Community reference material for zero-knowledge proofs

(A brief update and a call for participation)

Daniel Benarroch (QEDIT) and Luís Brandão (NIST)

Presented at the ZKProof Community Event
October 29, 2019 @ Amsterdam, Netherlands

Outline

Outline

Goals of the presentation

- Inform about the ZKProof Community Reference
- Motivate collaboration

Outline 1

The ZKProof Community Reference

Helps onboard newcomers

Informs practitioners

Promote best practices

> D2. **Purpose.** For example: *The purpose of developing the ZKProof reference document is to provide, within the principles laid out by the ZKProof charter, a reference for the development of zero-knowledge-proof technology that is secure, practical and interoperable.*
>
> D3. **Aim.** For example: *The aim of the document is to consolidate the reference material developed in collaborative processes during the ZKProof workshops. The document intends to be accessible to a large audience, which includes the general public, the media, the industry, developers and cryptographers.*
>
> D4. **Scope.** For example: *The document intends to cover material relevant for the development of secure, practical and interoperable technology, as identified in the* purpose. *The document will also elaborate on introductory concepts or works, as a way to enable an easier understanding of more advanced techniques. When a focus is chosen from several alternative options, the document should try to include a rationale describing, if possible, comparative advantages, disadvantages and applicability. However, the document does not intend to be a thorough survey about ZKPs, and does not need to cover every conceivable scenario.*

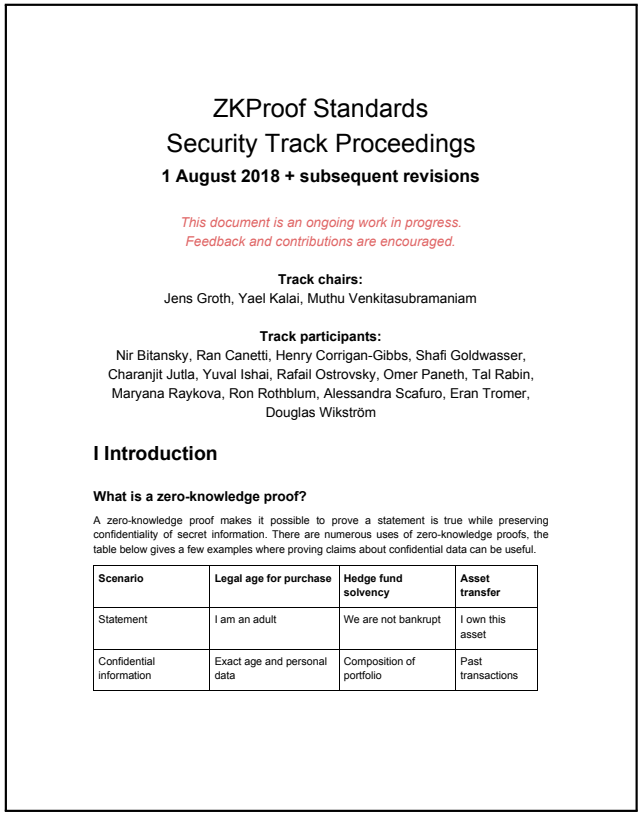
Excerpt from the "NIST comments on the initial ZKProof documentation" (April 06, 2019)

Documentation from the get-go

- First workshop fully dedicated to compile knowledge into 3 documents
- Had several chairs, and many contributors
- First steps towards a reference - building infrastructure

Security Track

- Zero-knowledge proofs definitions (and variants)
- Security assumptions (and their validity)
- Syntax and frameworks for building ZKPs (GKR, MPC-in-the-head, Bulletproofs, pairing-based, IOPs)
- Nuances for understanding ZKPs



ZKProof Standards
Security Track Proceedings
1 August 2018 + subsequent revisions

This document is an ongoing work in progress.
Feedback and contributions are encouraged.

Track chairs:
Jens Groth, Yael Kalai, Muthu Venkitasubramaniam

Track participants:
Nir Bitansky, Ran Canetti, Henry Corrigan-Gibbs, Shafi Goldwasser, Charanjit Jutla, Yuval Ishai, Rafail Ostrovsky, Omer Paneth, Tal Rabin, Maryana Raykova, Ron Rothblum, Alessandra Scafuro, Eran Tromer, Douglas Wikström

I Introduction

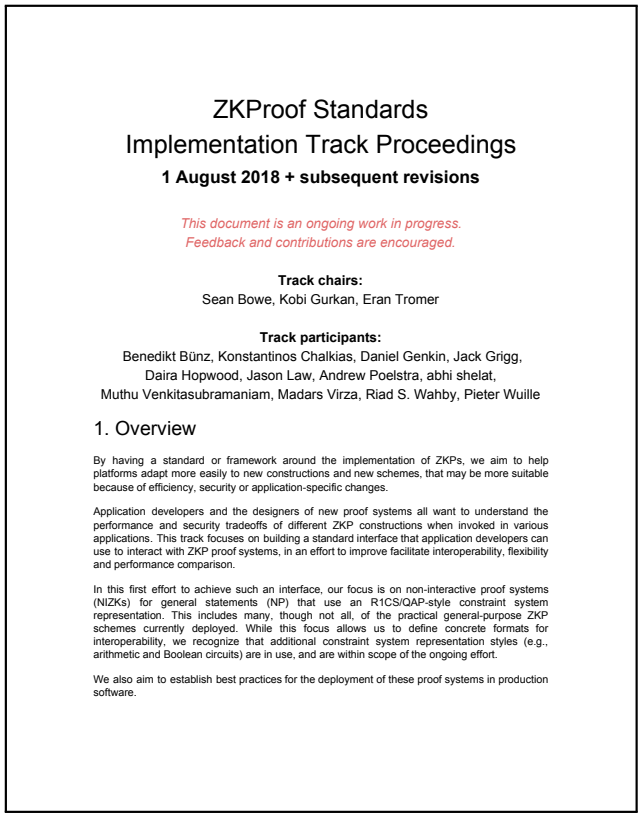
What is a zero-knowledge proof?

A zero-knowledge proof makes it possible to prove a statement is true while preserving confidentiality of secret information. There are numerous uses of zero-knowledge proofs, the table below gives a few examples where proving claims about confidential data can be useful.

Scenario	**Legal age for purchase**	**Hedge fund solvency**	**Asset transfer**
Statement	I am an adult	We are not bankrupt	I own this asset
Confidential information	Exact age and personal data	Composition of portfolio	Past transactions

Implementation Track

- Abstracted model - front-ends / back-ends
- Interoperability types (proof, systems, frameworks, etc...)
- Benchmarking of schemes
- SRS generation (secure MPC)
- DSLs, APIs, Formats
- Secure implementations, trust and correctness



ZKProof Standards
Implementation Track Proceedings
1 August 2018 + subsequent revisions

This document is an ongoing work in progress.
Feedback and contributions are encouraged.

Track chairs:
Sean Bowe, Kobi Gurkan, Eran Tromer

Track participants:
Benedikt Bünz, Konstantinos Chalkias, Daniel Genkin, Jack Grigg,
Daira Hopwood, Jason Law, Andrew Poelstra, abhi shelat,
Muthu Venkitasubramaniam, Madars Virza, Riad S. Wahby, Pieter Wuille

1. Overview

By having a standard or framework around the implementation of ZKPs, we aim to help platforms adapt more easily to new constructions and new schemes, that may be more suitable because of efficiency, security or application-specific changes.

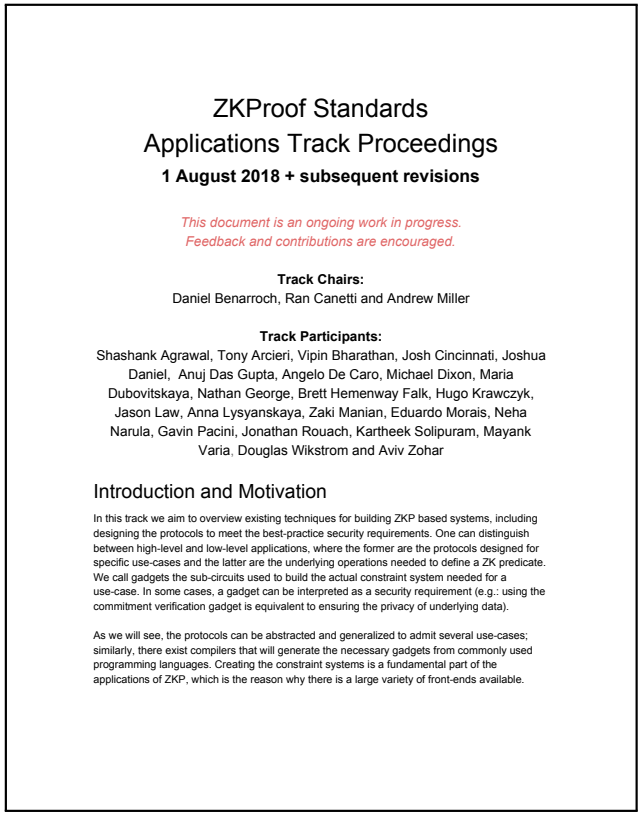
Application developers and the designers of new proof systems all want to understand the performance and security tradeoffs of different ZKP constructions when invoked in various applications. This track focuses on building a standard interface that application developers can use to interact with ZKP proof systems, in an effort to improve facilitate interoperability, flexibility and performance comparison.

In this first effort to achieve such an interface, our focus is on non-interactive proof systems (NIZKs) for general statements (NP) that use an R1CS/QAP-style constraint system representation. This includes many, though not all, of the practical general-purpose ZKP schemes currently deployed. While this focus allows us to define concrete formats for interoperability, we recognize that additional constraint system representation styles (e.g., arithmetic and Boolean circuits) are in use, and are within scope of the ongoing effort.

We also aim to establish best practices for the deployment of these proof systems in production software.

Applications Track

- Gadget libraries (commitments, signatures, encryption, etc...)
- Use-cases exploration
  - Identity
  - Asset Transfers
  - Regulation compliance
- “baby”protocols for use-cases
- Best practices for building prooof statements (predicates)



ZKProof Standards

Applications Track Proceedings

1 August 2018 + subsequent revisions

This document is an ongoing work in progress.
Feedback and contributions are encouraged.

Track Chairs:
Daniel Benarroch, Ran Canetti and Andrew Miller

Track Participants:
Shashank Agrawal, Tony Arcieri, Vipin Bharathan, Josh Cincinnati, Joshua Daniel, Anuj Das Gupta, Angelo De Caro, Michael Dixon, Maria Dubovitskaya, Nathan George, Brett Hemenway Falk, Hugo Krawczyk, Jason Law, Anna Lysyanskaya, Zaki Manian, Eduardo Morais, Neha Narula, Gavin Pacini, Jonathan Rouach, Kartheek Solipuram, Mayank Varia, Douglas Wikstrom and Aviv Zohar

Introduction and Motivation

In this track we aim to overview existing techniques for building ZKP based systems, including designing the protocols to meet the best-practice security requirements. One can distinguish between high-level and low-level applications, where the former are the protocols designed for specific use-cases and the latter are the underlying operations needed to define a ZK predicate. We call gadgets the sub-circuits used to build the actual constraint system needed for a use-case. In some cases, a gadget can be interpreted as a security requirement (e.g.: using the commitment verification gadget is equivalent to ensuring the privacy of underlying data).

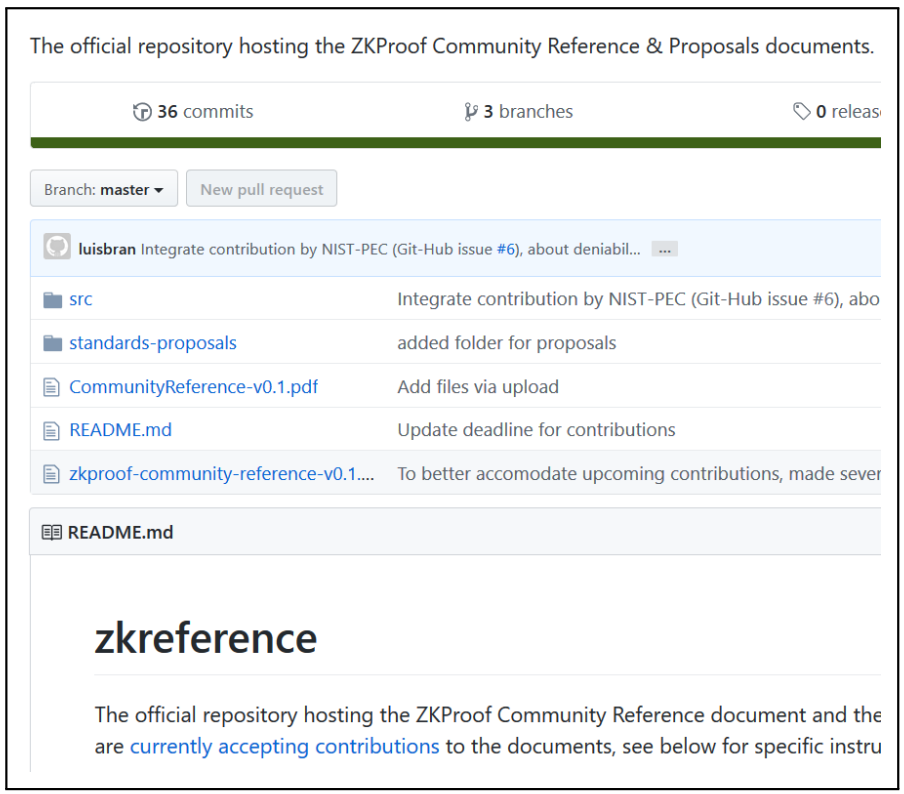
As we will see, the protocols can be abstracted and generalized to admit several use-cases; similarly, there exist compilers that will generate the necessary gadgets from commonly used programming languages. Creating the constraint systems is a fundamental part of the applications of ZKP, which is the reason why there is a large variety of front-ends available.

Current state

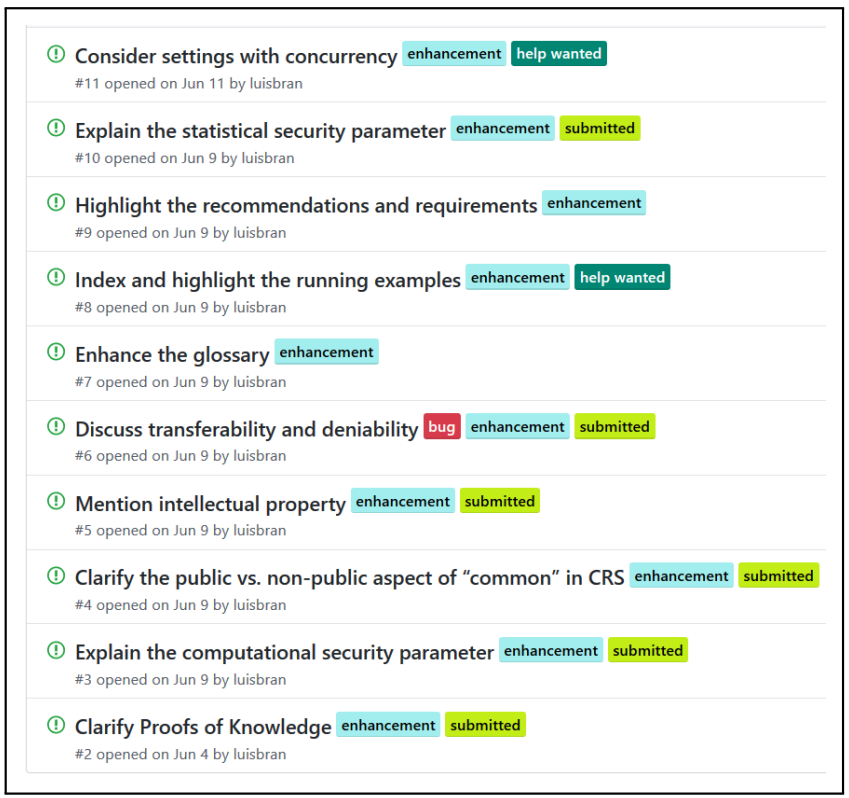
- LateX transport by NIST ⟵ Community Reference document (0.1)
- 2nd Workshop with many sessions focused on adding content
- Editorial process defined (contributions, comments, etc...)
- Total of about 15 contributions for first round

Screenshots of the GitHub repository

https://github.com/zkpstandard/zkreference



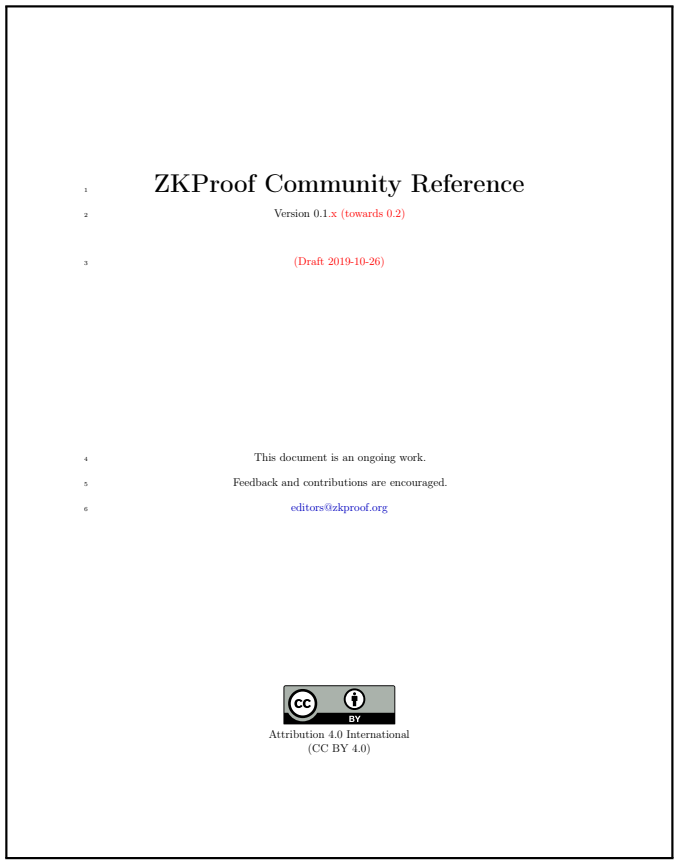
LaTeX source



Currently 26 issues

Outline 2

A few revised aspects in the Ref. 0.2



ZKProof Community Reference

Version 0.1.x (towards 0.2)

(Draft 2019-10-26)

This document is an ongoing work.

Feedback and contributions are encouraged.

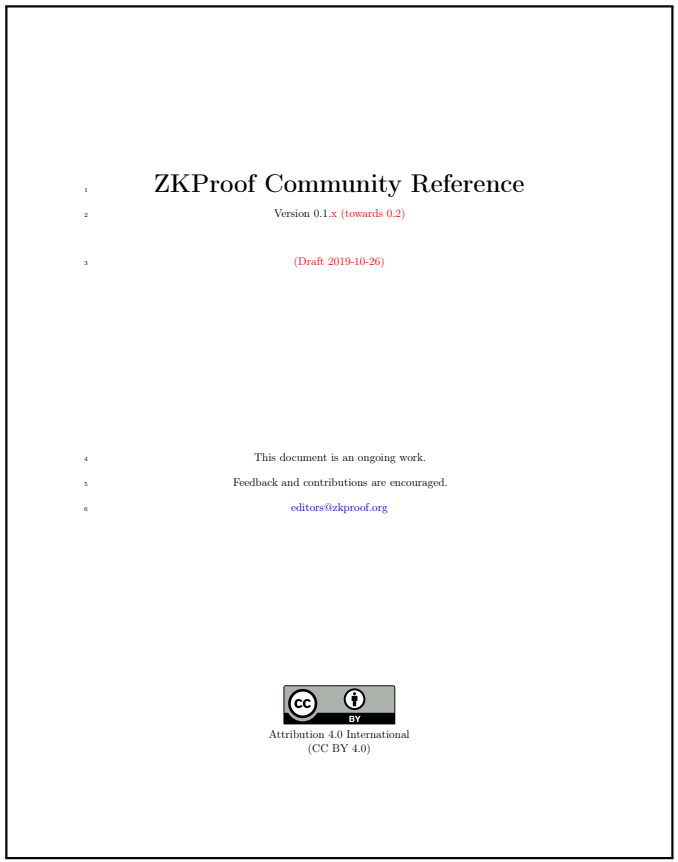
editors@zkproof.org

Now: version 0.1.x (integrating contributions)

Soon: draft 0.2, expected 2nd half Nov.

Public feedback: editors@zkproof.org

A few revised aspects in the Ref. 0.2



ZKProof Community Reference

Version 0.1.x (towards 0.2)

(Draft 2019-10-26)

This document is an ongoing work.

Feedback and contributions are encouraged.

editors@zkproof.org

Examples in next slides:

1. High-level ZKP examples
2. Transferability/deniability
3. Types of proof: knowledge vs. membership

Goal: convey some conceptual nuances

Now: version 0.1.x (integrating contributions)

Soon: draft 0.2, expected 2nd half Nov.

Public feedback: editors@zkproof.org

High-level examples of ZKP

The challenge of explaining ZKPs to people outside of the area.

At what level do we explain ZKPs, when we explain ZKPs at a high level?

(redundancy intended)

High-level examples of ZKP

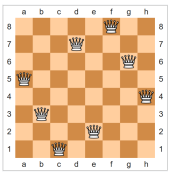
The challenge of explaining ZKPs to people outside of the area.

At what level do we explain ZKPs, when we explain ZKPs at a high level?

(redundancy intended)

Compare the following: *With a ZKP, can I prove?:*

- I am an adult
- My car license-plate starts with an 'A'
- My favorite color is in {blue, red, green}
- I know a Chess solution to the "8 queens puzzle"



High-level examples of ZKP

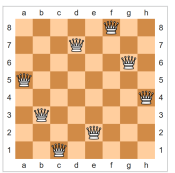
The challenge of explaining ZKPs to people outside of the area.

At what level do we explain ZKPs, when we explain ZKPs at a high level?

(redundancy intended)

Compare the following: *With a ZKP, can I prove?:*

- I am an adult [I surely have a birthdate, but how to confirm it is correct?]
- My car license-plate starts with an 'A' [Do you even know if I have a car?]
- My favorite color is in {blue, red, green} [Can lie about it]
- I know a Chess solution to the "8 queens puzzle" [Well defined]



High-level examples of ZKP

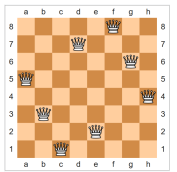
The challenge of explaining ZKPs to people outside of the area.

At what level do we explain ZKPs, when we explain ZKPs at a high level?

(redundancy intended)

Compare the following: *With a ZKP, can I prove?:*

- I am an adult [I surely have a birthdate, but how to confirm it is correct?]
- My car license-plate starts with an 'A' [Do you even know if I have a car?]
- My favorite color is in {blue, red, green} [Can lie about it]
- I know a Chess solution to the "8 queens puzzle" [Well defined]



The *statement* (and the implicit simplified *witness*) may be insufficient ...

to allow an actively interested listener to build more intuition.

Example scenarios for zero-knowledge proofs:

Version high-high level:

- Prove adulthood, without revealing the birth date;
- Prove solvency (not bankrupt), without showing portfolio composition;
- Prove a chessboard config is valid, without revealing sequence of moves;

Example scenarios for zero-knowledge proofs:

Version high-high level:

- Prove adulthood, without revealing the birth date;
- Prove solvency (not bankrupt), without showing portfolio composition;
- Prove a chessboard config is valid, without revealing sequence of moves;

Version high level:

#	Elements / Scenarios	**Statement** being proven		**Witness** treated as confidential
1	**Legal age for purchase**	I am an adult		Birthdate and personal data
2	**Hedge fund solvency**	We are not bankrupt		Portfolio data
4	**Chessboard configuration**	This <configuration> can be reached		A sequence of valid chess moves

Example scenarios for zero-knowledge proofs:

Version high-high level:

- Prove adulthood, without revealing the birth date;
- Prove solvency (not bankrupt), without showing portfolio composition;
- Prove a chessboard config is valid, without revealing sequence of moves;

Version high level:

#	Elements / Scenarios	**Statement** being proven	**Instance** used as substrate	**Witness** treated as confidential
1	**Legal age for purchase**	I am an adult	Tamper-resistant identification chip	Birthdate and personal data (signed by a certification authority)
2	**Hedge fund solvency**	We are not bankrupt	Encrypted & certified bank records	Portfolio data and decryption key
4	**Chessboard configuration**	This <configuration> can be reached	(The rules of Chess)	A sequence of valid chess moves

Proofs of Knowledge vs. of Membership?

A ZKP proves that a *statement* is truthful and reveals nothing else (?)

Proofs of Knowledge vs. of Membership?

A ZKP proves that a *statement* is truthful and reveals nothing else (?)

What kind of *statement* is proven? (About a public *instance* $\boldsymbol{x}$.)

- Statement of membership: $\boldsymbol{x} \in L$
- Statement of knowledge: I known witness $\boldsymbol{w}$ such that $(\boldsymbol{x}, \boldsymbol{w}) \in R$

Proofs of Knowledge vs. of Membership?

A ZKP proves that a *statement* is truthful and reveals nothing else (?)

What kind of *statement* is proven? (About a public *instance* $\boldsymbol{x}$.)

- Statement of membership: $\boldsymbol{x} \in L$
- Statement of knowledge: I known witness $\boldsymbol{w}$ such that $(\boldsymbol{x}, \boldsymbol{w}) \in R$

Sometimes interchangeable, but not always!

Proofs of Knowledge vs. of Membership?

A ZKP proves that a *statement* is truthful and reveals nothing else (?)

What kind of *statement* is proven? (About a public *instance* $\boldsymbol{x}$.)

- Statement of membership: $\boldsymbol{x} \in L$
- Statement of knowledge: I known witness $\boldsymbol{w}$ such that $(\boldsymbol{x}, \boldsymbol{w}) \in R$

Sometimes interchangeable, but not always!

Examples ... Prove:

- knowledge of discrete log $\boldsymbol{w} : \boldsymbol{x} = g^{\boldsymbol{w}}$
- that the graphs in the pair $\boldsymbol{x} = (G_1, G_2)$ are non-isomorphic, i.e., $\boldsymbol{x} \in$ GNI
- knowledge of a hash pre-image $\boldsymbol{w} : \boldsymbol{x} = H(\boldsymbol{w})$
- that a value $\boldsymbol{x}$ has a hash pre-image, i.e., $\boldsymbol{x} \in \{y' = H(w) : w \in *\}$

Proofs of Knowledge vs. of Membership?

A ZKP proves that a *statement* is truthful and reveals nothing else (?)

What kind of *statement* is proven? (About a public *instance* $\boldsymbol{x}$.)

- Statement of membership: $\boldsymbol{x} \in L$
- Statement of knowledge: I known witness $\boldsymbol{w}$ such that $(\boldsymbol{x}, \boldsymbol{w}) \in R$

Sometimes interchangeable, but not always!

Examples ... Prove:

- knowledge of discrete log $\boldsymbol{w} : \boldsymbol{x} = g^{\boldsymbol{w}}$
- that the graphs in the pair $\boldsymbol{x} = (G_1, G_2)$ are non-isomorphic, i.e., $\boldsymbol{x} \in$ GNI
- knowledge of a hash pre-image $\boldsymbol{w} : \boldsymbol{x} = H(\boldsymbol{w})$
- that a value $\boldsymbol{x}$ has a hash pre-image, i.e., $\boldsymbol{x} \in \{y' = H(w) : w \in *\}$

Why is this important? The security properties are different:

soundness (ZKP of membership) vs. **extractability** (ZKP of knowledge).

Transferability vs. deniability

Each of them can be considered a feature

Transferability vs. deniability

Each of them can be considered a feature ... or a bug!?

Transferability vs. deniability

Each of them can be considered a feature ... or a bug!?

- Traditional ZKPs are deniable (i.e., non-transferable)
- Certain applications benefit from transferable proofs / publicly verifiable

Transferability vs. deniability

Each of them can be considered a feature ... or a bug!?

- Traditional ZKPs are deniable (i.e., non-transferable)
- Certain applications benefit from transferable proofs / publicly verifiable

Non-interactive setting: can you do a *deniable* ZKPoK of witness w?

Transferability vs. deniability

Each of them can be considered a feature ... or a bug!?

- Traditional ZKPs are deniable (i.e., non-transferable)
- Certain applications benefit from transferable proofs / publicly verifiable

Non-interactive setting: can you do a *deniable* ZKPoK of witness w? **Yes**, e.g., prove knowledge of w OR of the verifier's private key (within a PKI).

Transferability vs. deniability

Each of them can be considered a feature ... or a bug!?

- Traditional ZKPs are deniable (i.e., non-transferable)
- Certain applications benefit from transferable proofs / publicly verifiable

Non-interactive setting: can you do a *deniable* ZKPoK of witness w? **Yes**, e.g., prove knowledge of w OR of the verifier's private key (within a PKI).

Interactive setting: can it be *transferable*?

Transferability vs. deniability

Each of them can be considered a feature ... or a bug!?

- Traditional ZKPs are deniable (i.e., non-transferable)
- Certain applications benefit from transferable proofs / publicly verifiable

Non-interactive setting: can you do a *deniable* ZKPoK of witness w? **Yes**, e.g., prove knowledge of w OR of the verifier's private key (within a PKI).

Interactive setting: can it be *transferable*? **Yes**, e.g., Fiat-Shamir based.

Transferability vs. deniability

Each of them can be considered a feature ... or a bug!?

- Traditional ZKPs are deniable (i.e., non-transferable)
- Certain applications benefit from transferable proofs / publicly verifiable

Non-interactive setting: can you do a *deniable* ZKPoK of witness w? **Yes**, e.g., prove knowledge of w OR of the verifier's private key (within a PKI).

Interactive setting: can it be *transferable*? **Yes**, e.g., Fiat-Shamir based.

A funny case: give a non-transferable proof that you possess a transferable proof.

Transferability vs. deniability

Each of them can be considered a feature ... or a bug!?

- Traditional ZKPs are deniable (i.e., non-transferable)
- Certain applications benefit from transferable proofs / publicly verifiable

Non-interactive setting: can you do a *deniable* ZKPoK of witness w? **Yes**, e.g., prove knowledge of w OR of the verifier's private key (within a PKI).

Interactive setting: can it be *transferable*? **Yes**, e.g., Fiat-Shamir based.

A funny case: give a non-transferable proof that you possess a transferable proof. **Example:** an auditor obtains a *transferable* ZKP transcript; later it responds *deniably* to a query from an ongoing investigation.

Transferability vs. deniability

Each of them can be considered a feature ... or a bug!?

- Traditional ZKPs are deniable (i.e., non-transferable)
- Certain applications benefit from transferable proofs / publicly verifiable

Non-interactive setting: can you do a *deniable* ZKPoK of witness w? **Yes**, e.g., prove knowledge of w OR of the verifier's private key (within a PKI).

Interactive setting: can it be *transferable*? **Yes**, e.g., Fiat-Shamir based.

A funny case: give a non-transferable proof that you possess a transferable proof. **Example:** an auditor obtains a *transferable* ZKP transcript; later it responds *deniably* to a query from an ongoing investigation.

A composability case: Assume a deniable proof ... what if the underlying communication protocol authenticated all the messages?

Call for contributions

Soon: Version 0.2 out for public comments (tentative: 2nd half of November).

Call for contributions

Soon: Version 0.2 out for public comments (tentative: 2nd half of November).

Please consider contributing with your feedback!

- What is not clear?
- Missing content or explanations?
- Technical accuracy
- General text revision
- Use-cases of interest (next slides)

Call for contributions

Soon: Version 0.2 out for public comments (tentative: 2nd half of November).

Please consider contributing with your feedback!

- What is not clear?
- Missing content or explanations?
- Technical accuracy
- General text revision
- Use-cases of interest (next slides)

Later: 3rd ZKProof Workshop (April 2020) — organize more contributions.

Intellectual Property (expectations)

ZKProof is an open initiative that seeks to promote the secure and interoperable use of zero-knowledge proofs. To foster open development and wide adoption, it is valuable to promote technologies with open-source implementations, unencumbered by royalty-bearing patents. However, some useful technologies may fall within the scope of patent claims. Since ZKProof seeks to represent the technology, research and community in an inclusive manner, it is valuable to set expectations about the disclosure of intellectual property and the handling of patent claims.

The members of the ZKProof community are hereby strongly encouraged to provide information on known patent claims potentially applicable to the guidance, requirements, recommendations, proposals and examples provided in ZKProof documentation, including by disclosing known pending patent applications or any relevant unexpired patent. Particularly, such disclosure is promptly required from the patent holders, or those acting on their behalf, as a condition for providing content contributions to the "Community Reference" and to "Proposals" submitted to ZKProof for consideration by the community. Furthermore, any technology that is promoted in said ZKProof documentation and that falls within patent claims should be made available under licensing terms that are reasonable, and demonstrably free of unfair discrimination, preferably allowing free open-source implementations.

The ZKProof documentation will be updated based on received disclosures about pertinent patent claims. Please email information to editors@zkproof.org.

Intellectual Property (expectations)

ZKProof is an open initiative that seeks to promote the secure and interoperable use of zero-knowledge proofs. To foster open development and wide adoption, it is valuable to promote technologies with open-source implementations, unencumbered by royalty-bearing patents. However, some useful technologies may fall within the scope of patent claims. Since ZKProof seeks to represent the technology, research and community in an inclusive manner, it is valuable to set expectations about the disclosure of intellectual property and the handling of patent claims.

The members of the ZKProof community are hereby strongly encouraged to provide information on known patent claims potentially applicable to the guidance, requirements, recommendations, proposals and examples provided in ZKProof documentation, including by disclosing known pending patent applications or any relevant unexpired patent. Particularly, such disclosure is promptly required from the patent holders, or those acting on their behalf, as a condition for providing content contributions to the "Community Reference" and to "Proposals" submitted to ZKProof for consideration by the community. Furthermore, any technology that is promoted in said ZKProof documentation and that falls within patent claims should be made available under licensing terms that are reasonable, and demonstrably free of unfair discrimination, preferably allowing free open-source implementations.

The ZKProof documentation will be updated based on received disclosures about pertinent patent claims. Please email information to editors@zkproof.org.

Outline 3

Recommendations

The reference document will improve if complemented with recommendations and examples of interoperable components.

Next slides:

- Security levels
- Metrics
- Intellectual property
- NIST-PEC proposal of a use-case suite

Security level parameters

In terms of **computational security** in benchmarks:

- The reference **requires** $\kappa \geq 128$; **suggests** one more $\kappa \in \{192, 256\}$
- Soundness despite long pre- or online computation
- Zero-knowledge despite long pre-, online, or post-computation

Security level parameters

In terms of **computational security** in benchmarks:

- The reference **requires** $\kappa \geq 128$; **suggests** one more $\kappa \in \{192, 256\}$
- Soundness despite long pre- or online computation
- Zero-knowledge despite long pre-, online, or post-computation

Statistical security (in interactive case):

- The reference **requires** $\sigma \geq 64$; **suggests** one more $\sigma \in \{40, 80, 128\}$
- One-shot online security for statistical soundness
- Fiat-Shamir may require $\sigma \approx \kappa$ (statistical $\rightarrow$ computational)

Security level parameters

In terms of **computational security** in benchmarks:

- The reference **requires** $\kappa \geq 128$; **suggests** one more $\kappa \in \{192, 256\}$
- Soundness despite long pre- or online computation
- Zero-knowledge despite long pre-, online, or post-computation

Statistical security (in interactive case):

- The reference **requires** $\sigma \geq 64$; **suggests** one more $\sigma \in \{40, 80, 128\}$
- One-shot online security for statistical soundness
- Fiat-Shamir may require $\sigma \approx \kappa$ (statistical $\rightarrow$ computational)

Exceptions for lower security levels (to be careful):

- if needing short term comp. security, e.g., for temporary binding or hiding;
- if predicate being proven is protected by less security strength.

Benchmarking suggestions

The Reference gives a few generic suggestions.

Measure and compare several metrics:

- Communication and computational complexity
- Per phase (prove vs. verify), per implementation platform
- Many nuances: Parallelizability, Batching, memory, disk, cpu, tradeoffs

Benchmarking suggestions

The Reference gives a few generic suggestions.

Measure and compare several metrics:

- Communication and computational complexity
- Per phase (prove vs. verify), per implementation platform
- Many nuances: Parallelizability, Batching, memory, disk, cpu, tradeoffs

Several functions in ZKP (knowledge of pre-image or of committed input/output): SHA-256, AES-128, matrix-multiplication, Scrypt, number theoretical transforms (small and big fields), ...

Beyound benchmarking

How about when efficiency is not an issue at all?

Beyound benchmarking

How about when efficiency is not an issue at all?

Proposal: a use-case suite could be good to facilitate experimentation by new implementors.

- Collect fully functional open-source implementations of very concrete use-cases, e.g., proving adulthood based on a digital certificate.
- NIST-PEC wants to propose such a suite (≈6 months) including use-cases on ZKP, SMPC, ...

A use-case: public auditability from public randomness

Public		Inherently private			Derived private	
# (i)	Rand id	Name (N)	a_1	a_2	Weight (w)	Acc. (W)
1	371	**C**ai	1	2	0.1	0.1
2	942	**E**ve	2	7	0.3	0.4
3	107	**B**ob	1	5	0.2	0.6
4	527	**A**nn	1	9	0.3	0.9
5	123	**D**an	3	1	0.1	1.0

A use-case: public auditability from public randomness

Public		Inherently private			Derived private	
# (i)	Rand id	Name (N)	a_1	a_2	Weight (w)	Acc. (W)
1	371	**C**ai	1	2	0.1	0.1
2	942	**E**ve	2	7	0.3	0.4
3	107	**B**ob	1	5	0.2	0.6
4	527	**A**nn	1	9	0.3	0.9
5	123	**D**an	3	1	0.1	1.0

Publicize a table with all attributes **committed**

A use-case: public auditability from public randomness

Public		Inherently private			Derived private	
# (i)	Rand id	Name (N)	a_1	a_2	Weight (w)	Acc. (W)
1	371	**C**ai	1	2	0.1	0.1
2	942	**E**ve	2	7	0.3	0.4
3	107	**B**ob	1	5	0.2	0.6
4	527	**A**nn	1	9	0.3	0.9
5	123	**D**an	3	1	0.1	1.0

Publicize a table with all attributes **committed** ... then **prove in ZK**:

1. $a_i \in A$ (e.g., salary level); $b_i \in B$ (e.g., years at work);
2. $w_i = f(a_i, b_i)$ (correct weight calculation);
3. $\sum_i w_i = 1$ (correct sum of probabilities);
4. $W_i = w_i + W_{i-1}$ (correct probability accumulation);
5. $\{N_i\}$ =NAMES (no repeated names from an appropriate set); ...

A use-case: public auditability from public randomness

Public		Inherently private			Derived private	
# (i)	Rand id	Name (N)	a_1	a_2	Weight (w)	Acc. (W)
1	371	**C**ai	1	2	0.1	0.1
2	942	**E**ve	2	7	0.3	0.4
3	107	**B**ob	1	5	0.2	0.6
4	527	**A**nn	1	9	0.3	0.9
5	123	**D**an	3	1	0.1	1.0

Publicize a table with all attributes **committed** ... then **prove in ZK**:

1. $a_i \in A$ (e.g., salary level); $b_i \in B$ (e.g., years at work);
2. $w_i = f(a_i, b_i)$ (correct weight calculation);
3. $\sum_i w_i = 1$ (correct sum of probabilities);
4. $W_i = w_i + W_{i-1}$ (correct probability accumulation);
5. $\{N_i\}$ =NAMES (no repeated names from an appropriate set); ...

Derive $R : 0 < R \leq 1$ (random) from Beacon and get # j: $W_{\text{máx}(1,j-1)} < R \leq W_j$

- **Prove in ZK** that j is consistent with R and the table of commitments

Intellectual Property (expectations)

ZKProof is an open initiative that seeks to promote the secure and interoperable use of zero-knowledge proofs. To foster open development and wide adoption, it is valuable to promote technologies with open-source implementations, unencumbered by royalty-bearing patents. However, some useful technologies may fall within the scope of patent claims. Since ZKProof seeks to represent the technology, research and community in an inclusive manner, it is valuable to set expectations about the disclosure of intellectual property and the handling of patent claims.

The members of the ZKProof community are hereby strongly encouraged to provide information on known patent claims potentially applicable to the guidance, requirements, recommendations, proposals and examples provided in ZKProof documentation, including by disclosing known pending patent applications or any relevant unexpired patent. Particularly, such disclosure is promptly required from the patent holders, or those acting on their behalf, as a condition for providing content contributions to the "Community Reference" and to "Proposals" submitted to ZKProof for consideration by the community. Furthermore, any technology that is promoted in said ZKProof documentation and that falls within patent claims should be made available under licensing terms that are reasonable, and demonstrably free of unfair discrimination, preferably allowing free open-source implementations.

The ZKProof documentation will be updated based on received disclosures about pertinent patent claims. Please email information to editors@zkproof.org.

Intellectual Property (expectations)

ZKProof is an open initiative that seeks to promote the secure and interoperable use of zero-knowledge proofs. To foster open development and wide adoption, it is valuable to promote technologies with open-source implementations, unencumbered by royalty-bearing patents. However, some useful technologies may fall within the scope of patent claims. Since ZKProof seeks to represent the technology, research and community in an inclusive manner, it is valuable to set expectations about the disclosure of intellectual property and the handling of patent claims.

The members of the ZKProof community are hereby strongly encouraged to provide information on known patent claims potentially applicable to the guidance, requirements, recommendations, proposals and examples provided in ZKProof documentation, including by disclosing known pending patent applications or any relevant unexpired patent. Particularly, such disclosure is promptly required from the patent holders, or those acting on their behalf, as a condition for providing content contributions to the "Community Reference" and to "Proposals" submitted to ZKProof for consideration by the community. Furthermore, any technology that is promoted in said ZKProof documentation and that falls within patent claims should be made available under licensing terms that are reasonable, and demonstrably free of unfair discrimination, preferably allowing free open-source implementations.

The ZKProof documentation will be updated based on received disclosures about pertinent patent claims. Please email information to editors@zkproof.org.

Outline 4

Discussion

Suggested questions for brainstorming:

- What is best way to contribute and to attract community?
- What is the applicability of the Z.C.Reference to your setting!?
- Use-cases of interest for the suite?

What else would you like to see in the Z.C.Reference?

Discussion part 2

What about more concretely?

- A library of secure gadgets for usage
- A proper review of elliptic curve parameters for different usages

What else do we need to standardize to build the ZK infrastructure for adoption?

Questions?

Thank you!