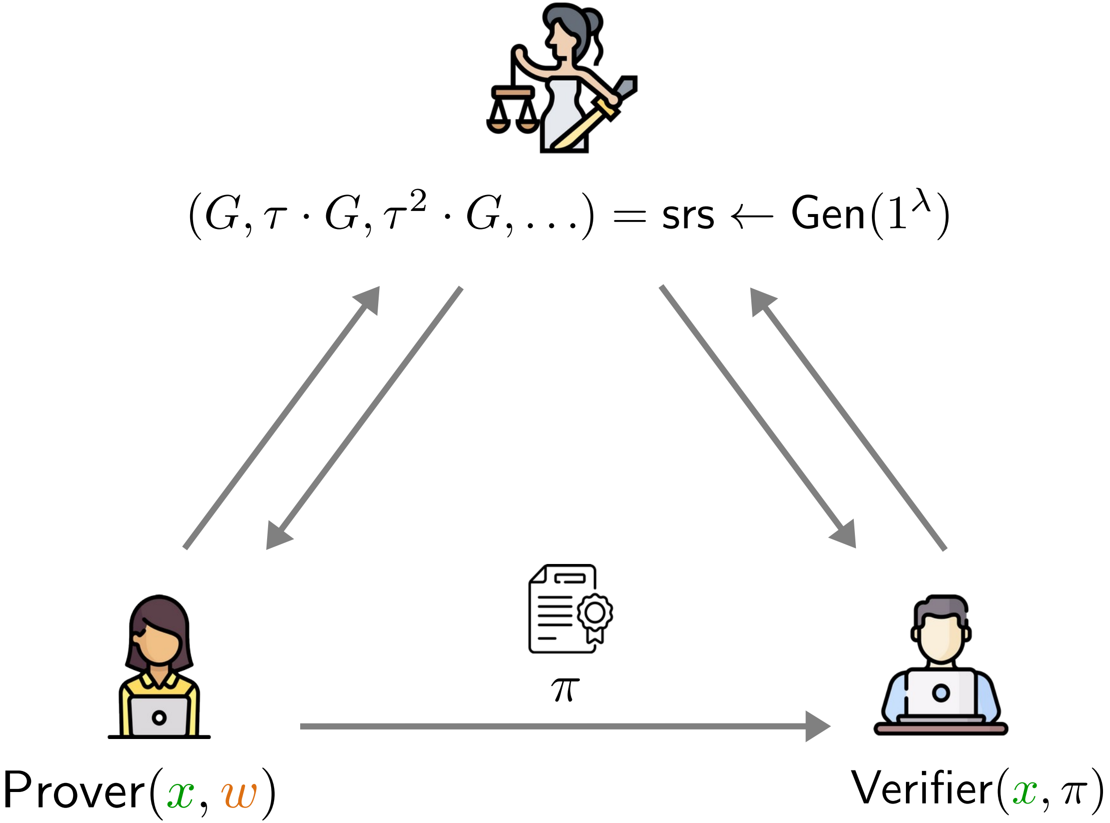
Removing Interactions

- Ideally, Prover should create a one-shot proof string π
- Verifier checks π asynchronously
- Such π is reusable and can be checked by potentially many verifiers

Types of Trusted Setup

- Structured Reference String (SRS)
- Hash function modeled as Random Oracle
- Or both!

Non-interactive Zero Knowledge Proof (NIZK)



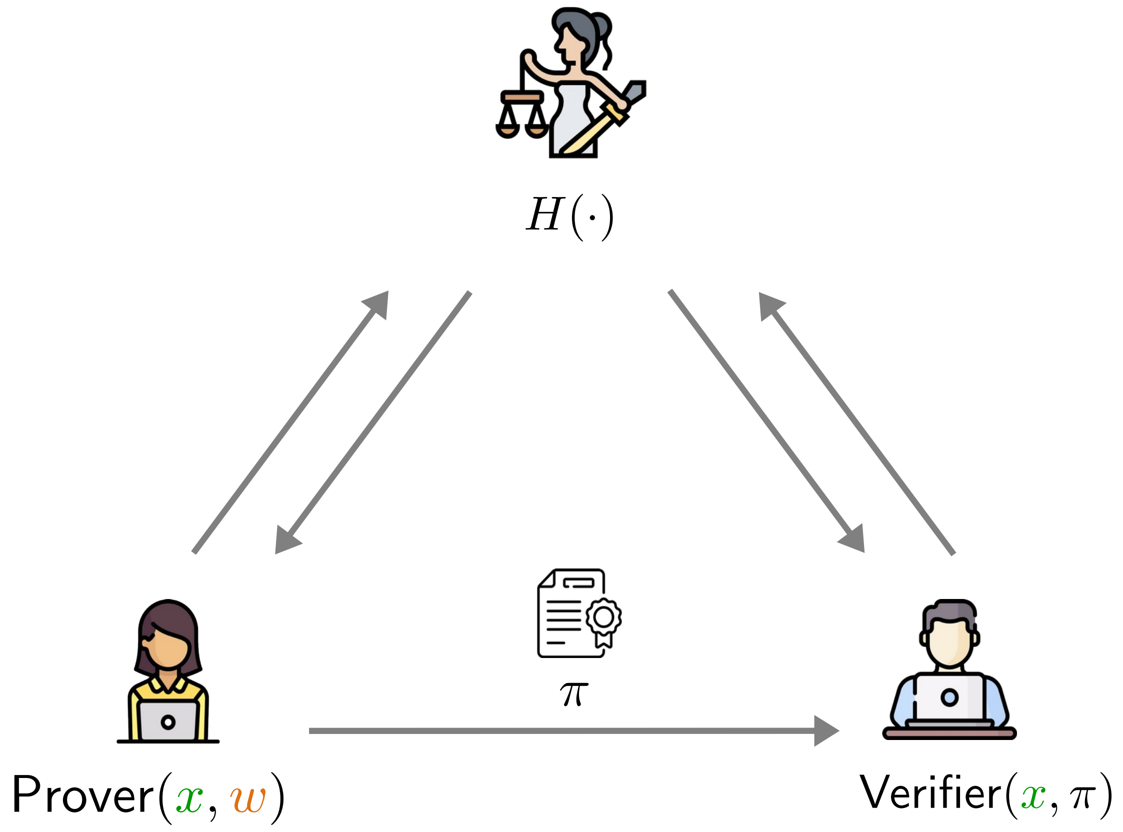
Removing Interactions

- Ideally, Prover should create a one-shot proof string π
- Verifier checks π asynchronously
- Such π is reusable and can be checked by potentially many verifiers

Types of Trusted Setup

- Structured Reference String (SRS)
- Hash function modeled as Random Oracle
- Or both!

Non-interactive Zero Knowledge Proof (NIZK)



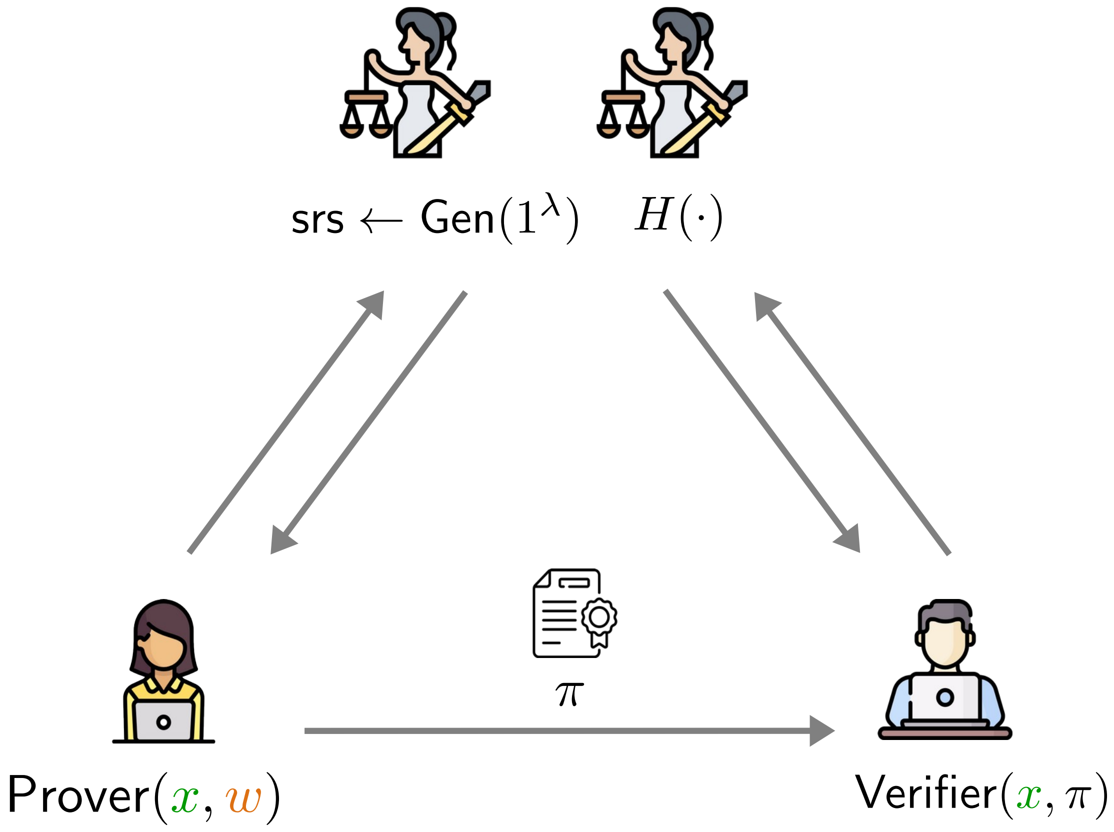
Removing Interactions

- Ideally, Prover should create a one-shot proof string π
- Verifier checks π asynchronously
- Such π is reusable and can be checked by potentially many verifiers

Types of Trusted Setup

- Structured Reference String (SRS)
- Hash function modeled as Random Oracle
- Or both!

Non-interactive Zero Knowledge Proof (NIZK)



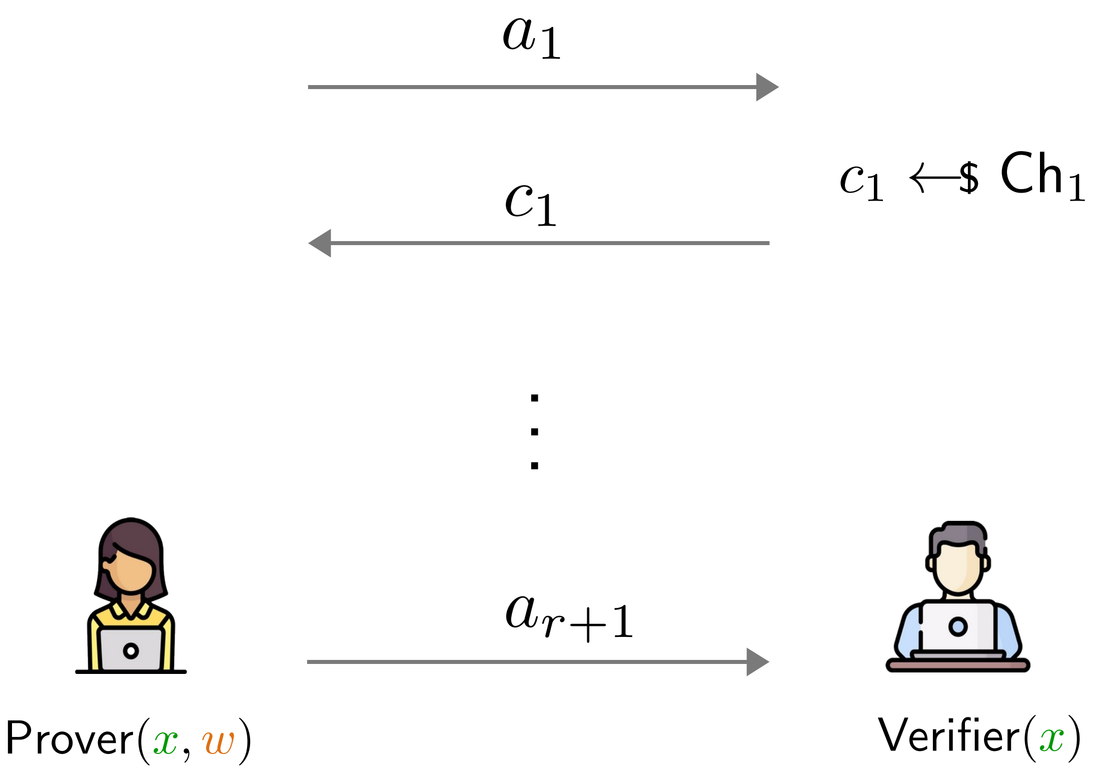
Removing Interactions

- Ideally, Prover should create a one-shot proof string π
- Verifier checks π asynchronously
- Such π is reusable and can be checked by potentially many verifiers

Types of Trusted Setup

- Structured Reference String (SRS)
- Hash function modeled as Random Oracle
- Or both!

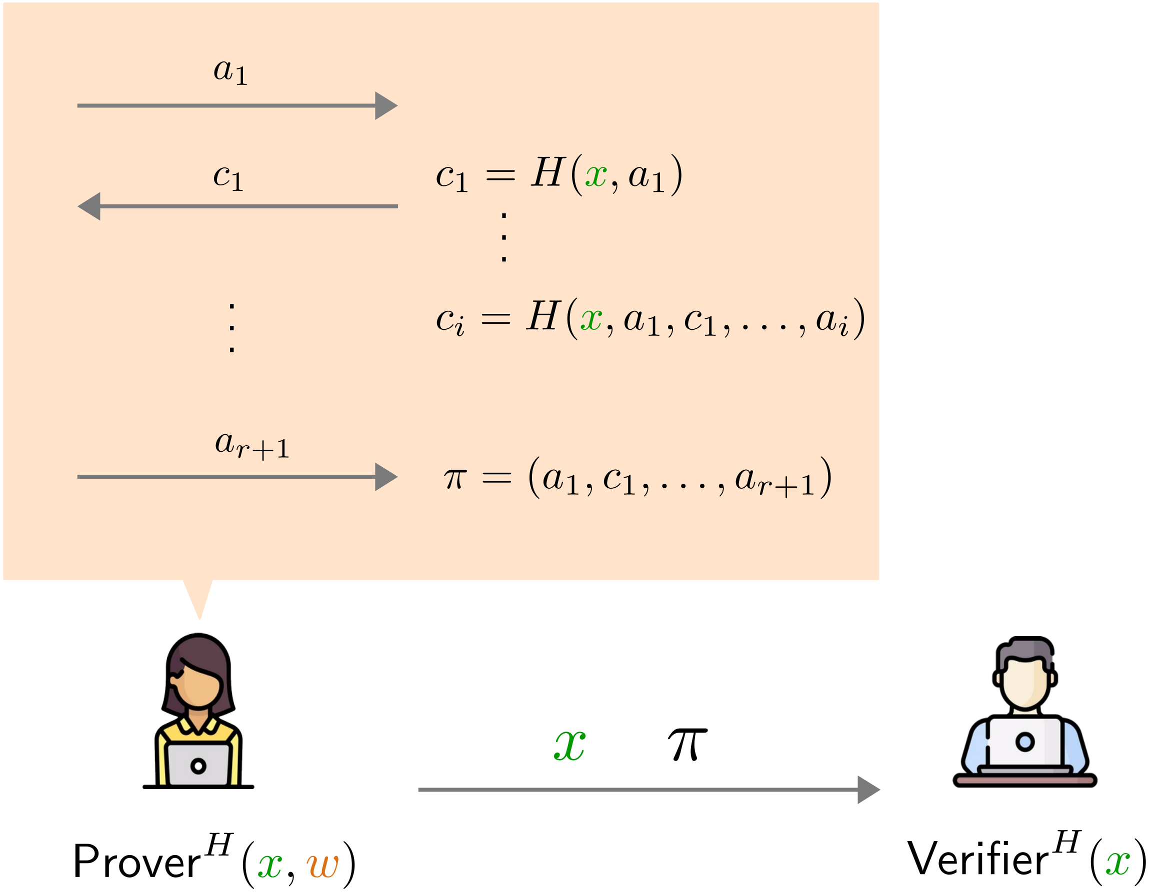
Paradigm of NIZK I: Fiat-Shamir [FS87]



Modular Design of NIZK

- **Step 1.** Construct a "public-coin" interactive protocol
  - Verifier does not require a secret state
  - ZK against semi-honest Verifier (Honest-Verifier ZK)
- **Step 2.** NI Prover and Verifier obtain challenge by locally hashing a partial transcript so far
- Bonus: By hashing the message, FS-NIZK gives rise to a digital signature
- Example: Schnorr/EdDSA, CRYSTALS-Dilithium, PLONK family, Bulletproofs, etc.
- Many modern SNARKs are constructed from (Polynomial) Interactive Oracle Proofs converted to NIZK via Fiat-Shamir [BCS16, CHMMVW19, BFS19, GWC19, CFFQR20,…]

Paradigm of NIZK I: Fiat-Shamir [FS87]



Modular Design of NIZK

- **Step 1.** Construct a "public-coin" interactive protocol
  - Verifier does not require a secret state
  - ZK against semi-honest Verifier (Honest-Verifier ZK)
- **Step 2.** NI Prover and Verifier obtain challenge by locally hashing a partial transcript so far
- Bonus: By hashing the message, FS-NIZK gives rise to a digital signature
- Example: Schnorr/EdDSA, CRYSTALS-Dilithium, PLONK family, Bulletproofs, etc.
- Many modern SNARKs are constructed from (Polynomial) Interactive Oracle Proofs converted to NIZK via Fiat-Shamir [BCS16, CHMMVW19, BFS19, GWC19, CFFQR20,…]

Paradigm of NIZK I: Fiat-Shamir [FS87]

$\xrightarrow{a_1}$

$\xleftarrow{c_1}$ $\quad c_1 = H(vk, a_1, \mathsf{msg})$

$\vdots$

$\vdots \quad c_i = H(vk, a_1, c_1, \ldots, a_i, \mathsf{msg})$

$\xrightarrow{a_{r+1}}$ $\quad \sigma = (a_1, c_1, \ldots, a_{r+1})$

$\xrightarrow{vk,\ \mathsf{msg},\ \sigma}$

$\mathsf{Signer}^H(vk, sk, \mathsf{msg})$ $\quad\quad$ $\mathsf{Verifier}^H(vk, \sigma, \mathsf{msg})$

Modular Design of NIZK

- **Step 1.** Construct a "public-coin" interactive protocol
  - Verifier does not require a secret state
  - ZK against semi-honest Verifier (Honest-Verifier ZK)
- **Step 2.** NI Prover and Verifier obtain challenge by locally hashing a partial transcript so far
- Bonus: By hashing the message, FS-NIZK gives rise to a digital signature
- Example: Schnorr/EdDSA, CRYSTALS-Dilithium, PLONK family, Bulletproofs, etc.
- Many modern SNARKs are constructed from (Polynomial) Interactive Oracle Proofs converted to NIZK via Fiat-Shamir [BCS16, CHMMVW19, BFS19, GWC19, CFFQR20,…]

Paradigm of NIZK I: Fiat-Shamir [FS87]

Interactive Oracle Proof

(No computational assumption)

↓ + Cryptographic Commitment

Interactive Zero Knowledge Proof

(Often only secure against computationally bounded adversaries)

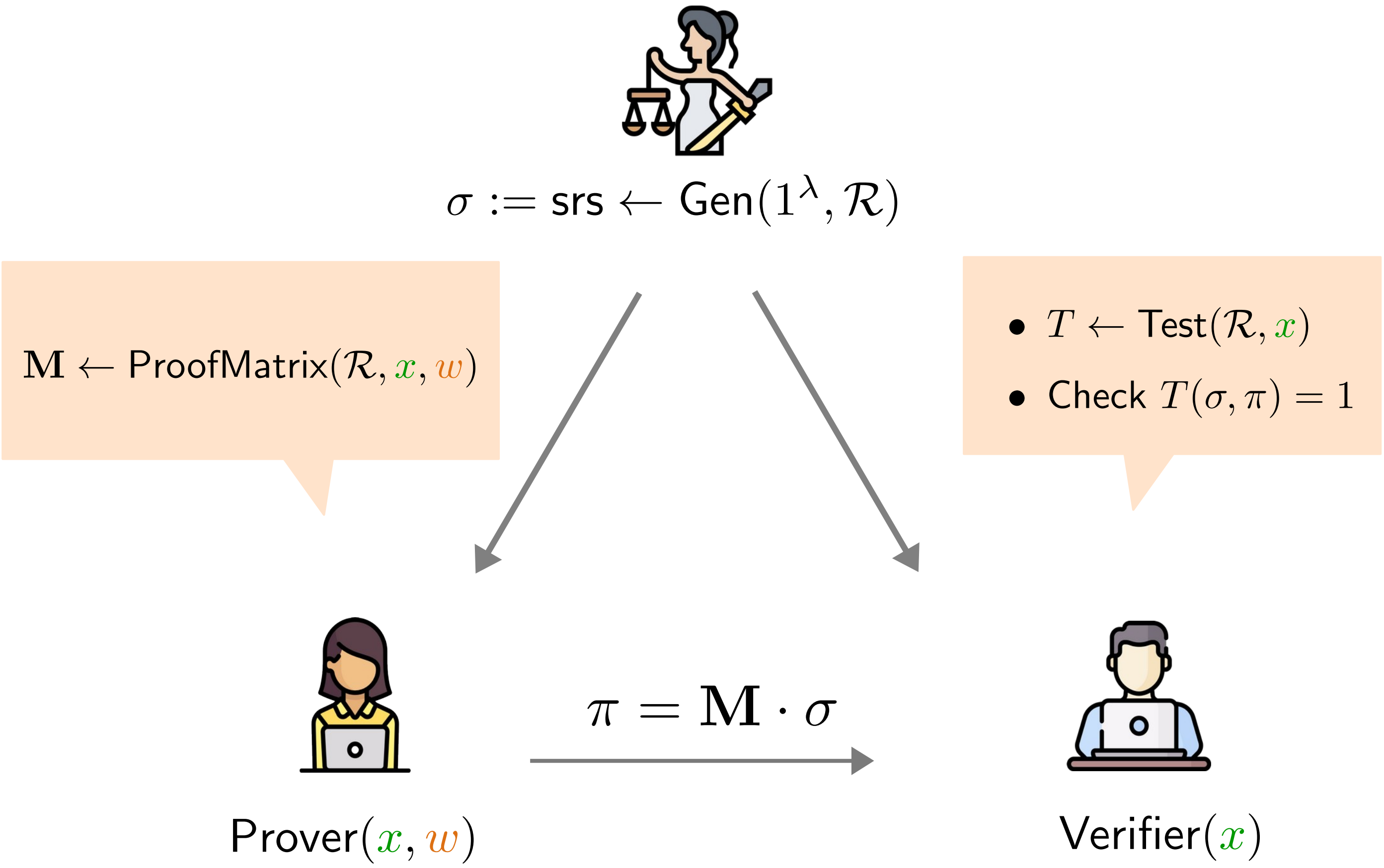
↓ + Fiat-Shamir

Non-interactive Zero Knowledge Proof

Modular Design of NIZK

- **Step 1.** Construct a "public-coin" interactive protocol
  - Verifier does not require a secret state
  - ZK against semi-honest Verifier (Honest-Verifier ZK)
- **Step 2.** NI Prover and Verifier obtain challenge by locally hashing a partial transcript so far
- Bonus: By hashing the message, FS-NIZK gives rise to a digital signature
- Example: Schnorr/EdDSA, CRYSTALS-Dilithium, PLONK family, Bulletproofs, etc.
- Many modern SNARKs are constructed from (Polynomial) Interactive Oracle Proofs converted to NIZK via Fiat-Shamir [BCS16, CHMMVW19, BFS19, GWC19, CFFQR20,…]

Paradigm of NIZK II: Linear Interactive Proofs [GGPR13,BCI+13]



NIZK without Fiat-Shamir

- **Step 1.** srs generator outputs a relation-dependent vector
- **Step 2.** NI Prover applies linear transformation to srs
- **Step 3.** NI Verifier derives a testing function, allowing to check whether correct linear transformation has been applied
- Example: [Groth16]
- Important: Prover and Verifier should never learn internal randomness of Gen; otherwise, malicious prover can easily prove a false statement

Technical Challenges

1) Balancing Generality, Efficiency and Assumptions

2) Advanced Security

3) Interoperability

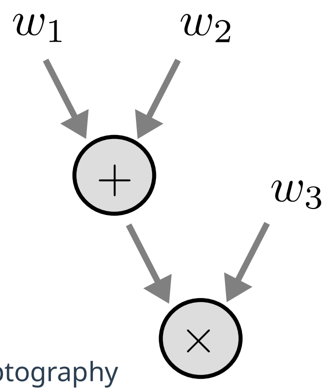
Types of ZKP

General-Purpose ZKP

- Supports arbitrary NP relation R
- Relation is often described using an arithmetic circuit

$$\mathcal{R}_C = \{(x, w) : C(x, w) = 1\}$$

- Pros:
  - Can prove correct execution of *any* program
  - Suitable for verifiable and outsourced computation
- Cons:
  - circuit gets complex for certain non-linear computations
  - E.g., elliptic curve arithmetic, comparison, table lookup, etc.



Specialized ZKP

- Designed for particular type of NP relation R

$$\mathcal{R}_{\text{DL}} = \{(X, w) : X = w \cdot G\}$$

$$\mathcal{R}_{\text{SIS}} = \{(\mathbf{x}, \mathbf{w}) : \mathbf{x} = \mathbf{A}\mathbf{w} \bmod q, \|\mathbf{w}\| \leq \beta\}$$

$$\mathcal{R}_{\text{Lookup}} = \{(\mathbf{x}, \mathbf{w}) : \mathbf{w} \text{ is a subvector of } \mathbf{x}\}$$

- Pros:
  - Can prove and verify designated relations efficiently
  - Sufficient for some useful applications, e.g., proof of correct encryption, distributed key generation, signatures, etc.
- Cons:
  - Requires careful integration with general-purpose ZKP to support more complex statements

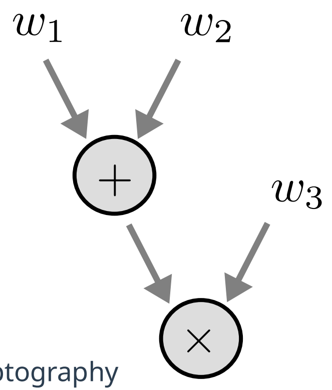
Types of ZKP

General-Purpose ZKP

- Supports arbitrary NP relation R
- Relation is often described using an arithmetic circuit

$$\mathcal{R}_C = \{(x, w) : C(x, w) = 1\}$$

- Pros:
  - Can prove correct execution of *any* program
  - Suitable for verifiable and outsourced computation
- Cons:
  - circuit gets complex for certain non-linear computations
  - E.g., elliptic curve arithmetic, comparison, table lookup, etc.



Specialized ZKP

- Designed for particular type of NP relation R

$$\mathcal{R}_{\mathrm{DL}} = \{(X, w) : X = w \cdot G\}$$

$$\mathcal{R}_{\mathrm{SIS}} = \{(\mathbf{x}, \mathbf{w}) : \mathbf{x} = \mathbf{A}\mathbf{w} \bmod q, \|\mathbf{w}\| \leq \beta\}$$

$$\mathcal{R}_{\mathrm{Lookup}} = \{(\mathbf{x}, \mathbf{w}) : \mathbf{w} \text{ is a subvector of } \mathbf{x}\}$$

- Pros:
  - Can prove and verify designated relations efficiently
  - Sufficient for some useful applications, e.g., proof of correct encryption, distributed key generation, signatures, etc.
- Cons:
  - Requires careful integration with general-purpose ZKP to support more complex statements

Desiderata

Proof Size

- Smaller proof saves storage and communication bandwidth
- Groth16 requires *only 3 group elements* from pairing-friendly curves
- State-of-the-art Polymath [Lip24] and PARI [DMS24] achieve even smaller proof sizes!

Assumptions

- To minimize a trust assumption, SRS should be avoided
- Better alternative: only trust the security of hash function modeled as RO (aka **transparent** setup), e.g., Bulletproofs, Brakedown, STARK, LaBRADOR, MPC/VOLE-in-the-Head, etc.
- Middle-ground solution: allows different parties to update SRS (aka **updatable SRS**) [GKMMM18]

Setup, Prover and Verifier Cost

- **Universal Setup**: Setup outputs SRS once and for all for arbitrary circuits [GKMMM18]

$$\mathsf{srs} \leftarrow \mathsf{Setup}; \mathsf{srs}_C \leftarrow \mathsf{Derive}(\mathsf{srs}, C)$$

- Verifier sub-linear in $|C|$
- Prover time linear in #non-linear gates

Scalability

- How can we prove a large statement efficiently?
  - **Proof Aggregation**: aggregate many, asynchronously generated proofs, e.g., SnarkPack
  - **Incrementally Verifiable Computation [Valiant08]**: succinct proof of incremental computations via recursion or folding, e.g., Halo2, Nova, etc.

Desiderata

Proof Size

- Smaller proof saves storage and communication bandwidth
- Groth16 requires *only 3 group elements* from pairing-friendly curves
- State-of-the-art Polymath [Lip24] and PARI [DMS24] achieve even smaller proof sizes!

Assumptions

- To minimize a trust assumption, SRS should be avoided
- Better alternative: only trust the security of hash function modeled as RO (aka **transparent** setup), e.g., Bulletproofs, Brakedown, STARK, LaBRADOR, MPC/VOLE-in-the-Head, etc.
- Middle-ground solution: allows different parties to update SRS (aka **updatable SRS**) [GKMMM18]

Setup, Prover and Verifier Cost

- **Universal Setup**: Setup outputs SRS once and for all for arbitrary circuits [GKMMM18]

$$\mathsf{srs} \leftarrow \mathsf{Setup}; \mathsf{srs}_C \leftarrow \mathsf{Derive}(\mathsf{srs}, C)$$

- Verifier sub-linear in $|C|$
- Prover time linear in #non-linear gates

Scalability

- How can we prove a large statement efficiently?
  - **Proof Aggregation**: aggregate many, asynchronously generated proofs, e.g., SnarkPack
  - **Incrementally Verifiable Computation [Valiant08]**: succinct proof of incremental computations via recursion or folding, e.g., Halo2, Nova, etc.

Desiderata

Proof Size

- Smaller proof saves storage and communication bandwidth
- Groth16 requires *only 3 group elements* from pairing-friendly curves
- State-of-the-art Polymath [Lip24] and PARI [DMS24] achieve even smaller proof sizes!

Assumptions

- To minimize a trust assumption, SRS should be avoided
- Better alternative: only trust the security of hash function modeled as RO (aka **transparent** setup), e.g., Bulletproofs, Brakedown, STARK, LaBRADOR, MPC/VOLE-in-the-Head, etc.
- Middle-ground solution: allows different parties to update SRS (aka **updatable SRS**) [GKMMM18]

Setup, Prover and Verifier Cost

- **Universal Setup**: Setup outputs SRS once and for all for arbitrary circuits [GKMMM18]

$$\mathsf{srs} \leftarrow \mathsf{Setup}; \mathsf{srs}_C \leftarrow \mathsf{Derive}(\mathsf{srs}, C)$$

- Verifier sub-linear in $|C|$
- Prover time linear in #non-linear gates

Scalability

- How can we prove a large statement efficiently?
  - **Proof Aggregation**: aggregate many, asynchronously generated proofs, e.g., SnarkPack
  - **Incrementally Verifiable Computation [Valiant08]**: succinct proof of incremental computations via recursion or folding, e.g., Halo2, Nova, etc.

Desiderata

Proof Size

- Smaller proof saves storage and communication bandwidth
- Groth16 requires *only 3 group elements* from pairing-friendly curves
- State-of-the-art Polymath [Lip24] and PARI [DMS24] achieve even smaller proof sizes!

Assumptions

- To minimize a trust assumption, SRS should be avoided
- Better alternative: only trust the security of hash function modeled as RO (aka transparent setup), e.g., Bulletproofs, Brakedown, STARK, LaBRADOR, MPC/VOLE-in-the-Head, etc.
- Middle-ground solution: allows different parties to update SRS (aka updatable SRS) [GKMMM18]

Setup, Prover and Verifier Cost

- Universal Setup: Setup outputs SRS once and for all for arbitrary circuits [GKMMM18]

$$\mathsf{srs} \leftarrow \mathsf{Setup}; \mathsf{srs}_C \leftarrow \mathsf{Derive}(\mathsf{srs}, C)$$

- Verifier sub-linear in $|C|$
- Prover time linear in #non-linear gates

Scalability

- How can we prove a large statement efficiently?
  - **Proof Aggregation**: aggregate many, asynchronously generated proofs, e.g., SnarkPack
  - **Incrementally Verifiable Computation [Valiant08]**: succinct proof of incremental computations via recursion or folding, e.g., Halo2, Nova, etc.

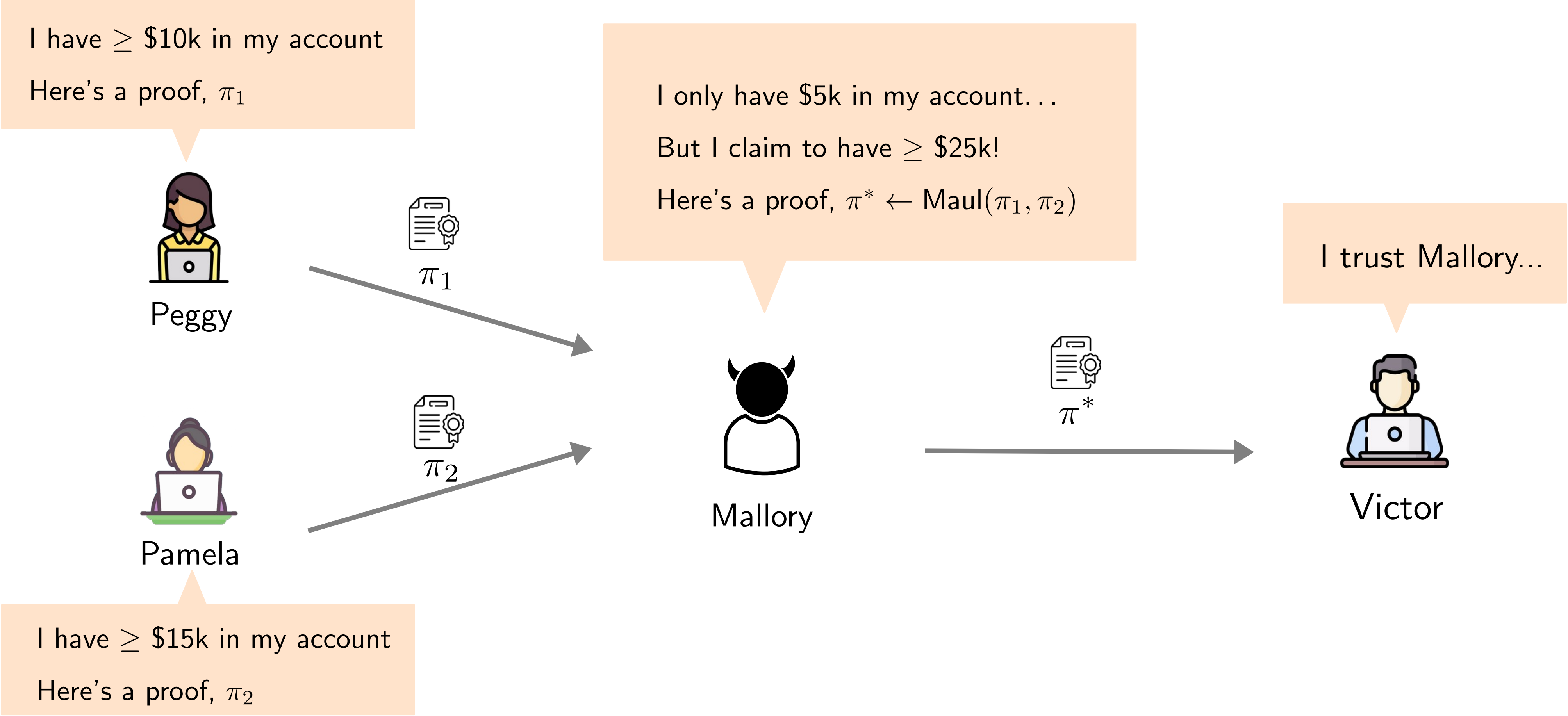
Technical Challenges

1) Balancing Generality, Efficiency and Assumptions

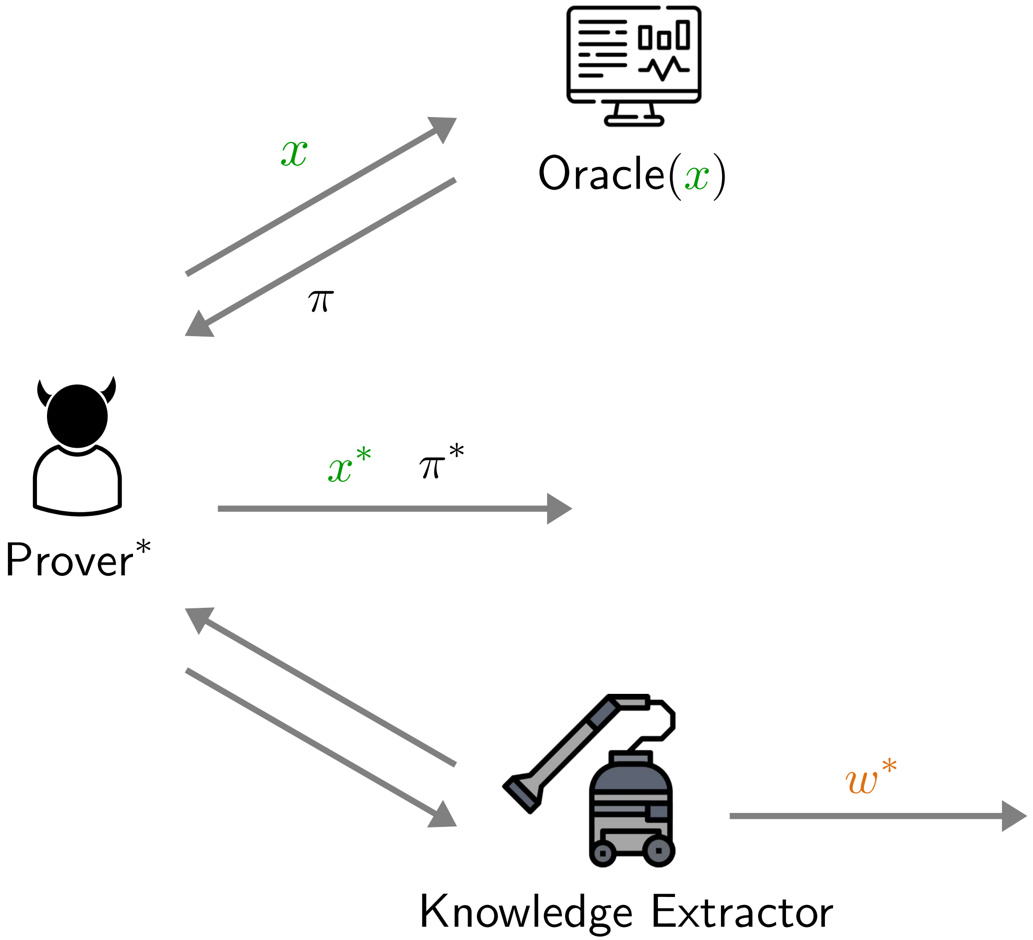
2) Advanced Security

3) Interoperability

ZK and Knowledge Soundness are not Enough: Malleability Attacks



Combined Notion: Simulation-Extractability [Sah99]



SIM-EXT Security

1. Prover* obtains fresh proof from Oracle
2. Prover* outputs “forgery” (x^*, π^*)
3. If (x^*, π^*) is accepting and not recorded by Oracle, then Prover* must know the corresponding witness w^*

- Intuitively, SIM-EXT guarantees **non-malleability:** a cheating prover cannot maul existing proofs to create a new one, without knowing a valid witness
- Cf. (S)EUF-CMA for signature and IND-CCA for PKE
- Crucial property NIZK should satisfy if used as a subroutine of another protocol
- Many practical NIZK schemes turn out to be SIM-EXT [GKKNZ22] [GOPTT22] [DG23] [FFKR23] [KPT23] [Lib24] [FFR24]
- Some schemes satisfy UC security [Canetti01] accepting some idealized setup [CF24] [BFKT24]

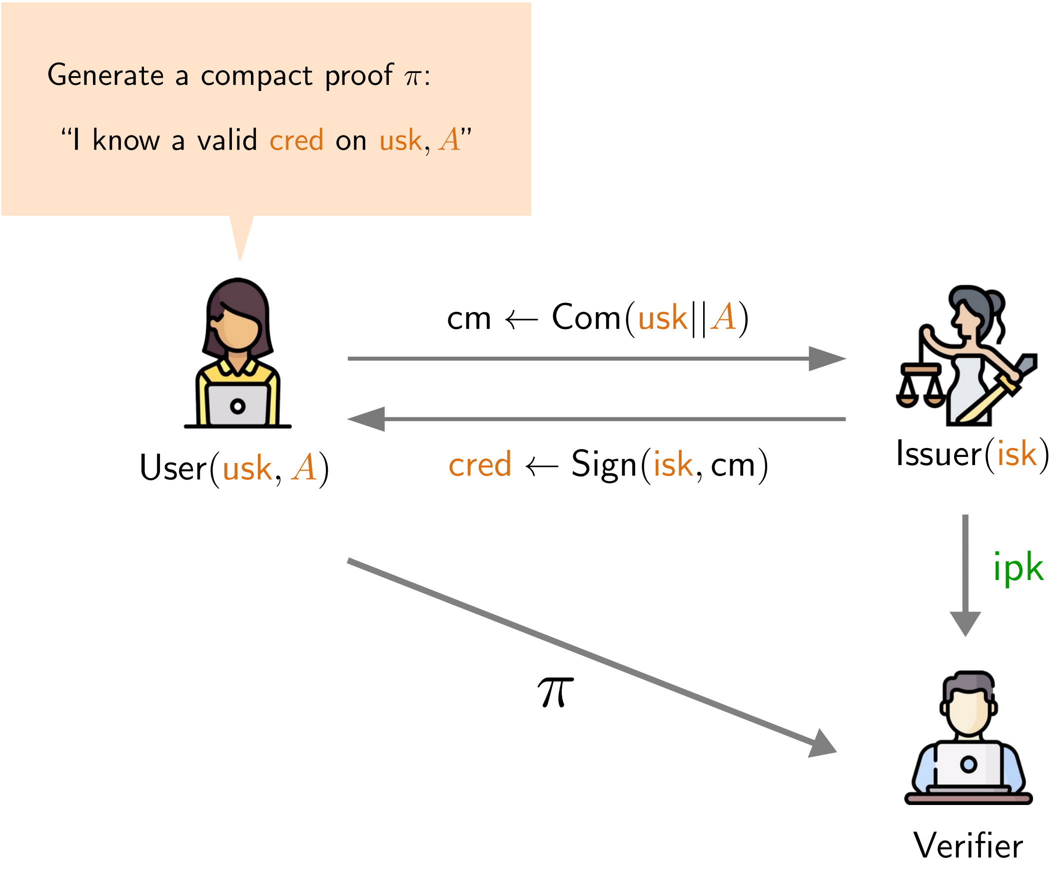
Technical Challenges

1) Balancing Generality, Efficiency and Assumptions

2) Advanced Security

3) Interoperability

Example: Anonymous Credentials (High Level)

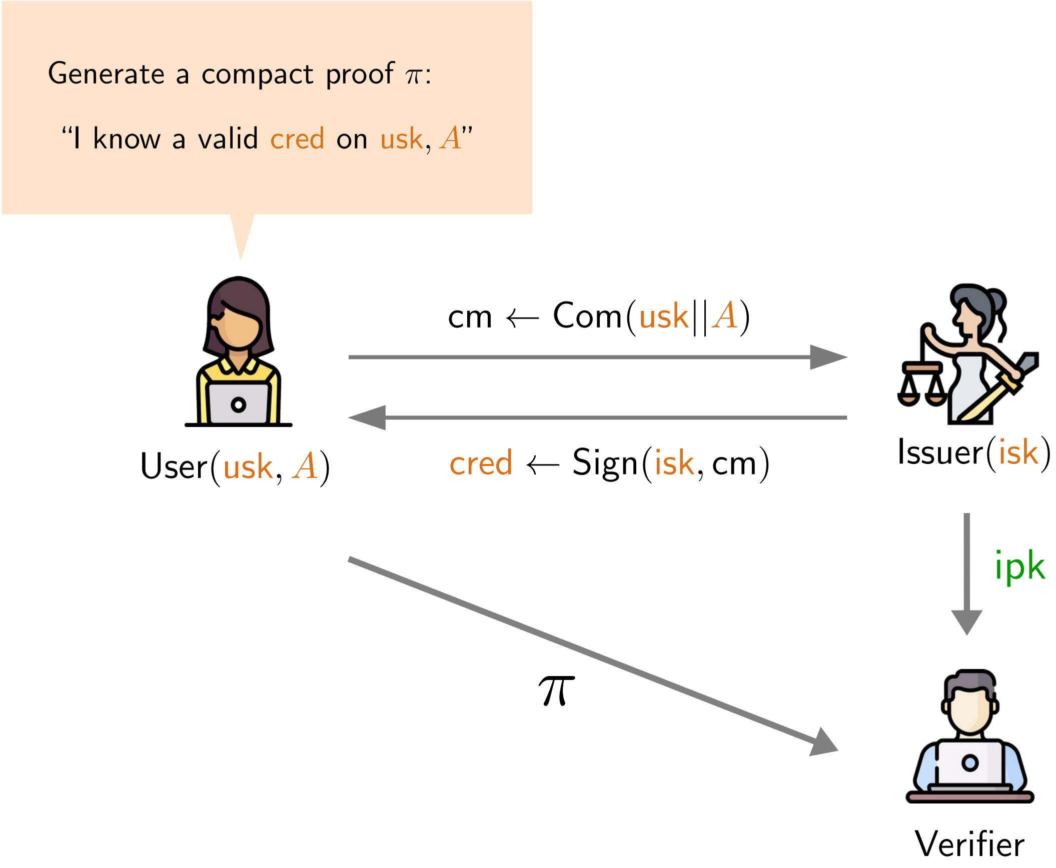


Protocol

- Issuer initially binds attributes and usk to secret credentials
- The owner of attributes produces a **proof string** in the form of ZKP
- By examining the proof string, Verifier gets convinced that User has valid attributes signed by Issuer
- Thanks to ZKP, the proof string only leaks minimum info about Prover's identity
- E.g., Verifier learns "User is => 21 years old" but nothing else

- isk: issuer secret key
- ipk: issuer public key
- usk: user secret key
- A: user attributes

Example: Anonymous Credentials (High Level)



Interoperability

- Central ZKP for AC: Proof-of-Knowledge of valid signature
- Verification algorithms of widely deployed signatures, e.g, RSA-PSS, ECDSA, EdDSA, etc. are not ZK-friendly
- Two directions:
  - Design more specialized and efficient ZKP for existing standardized schemes to retain interoperability
  - Design and standardize "ZK-friendly" primitives: Cf. BBS(+) signature

- isk: issuer secret key
- ipk: issuer public key
- usk: user secret key
- A: user attributes

Takeaways

- ZKP allows Prover to prove the knowledge of a secret, while Verifier learns nothing about the secret
- Basic Security Properties: Knowledge Soundness and Zero Knowledge
- What kind statement do you want to prove?
  - General-purpose ZKP, Specialized ZKP, Composition of both
- Which setup assumption is suitable for deployment?
  - Trusted, Transparent, Updatable, …
- What should you optimize?
  - Proof Size, Assumptions, Setup/Prover/Verifier Costs, Scalability.
- Advanced Security: Does the application need SIM-EXT or UC security?
- Interoperability: Standardize ZK-friendly primitives, or design standardization-friendly ZK

Credits: icons have been retrieved from Flaticon.com

Zero-Knowledge Proofs: Technical Challenges, Applications, and Real-world Deployment

NIST Workshop on Privacy-Enhancing Cryptography

Tjerand Silde & Akira Takahashi, September 26 – 2024

Norwegian University of Science and Technology

Content

Introduction to ZKP

Technical Challenges

Real-World Applications

Insights from ZKP Workshop

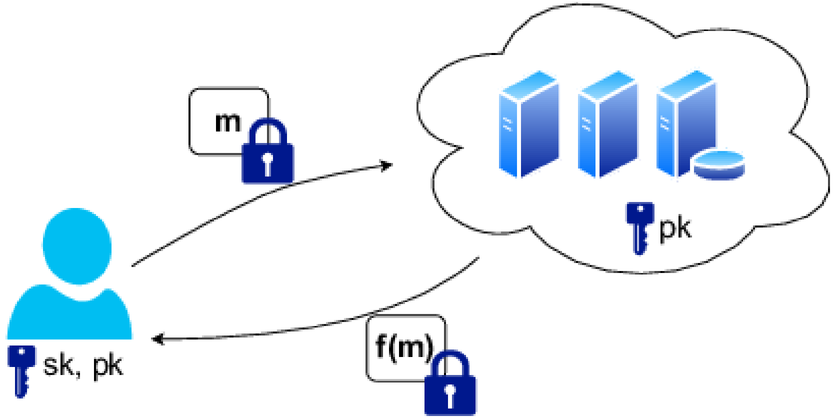
Resources and Standards

Verifiable and Outsourced Computation

Ensure that computation is conducted properly (server is the prover)

Might include secret data or algorithms, but does not have to do so

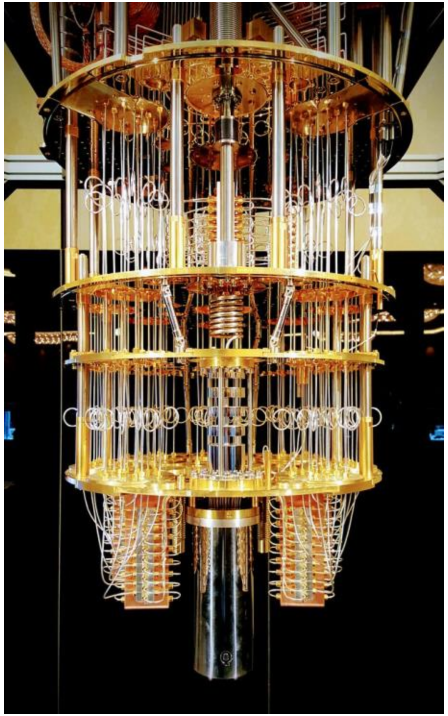
Use ZKP for compliance



Efficient (Post-Quantum) Digital Signatures

Quantum computers can break schemes based on factoring and DLOG

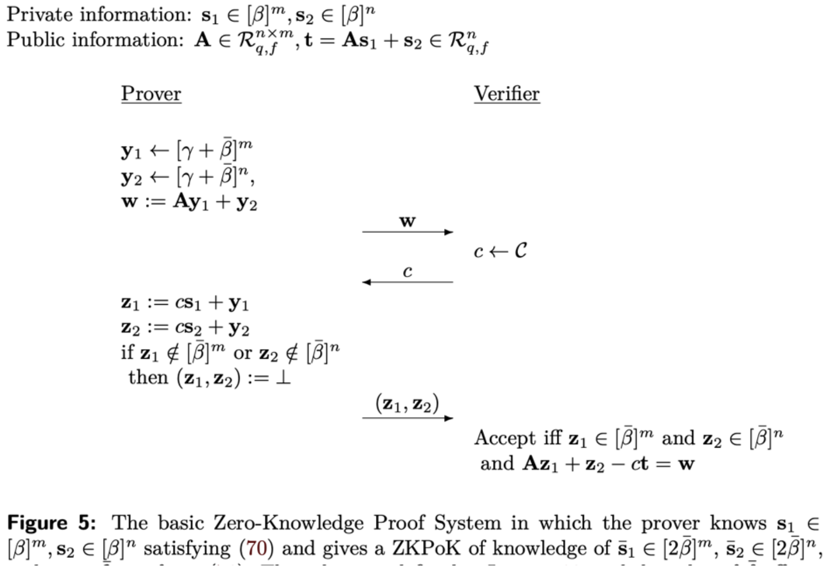
Can design signature schemes from zero-knowledge proofs and the Fiat-Shamir transform



Efficient (Post-Quantum) Digital Signatures

Dilithium is a NIZK based on the quantum-safe LWE/SIS-problems

Follows a similar structure as Schnorr-signatures for DLOG



Private information: $\mathbf{s}_1 \in [\beta]^m, \mathbf{s}_2 \in [\beta]^n$
Public information: $\mathbf{A} \in \mathcal{R}_{q,f}^{n \times m}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \in \mathcal{R}_{q,f}^n$

Prover		Verifier
$\mathbf{y}_1 \leftarrow [\gamma + \bar{\beta}]^m$ $\mathbf{y}_2 \leftarrow [\gamma + \bar{\beta}]^n$, $\mathbf{w} := \mathbf{A}\mathbf{y}_1 + \mathbf{y}_2$		
	$\xrightarrow{\mathbf{w}}$	
		$c \leftarrow \mathcal{C}$
	$\xleftarrow{c}$	
$\mathbf{z}_1 := c\mathbf{s}_1 + \mathbf{y}_1$ $\mathbf{z}_2 := c\mathbf{s}_2 + \mathbf{y}_2$ if $\mathbf{z}_1 \notin [\bar{\beta}]^m$ or $\mathbf{z}_2 \notin [\bar{\beta}]^n$ then $(\mathbf{z}_1, \mathbf{z}_2) := \perp$		
	$\xrightarrow{(\mathbf{z}_1, \mathbf{z}_2)}$	
		Accept iff $\mathbf{z}_1 \in [\bar{\beta}]^m$ and $\mathbf{z}_2 \in [\bar{\beta}]^n$ and $\mathbf{A}\mathbf{z}_1 + \mathbf{z}_2 - c\mathbf{t} = \mathbf{w}$

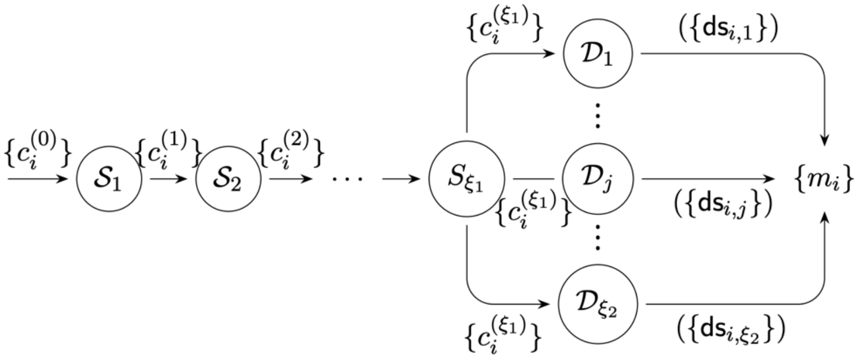
Figure 5: The basic Zero-Knowledge Proof System in which the prover knows $\mathbf{s}_1 \in [\beta]^m, \mathbf{s}_2 \in [\beta]^n$ satisfying (70) and gives a ZKPoK of knowledge of $\bar{\mathbf{s}}_1 \in [2\bar{\beta}]^m$, $\bar{\mathbf{s}}_2 \in [2\bar{\beta}]^n$,

https://eprint.iacr.org/2024/1287.pdf

Proof Systems in Electronic Voting

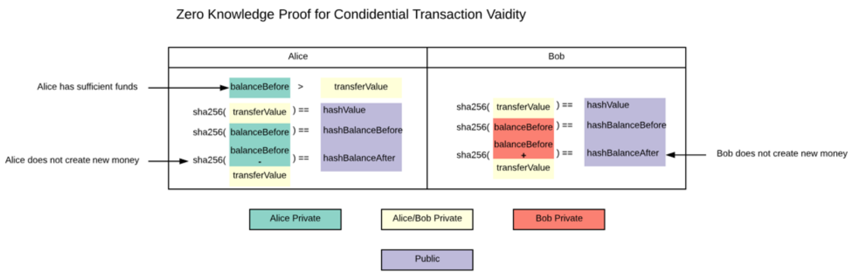
Need to break the connection between votes and voters by shuffling

Ensure correct encryption and decryption of votes

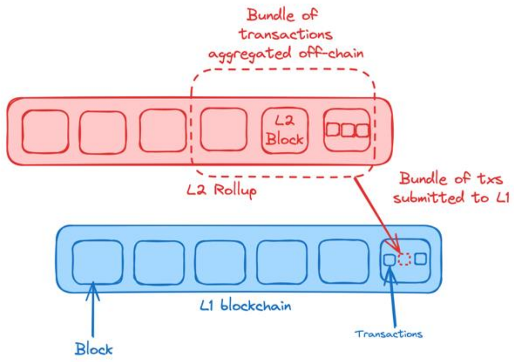


Blockchain Rollup and Private Transactions

For privacy: encrypt to make transactions private, use ZKP to ensure correctness and compliance to bank laws



For efficiency: batch many transactions together and prove that all were correct without checking each



Content

Introduction to ZKP

Technical Challenges

Real-World Applications

Insights from ZKP Workshop

Resources and Standards

International Centre for Mathematical Sciences

Sponsors and Funders:

ICMS Workshop on Foundations and Applications of Zero-Knowledge Proofs

A one-week workshop about ZKPs: going from the basics to some of the most advanced applications.

All the slides and recordings are available online.

Organized w/ Elizabeth Crites and Markulf Kolweiss.

icms.org.uk/ZeroKnowledgeProofs

Speakers

Jonathan Katz (UMD)
Michele Ciampi (UoE)
Carsten Baum (DTU)
Peter Scholl (AU)
Carla Rafols (UPF)

Arantxa Zapico (Ethereum)
Anca Nitulescu (IOG)
Lisa Kohl (CWI Amsterdam)
Ngoc Khanh Nguyen (KCL)
Dario Fiore (IMDEA)

Topics

- Introduction to ZKPs and their Security
- Sigma-Protocols and their Applications
- MPC-in-the-Head Techniques for ZKP and Signatures
- Group/pairing-based zkSNARK Constructions
- Polynomial Commitments for zkSNARKs
- Lattice-Based ZKPs and Polynomial Commitments
- ZKPs for Blockchain Applications
- ZKP for Machine Learning and Verifiable Computation

Lessons Learned

Recent advances in ZKP rely heavily on earlier works, and it is worthwhile to go in-depth on the foundations.

ZKP is a fast-moving field, and several invited speakers talked about new constructions published after we reached out.

ZKP has until recently been considered a theoretical field, but nowadays we see new and efficient implementations every week.

New constructions are quite complex, and it might be hard to keep up with the technical details and get a proper overview.

Content

Introduction to ZKP

Technical Challenges

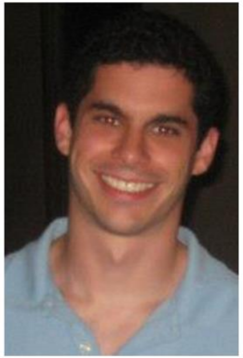
Real-World Applications

Insights from ZKP Workshop

Resources and Standards

Zero-Knowledge Proofs MOOC

Instructors

				
Dan Boneh	Shafi Goldwasser	Dawn Song	Justin Thaler	Yupeng Zhang
Stanford	UC Berkeley	UC Berkeley	Georgetown University	Texas A&M University

zk-learning.org

ZKProof Standards

About ZKProof

ZKProof is an open-industry academic initiative that seeks to mainstream zero-knowledge proof (ZKP) cryptography through an inclusive, community-driven standardization process that focuses on interoperability and security.

Annually-held ZKProof workshops, attended by world-renowned cryptographers, practitioners and industry leaders, are the optimal forum for discussing new proposals, reviewing cutting edge projects and advancing a community reference document that will ultimately serve as a trusted specification for the implementation of ZKP schemes and protocols.

zkproof.org

Blog-posts by Matthew Green

Matthew Green in fundamentals   November 27, 2014   4,100 Words

Zero Knowledge Proofs: An illustrated primer

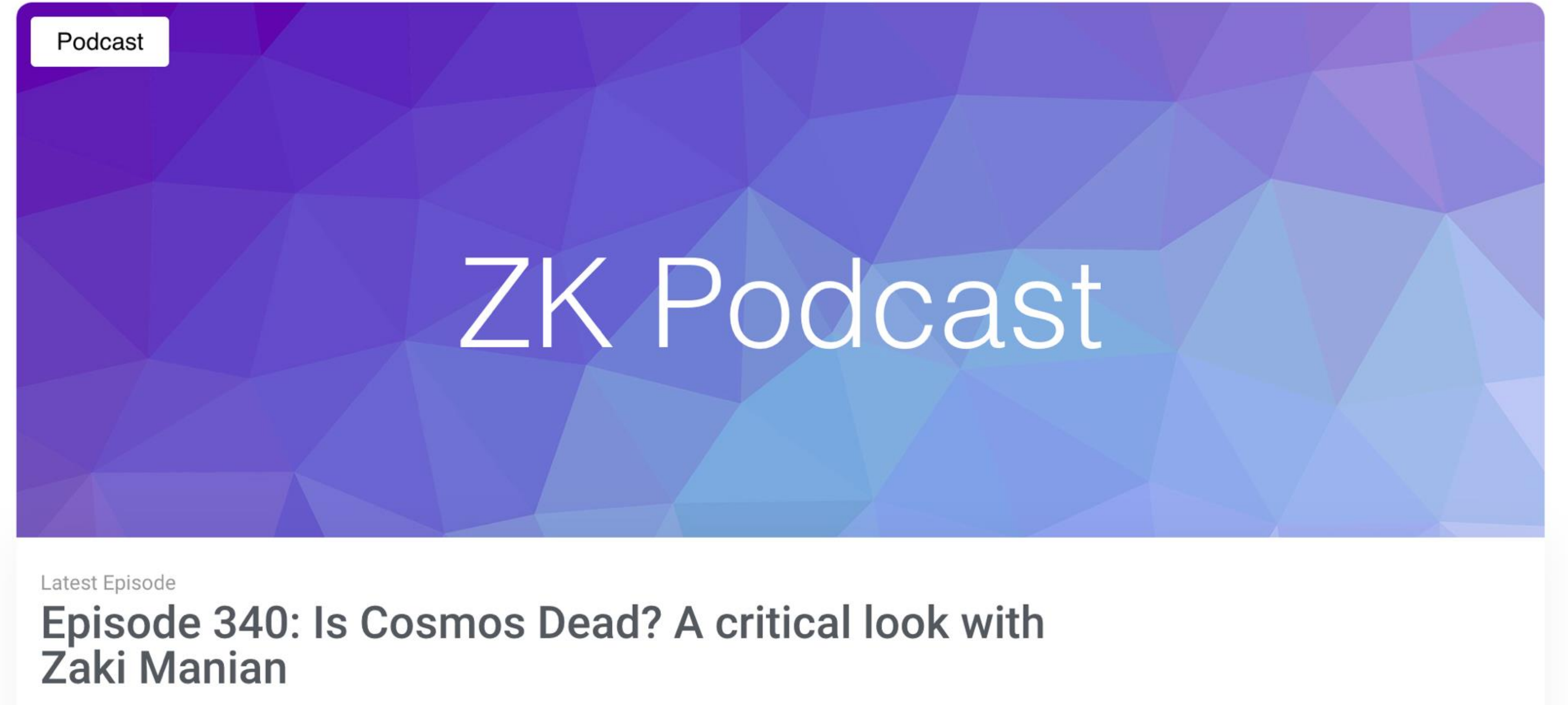
One of the best things about modern cryptography is the beautiful terminology. You could start any number of punk bands (or Tumblrs) named after cryptography terms like 'hard-core predicate', 'trapdoor function', ' or 'impossible differential cryptanalysis'. And of course, I haven't even mentioned the one term that surpasses all of these. That term is '*zero knowledge*'.

Matthew Green

I'm a cryptographer and professor at Johns Hopkins University. I've designed

blog.cryptographyengineering.com/2014/11/27/zero-knowledge-proofs-illustrated-primer

Zero-Knowledge Podcast



zeroknowledge.fm

Zero-Knowledge Summit

zksummit.com

DARPA-Funded ZKP Research

Generating Zero-Knowledge Proofs for Defense Capabilities

Program aims to advance method for making public statements without compromising sensitive underlying information

OUTREACH@DARPA.MIL
7/18/2019

darpa.mil/news-events/2019-07-18

ZKP in EU Digital Identity Wallet

Cryptographers' Feedback on the EU Digital Identity's ARF

Carsten Baum
Technical University of Denmark

Olivier Blazy
École Polytechnique

Jan Camenisch
Dfinity

Jaap-Henk Hoepman
Karlstad University
& Radboud University

Eysa Lee
Brown University

Anja Lehmann
Hasso-Plattner-Institute,
University of Potsdam

Anna Lysyanskaya
Brown University

René Mayrhofer
Johannes Kepler University Linz

Hart Montgomery*

Ngoc Khanh Nguyen
King's College London

Bart Preneel
KU Leuven

abhi shelat
Northeastern University

Daniel Slamanig
Universität der Bundeswehr München

Stefano Tessaro
University of Washington

Søren Eller Thomsen
Partisia

Carmela Troncoso
EPFL

June 2024

github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/discussions/211

Least Authority

Building the Zero-Knowledge Community: Engagement, Events, and Advocacy

September 18, 2024 Least Authority Team

leastauthority.com/blog/building-the-zero-knowledge-community-engagement-events-and-advocacy

zkSecurity

ZKSECURITY

Audits Development

We're experts

in ZKP, MPC, FHE, and advanced cryptography...

zksecurity.xyz

Trail of Bits

Serving up zero-knowledge proofs

POST   FEBRUARY 19, 2021   4 COMMENTS

By Jim Miller, Senior Cryptography Analyst

Zero-knowledge (ZK) proofs are gaining popularity, and exciting new applications for this technology are emerging, particularly in the blockchain space. So we'd like to shine a spotlight on an interesting source of implementation bugs that we've seen—the Fiat Shamir transformation.

blog.trailofbits.com/2021/02/19/serving-up-zero-knowledge-proofs

Workshop at Simons Institute

Cryptography 10 Years Later: Obfuscation, Proof Systems, and Secure Computation

Monday, May 19 – Friday, Aug. 15, 2025

simons.berkeley.edu/programs/cryptography-10-years-later-obfuscation-proof-systems-secure-computation

Thank you! Questions?

NIST Workshop on Privacy-Enhancing Cryptography

Tjerand Silde & Akira Takahashi, September 26 - 2024

Norwegian University of Science and Technology