Under The Hood of zkID:

The Common Design Pattern for Attested Data + Where Do We Go From Here

Nicolas Mohnblatt — 19 August 2024

ZK x ID applications

- Proving web2 data: ZK Email, Web Proofs, TLS Oracles
- Wallets using OpenID: OpenPubKey, zkLogin, Aptos Keyless
- Proving real-world data: zkPassport, VerITAS (provable photo edits), Proof of GPS location*, attested microphone*

Claim: all these systems rely on the same design pattern

* ZK Hack projects

This Talk

Part 1: explain the common design pattern

Part 2: what's next?

making these systems faster

building on top of the existing capabilities

Part 3 (bonus) O-SNARKs: fun security considerations

Background

Digital Signature

- secret key + public key
- Signer generates signature using sk, Verifier verifies using message and pk
- unforgeability + non-repudiation

NIZK

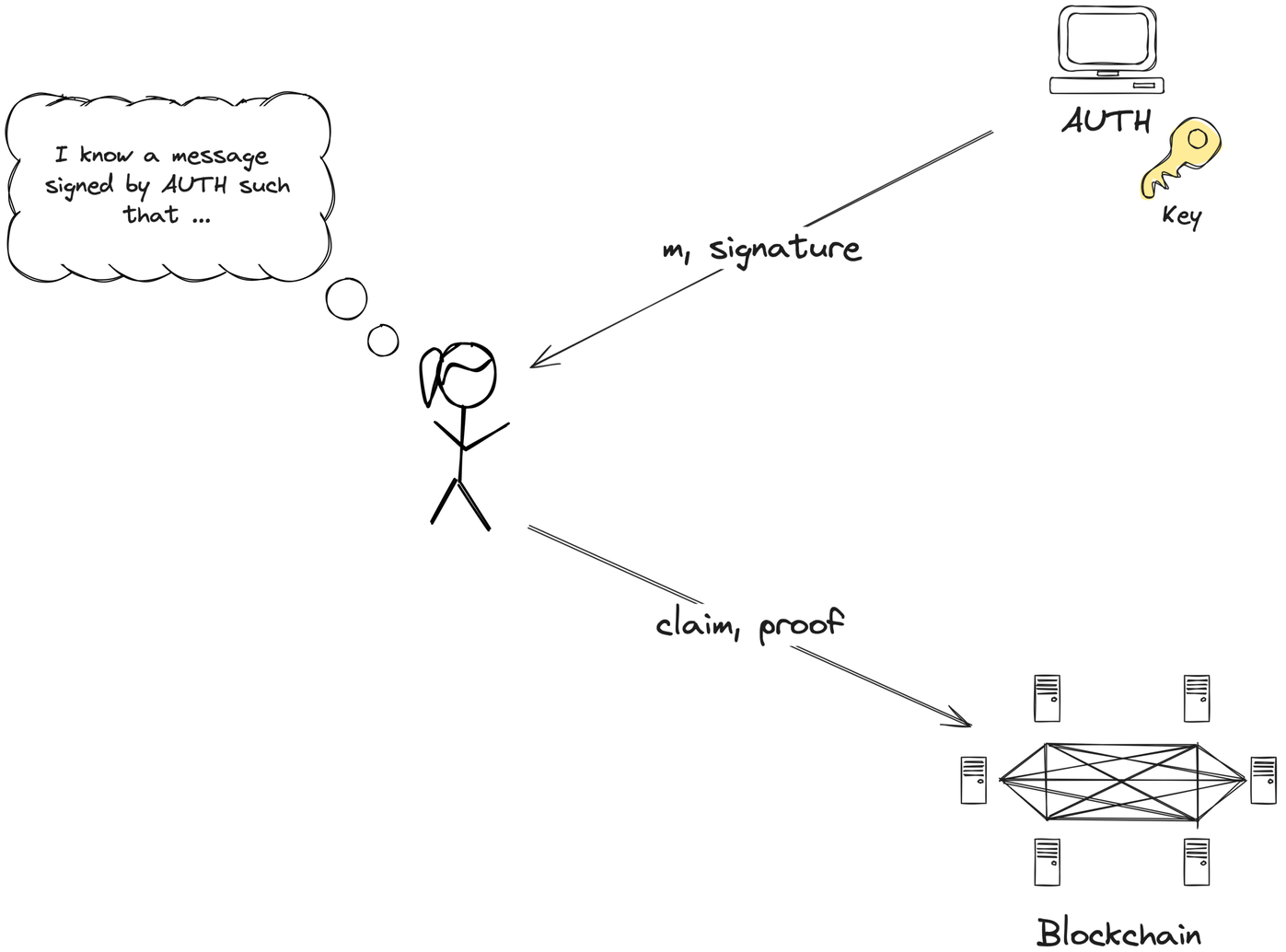
- Prover emits proof to Verifier
- Completeness, soundness and zero-knowledge

MAC

- symmetric key
- single key is used to authenticate and verify msg
- unforgeability only

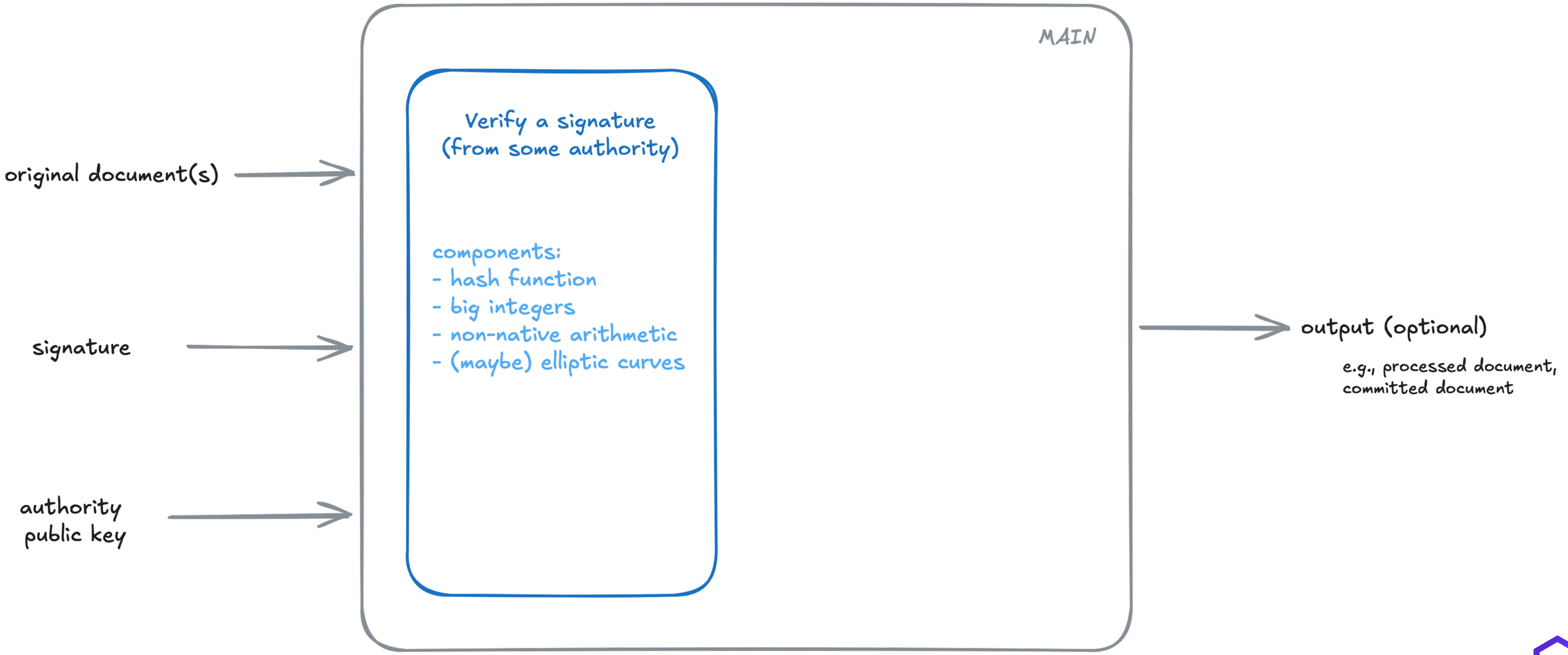
Part 1: the design pattern

Setting

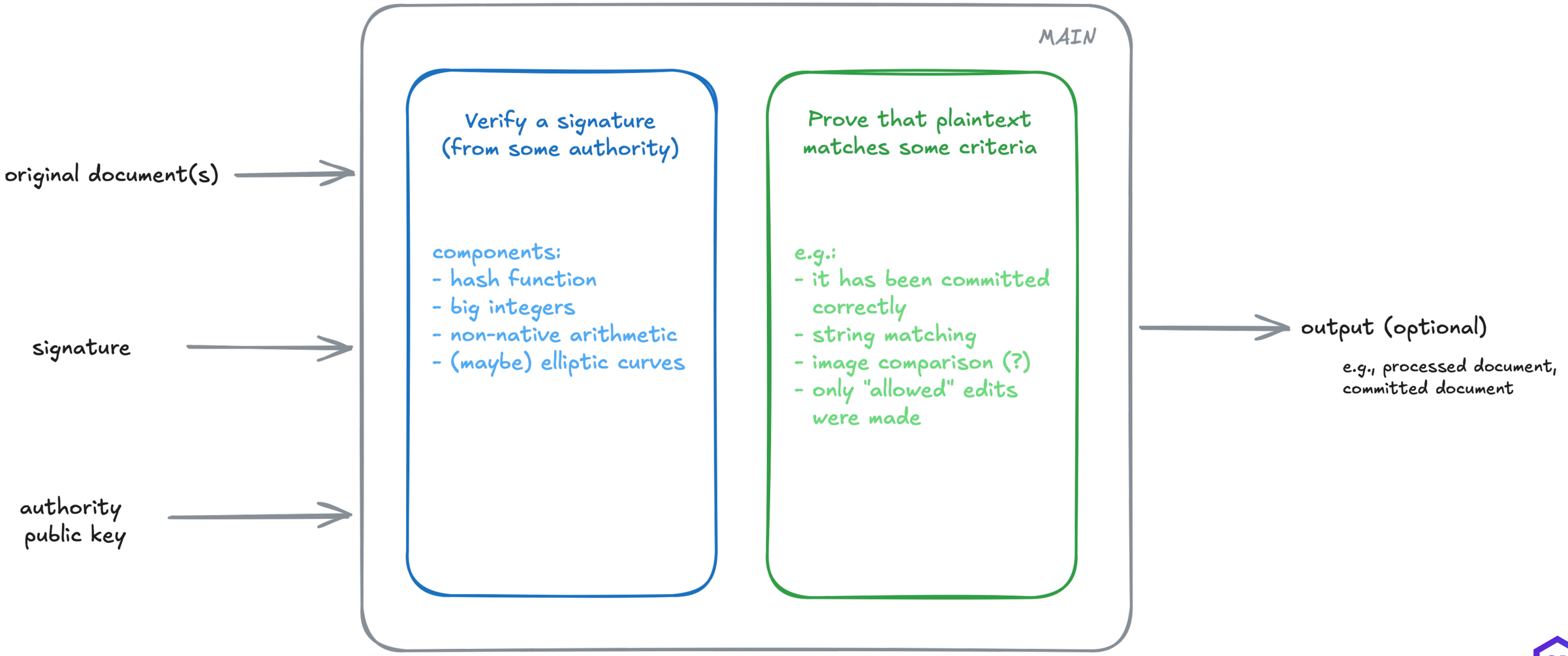

I know a message signed by AUTH such that ...
AUTH
Key
m, signature
claim, proof
Blockchain

The circuit we're all building

The circuit we're all building

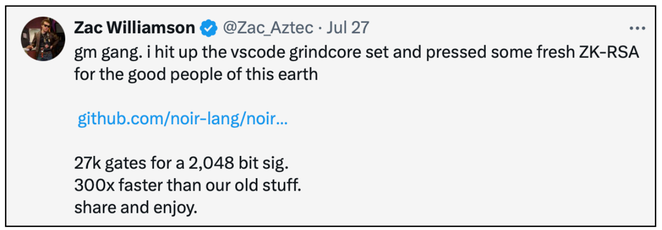


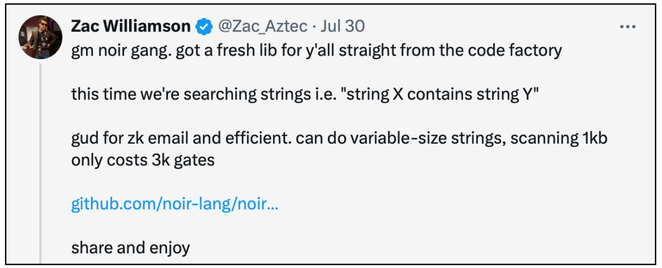
The circuit we're all building

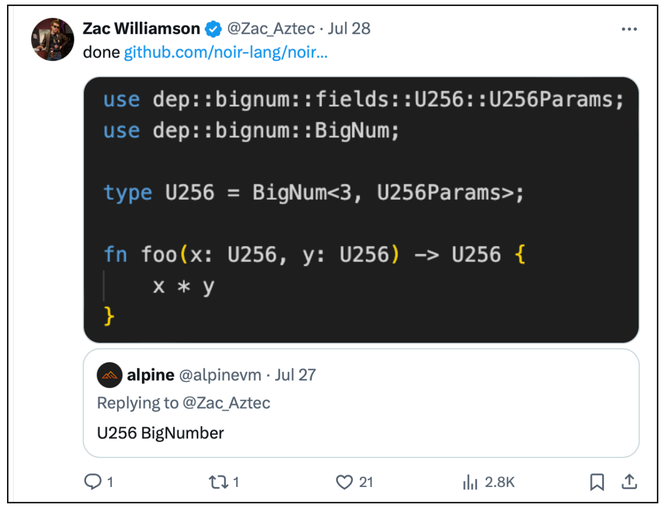


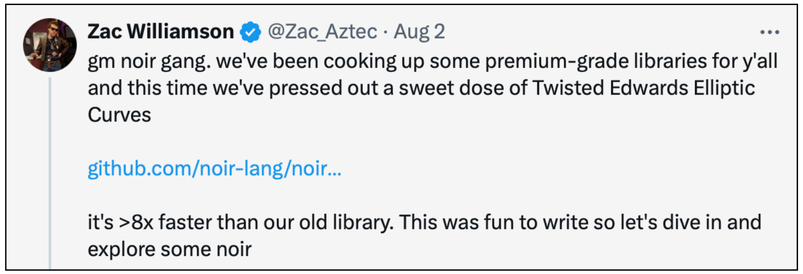
Performance requirements

- This circuit needs to be proven **client-side**, or with **privacy-preserving delegation**.
- Efforts are currently focused here:



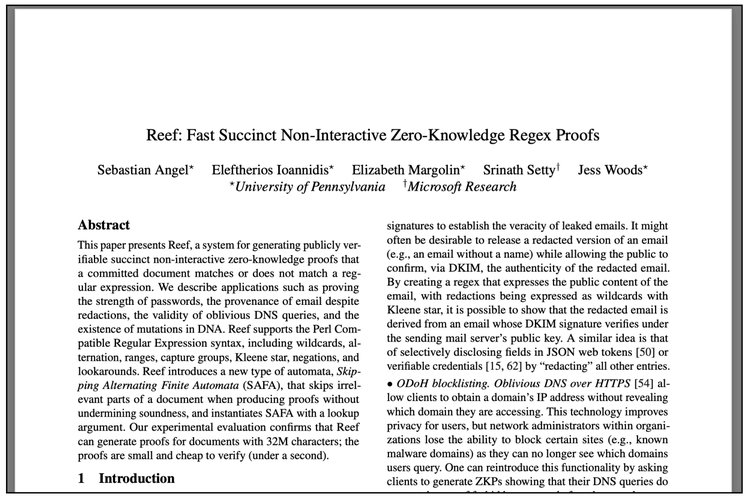






Performance requirements

- This circuit needs to be proven **client-side**, or with **privacy-preserving delegation**.
- Efforts are currently focused here:



Reef: Fast Succinct Non-Interactive Zero-Knowledge Regex Proofs

Sebastian Angel* Eleftherios Ioannidis* Elizabeth Margolin* Srinath Setty† Jess Woods*
*University of Pennsylvania †Microsoft Research

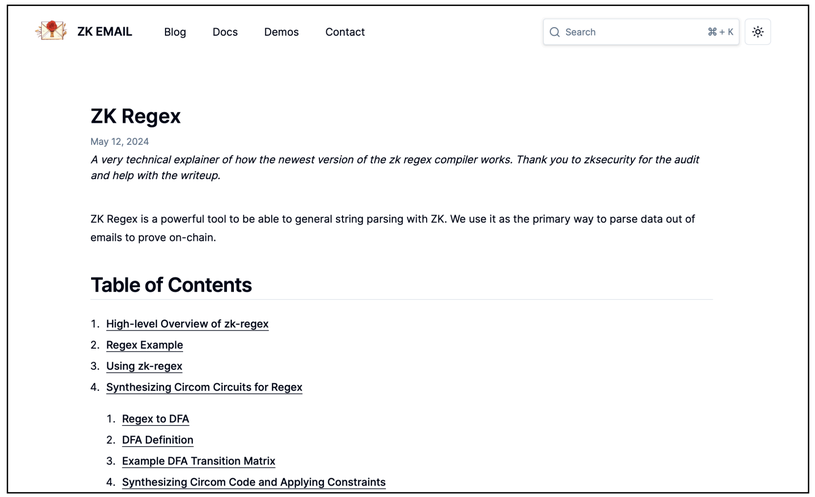
Abstract

This paper presents Reef, a system for generating publicly verifiable succinct non-interactive zero-knowledge proofs that a committed document matches or does not match a regular expression. We describe applications such as proving the strength of passwords, the provenance of email despite redactions, the validity of oblivious DNS queries, and the existence of mutations in DNA. Reef supports the Perl Compatible Regular Expression syntax, including wildcards, alternation, ranges, capture groups, Kleene star, negations, and lookarounds. Reef introduces a new type of automata, *Skipping Alternating Finite Automata* (SAFA), that skips irrelevant parts of a document when producing proofs without undermining soundness, and instantiates SAFA with a lookup argument. Our experimental evaluation confirms that Reef can generate proofs for documents with 32M characters; the proofs are small and cheap to verify (under a second).

1 Introduction

signatures to establish the veracity of leaked emails. It might often be desirable to release a redacted version of an email (e.g., an email without a name) while allowing the public to confirm, via DKIM, the authenticity of the redacted email. By creating a regex that expresses the public content of the email, with redactions being expressed as wildcards with Kleene star, it is possible to show that the redacted email is derived from an email whose DKIM signature verifies under the sending mail server's public key. A similar idea is that of selectively disclosing fields in JSON web tokens [50] or verifiable credentials [15, 62] by "redacting" all other entries.

• *ODoH blocklisting. Oblivious DNS over HTTPS* [54] allow clients to obtain a domain's IP address without revealing which domain they are accessing. This technology improves privacy for users, but network administrators within organizations lose the ability to block certain sites (e.g., known malware domains) as they can no longer see which domains users query. One can reintroduce this functionality by asking clients to generate ZKPs showing that their DNS queries do



ZK EMAIL Blog Docs Demos Contact Search ⌘+K

ZK Regex

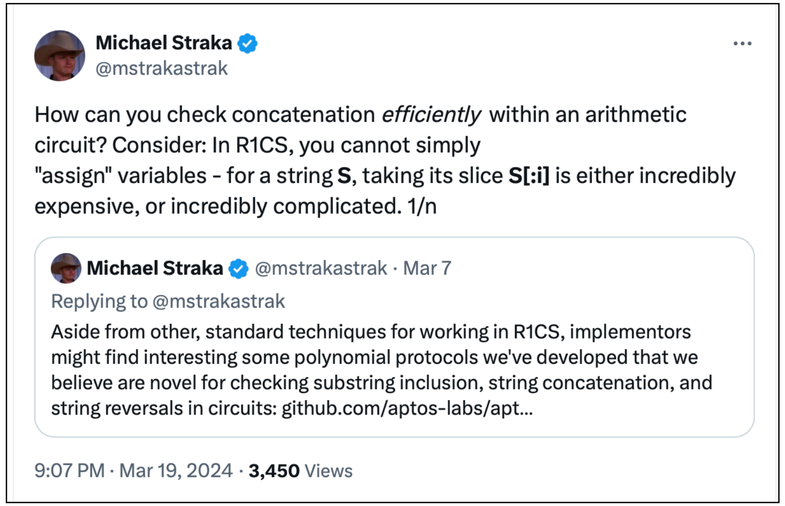
May 12, 2024

A very technical explainer of how the newest version of the zk regex compiler works. Thank you to zksecurity for the audit and help with the writeup.

ZK Regex is a powerful tool to be able to general string parsing with ZK. We use it as the primary way to parse data out of emails to prove on-chain.

Table of Contents

1. High-level Overview of zk-regex
2. Regex Example
3. Using zk-regex
4. Synthesizing Circom Circuits for Regex
    1. Regex to DFA
    2. DFA Definition
    3. Example DFA Transition Matrix
    4. Synthesizing Circom Code and Applying Constraints



Michael Straka
@mstrakastrak

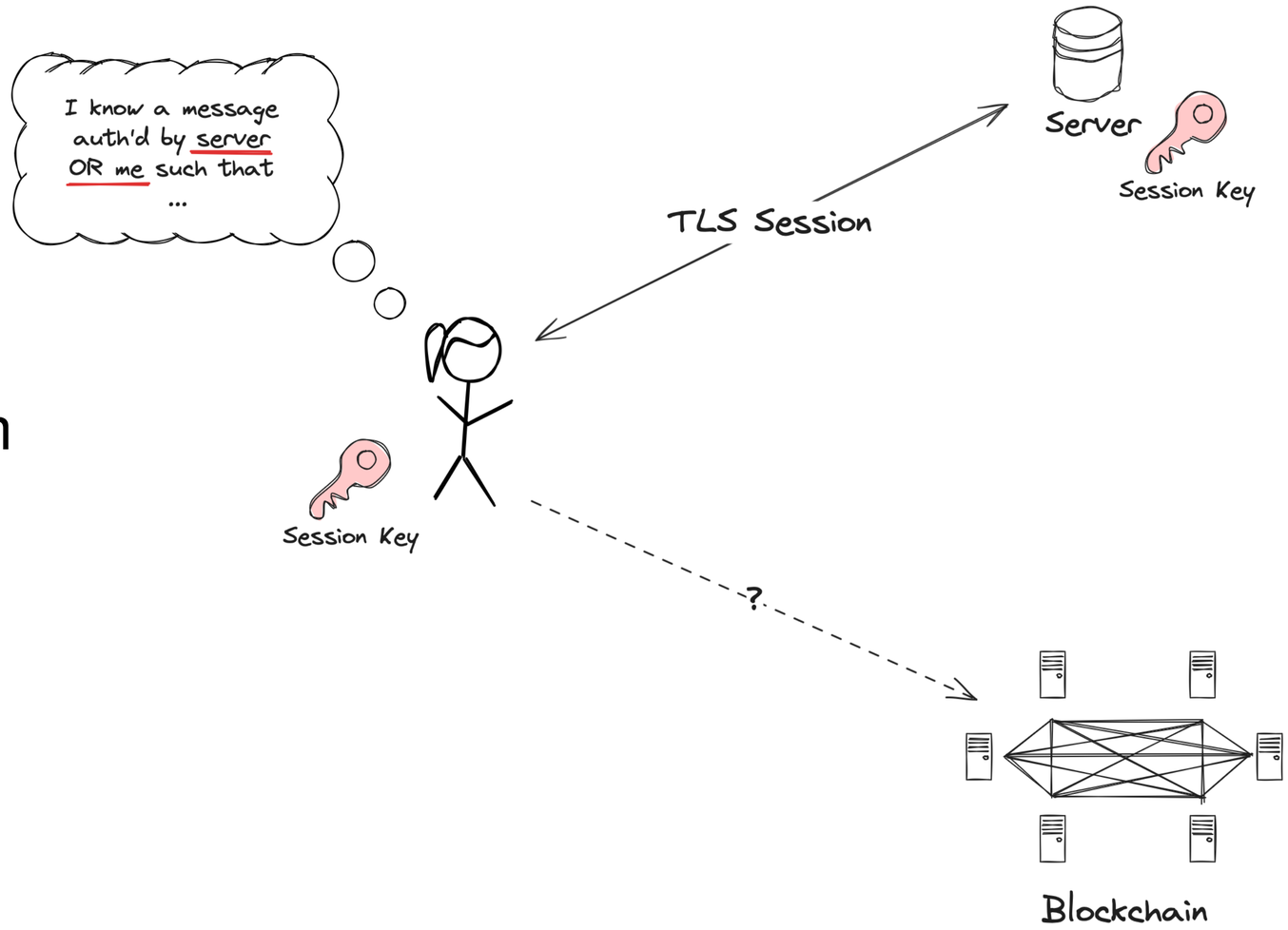
How can you check concatenation *efficiently* within an arithmetic circuit? Consider: In R1CS, you cannot simply "assign" variables - for a string **S**, taking its slice **S[:i]** is either incredibly expensive, or incredibly complicated. 1/n

> Michael Straka @mstrakastrak · Mar 7
> Replying to @mstrakastrak
> Aside from other, standard techniques for working in R1CS, implementors might find interesting some polynomial protocols we've developed that we believe are novel for checking substring inclusion, string concatenation, and string reversals in circuits: github.com/aptos-labs/apt...

9:07 PM · Mar 19, 2024 · **3,450** Views

Variant: proving TLS

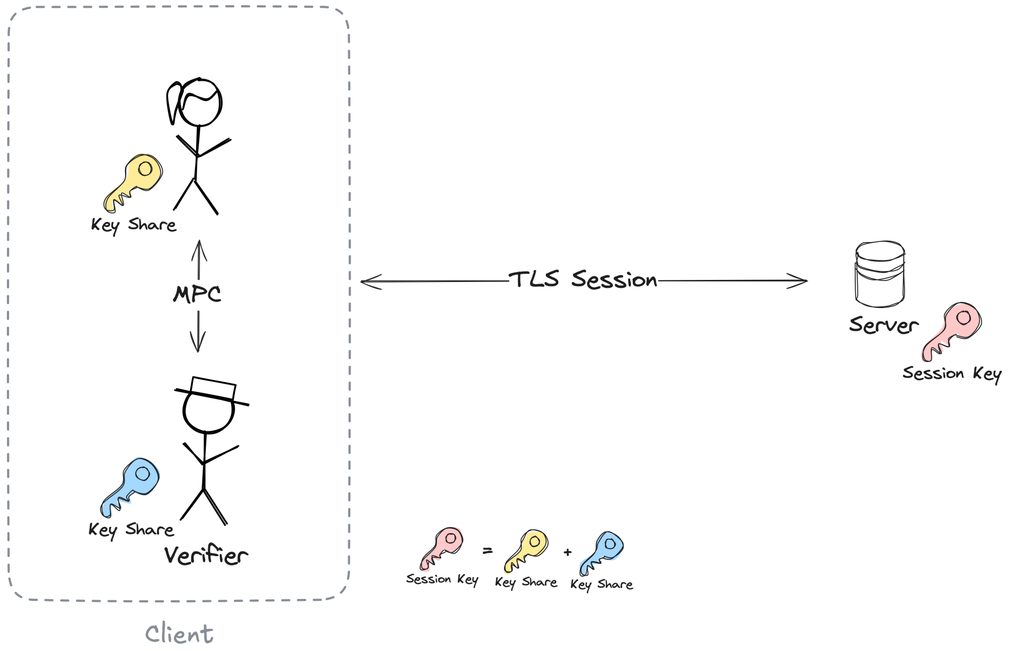
- TLS uses *symmetric* cryptography (recall MAC)
- Need to prevent the Prover from authenticating arbitrary messages



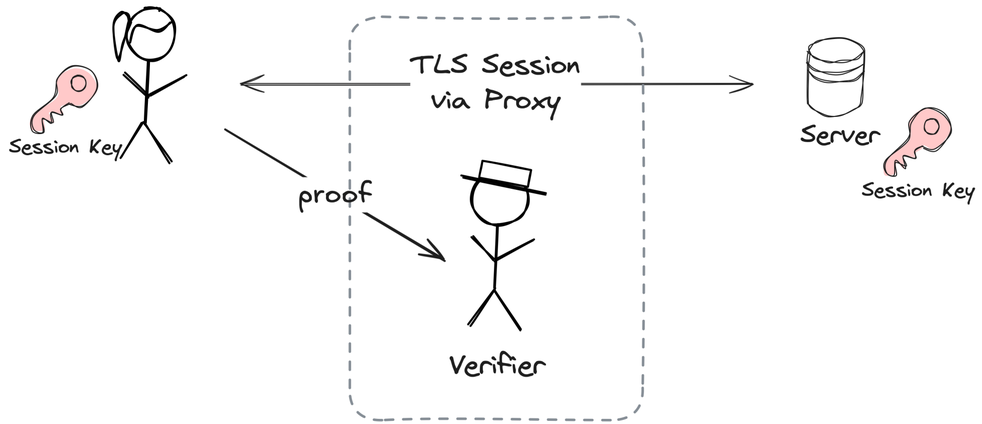
Variant: proving TLS

- TLS uses *symmetric* cryptography (recall MAC)
- Need to prevent the Prover from authenticating arbitrary messages
  - option 1: MPC
  - option 2: Proxy
- Downside: we are now in the **designated verifier** setting

Option 1: MPC



Option 2: Proxy

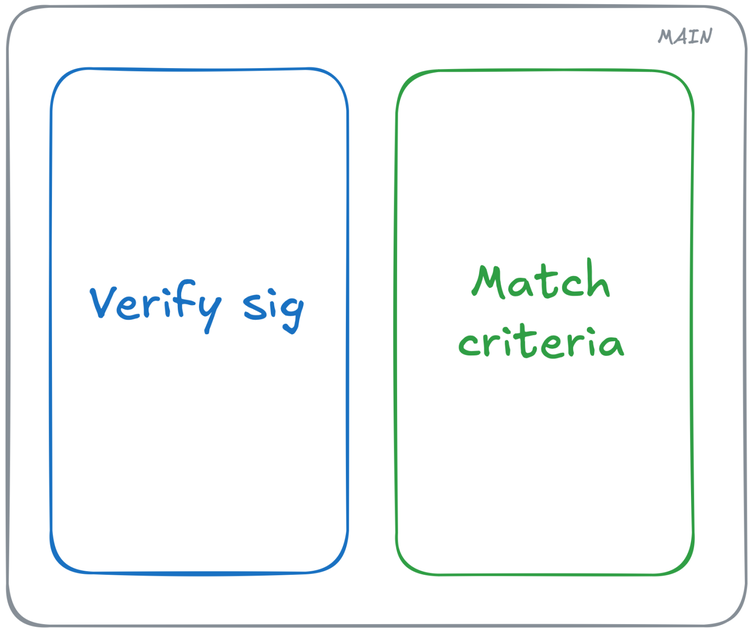


Part 2: what next?

Generating credentials

or how to avoid replication of prover work

- In the real world, proofs will be single-use to prevent **replay attacks**.
- Side effect: this forces the prover to **prove the circuit every time**.
- Verifying the signature is redundant work.



Generating credentials

or how to avoid replication of prover work

- In the real world, proofs will be single-use to prevent **replay attacks**.
- Side effect: this forces the prover to **prove the circuit every time**.
- Verifying the signature is redundant work.

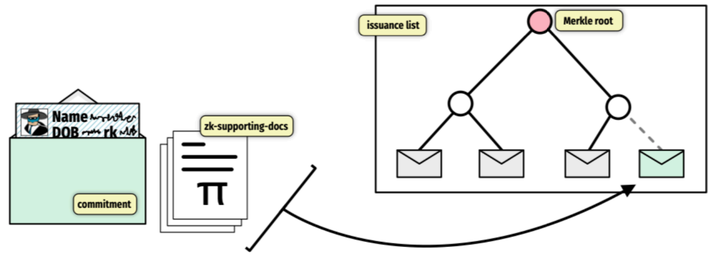


Figure 1: A credential in `zk-creds` is issued by adding it to a Merkle tree after (optionally) presenting zk-supporting-documentation to justify issuance.

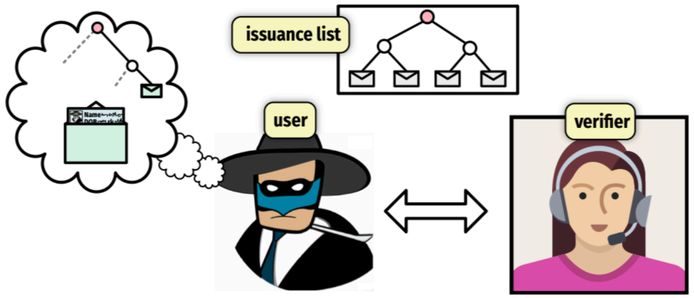


Figure 2: To show a credential in `zk-creds`, the prover uses knowledge of their credential opening and the position of the credential in the issuance list to construct a zero-knowledge proof. The verifier need only know the issuance list root.

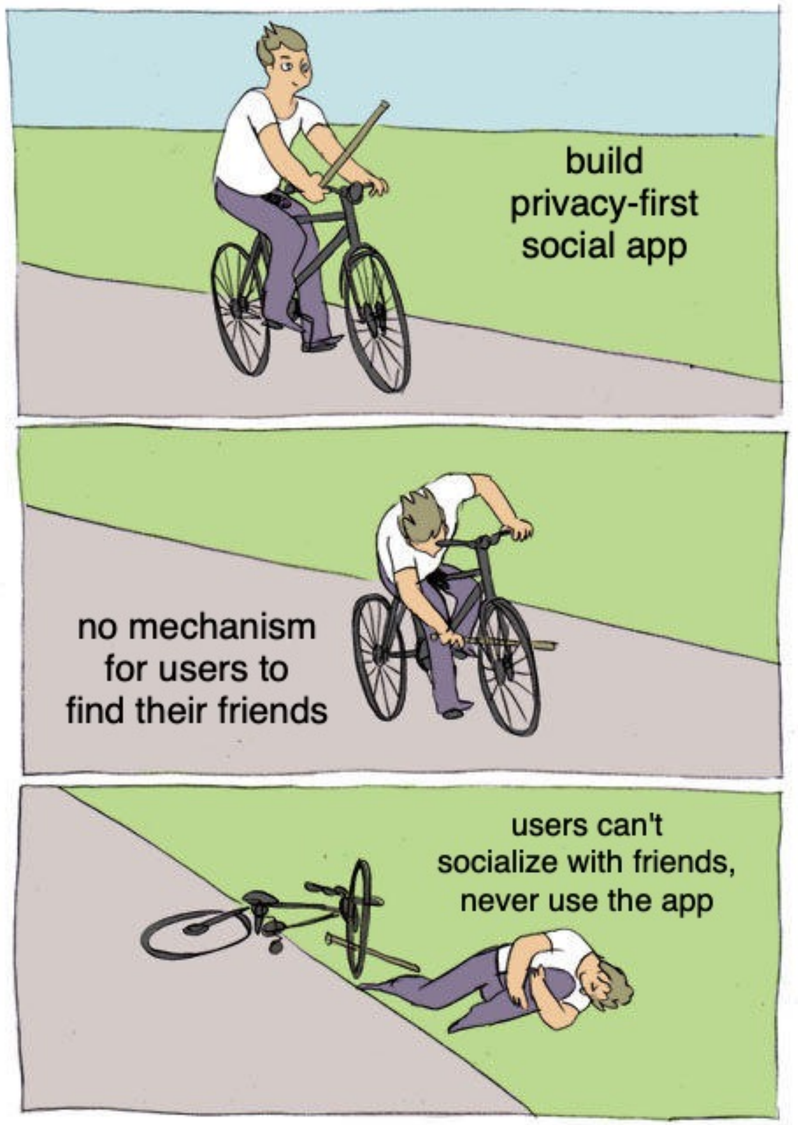
Idea (zk-creds, S&P ‘23): use a smart contract to verify the proof of signature once, then verify criteria separately.

diagram taken from the zk-creds paper

Making usable systems

…even if there is no observable data!

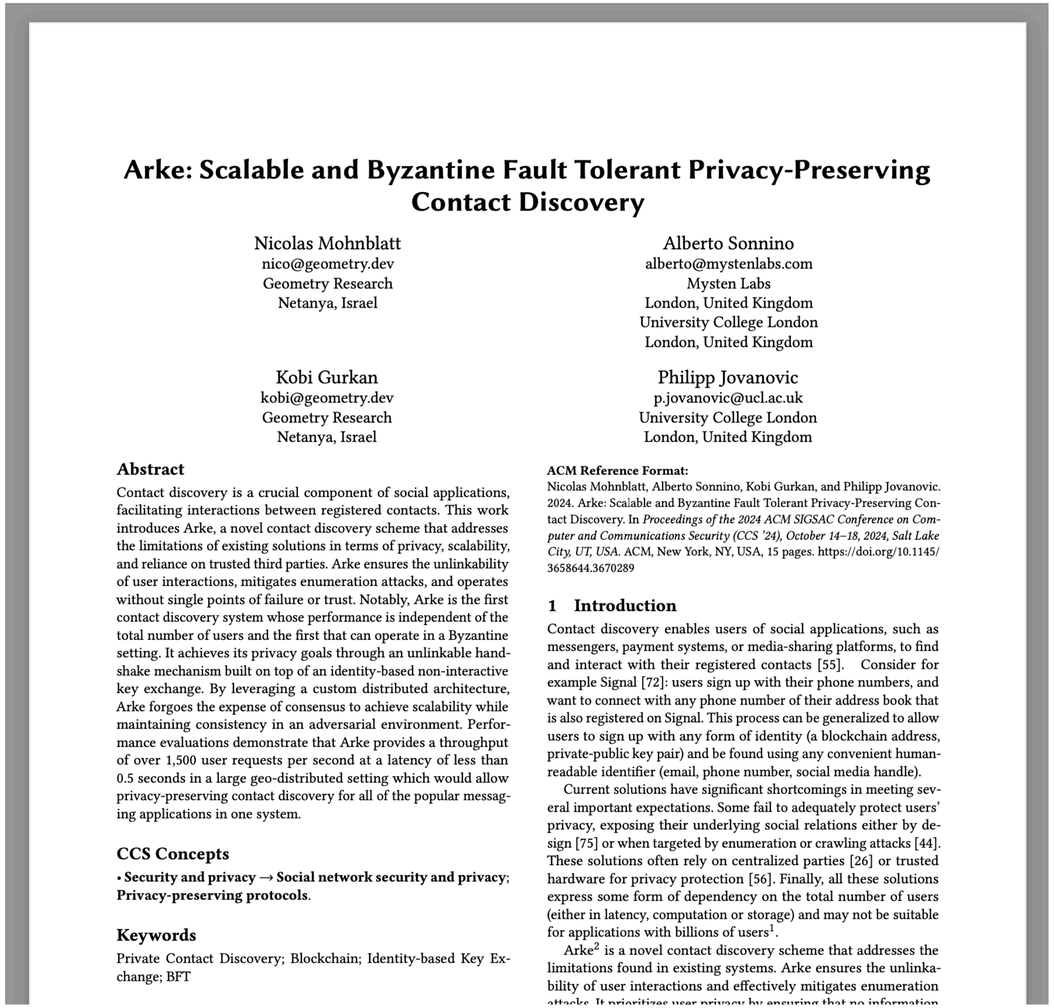
- Consider building a new private social network
  - profiles are private
  - social graph is hidden
- How can users find their friends?



Making usable systems

...even if there is no observable data!

- Arke: privacy-preserving contact discovery scheme
- Uses identity-based cryptography
- Users can use any of these ZK x ID systems to privately and legitimately register their existing identity (email, X handle, etc)



Arke: Scalable and Byzantine Fault Tolerant Privacy-Preserving Contact Discovery

Nicolas Mohnblatt
nico@geometry.dev
Geometry Research
Netanya, Israel

Alberto Sonnino
alberto@mystenlabs.com
Mysten Labs
London, United Kingdom
University College London
London, United Kingdom

Kobi Gurkan
kobi@geometry.dev
Geometry Research
Netanya, Israel

Philipp Jovanovic
p.jovanovic@ucl.ac.uk
University College London
London, United Kingdom

Abstract

Contact discovery is a crucial component of social applications, facilitating interactions between registered contacts. This work introduces Arke, a novel contact discovery scheme that addresses the limitations of existing solutions in terms of privacy, scalability, and reliance on trusted third parties. Arke ensures the unlinkability of user interactions, mitigates enumeration attacks, and operates without single points of failure or trust. Notably, Arke is the first contact discovery system whose performance is independent of the total number of users and the first that can operate in a Byzantine setting. It achieves its privacy goals through an unlinkable handshake mechanism built on top of an identity-based non-interactive key exchange. By leveraging a custom distributed architecture, Arke forgoes the expense of consensus to achieve scalability while maintaining consistency in an adversarial environment. Performance evaluations demonstrate that Arke provides a throughput of over 1,500 user requests per second at a latency of less than 0.5 seconds in a large geo-distributed setting which would allow privacy-preserving contact discovery for all of the popular messaging applications in one system.

CCS Concepts

• **Security and privacy → Social network security and privacy**; **Privacy-preserving protocols**.

Keywords

Private Contact Discovery; Blockchain; Identity-based Key Exchange; BFT

ACM Reference Format:
Nicolas Mohnblatt, Alberto Sonnino, Kobi Gurkan, and Philipp Jovanovic. 2024. Arke: Scalable and Byzantine Fault Tolerant Privacy-Preserving Contact Discovery. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3658644.3670289

1 Introduction

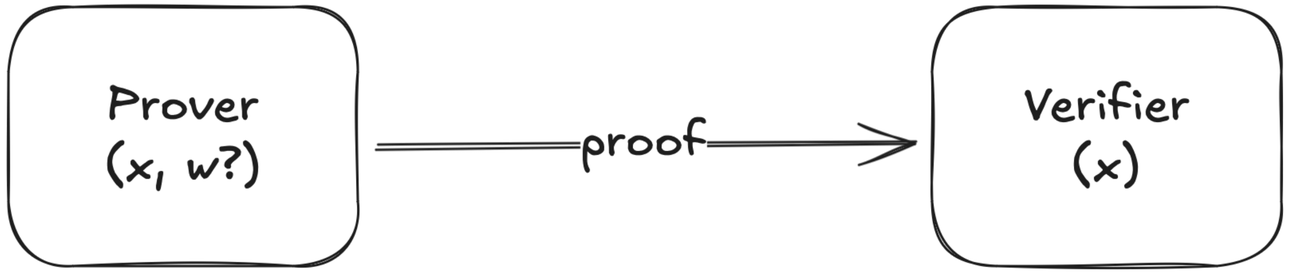
Contact discovery enables users of social applications, such as messengers, payment systems, or media-sharing platforms, to find and interact with their registered contacts [55]. Consider for example Signal [72]: users sign up with their phone numbers, and want to connect with any phone number of their address book that is also registered on Signal. This process can be generalized to allow users to sign up with any form of identity (a blockchain address, private-public key pair) and be found using any convenient human-readable identifier (email, phone number, social media handle).

Current solutions have significant shortcomings in meeting several important expectations. Some fail to adequately protect users' privacy, exposing their underlying social relations either by design [75] or when targeted by enumeration or crawling attacks [44]. These solutions often rely on centralized parties [26] or trusted hardware for privacy protection [56]. Finally, all these solutions express some form of dependency on the total number of users (either in latency, computation or storage) and may not be suitable for applications with billions of users[1].

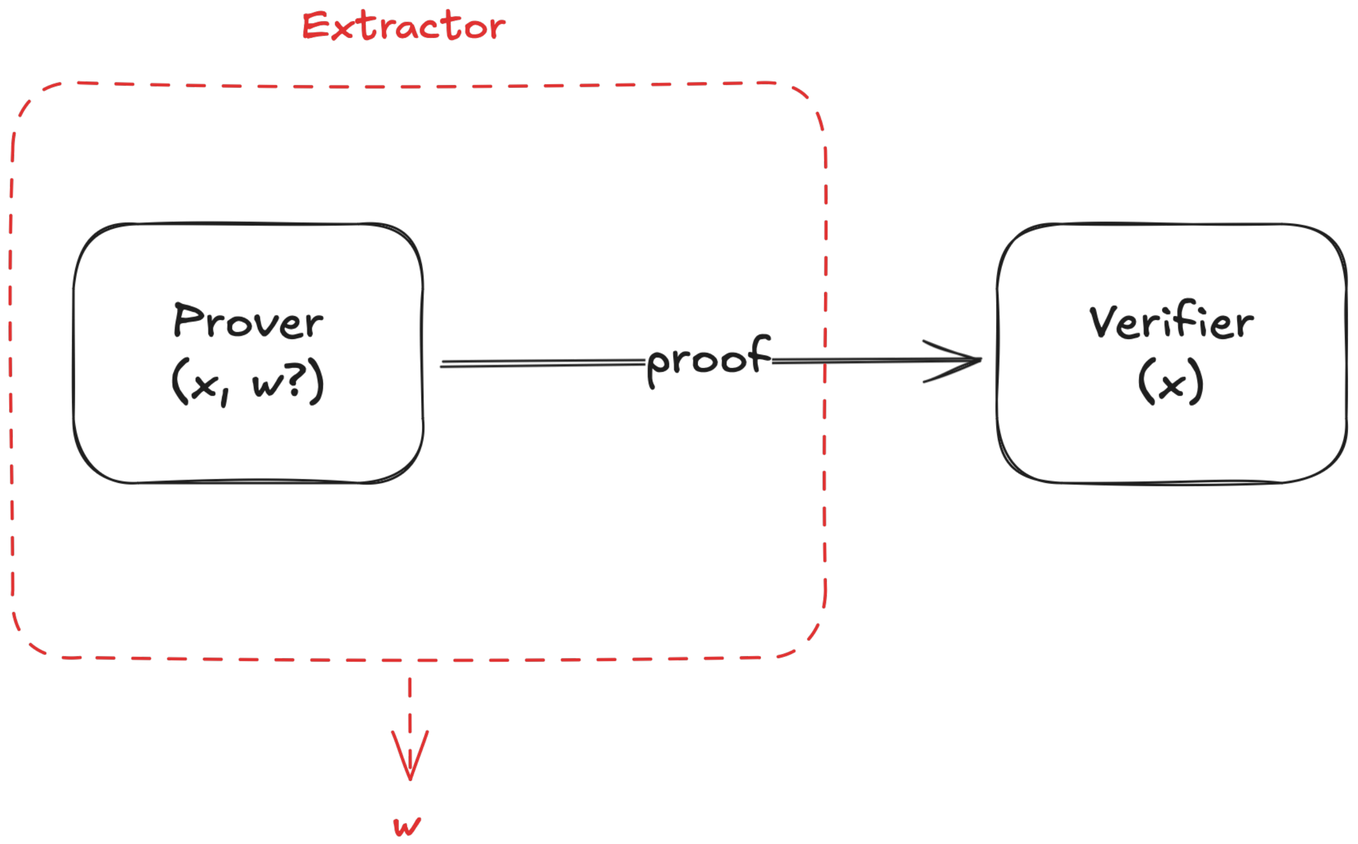
Arke[2] is a novel contact discovery scheme that addresses the limitations found in existing systems. Arke ensures the unlinkability of user interactions and effectively mitigates enumeration attacks. It prioritizes user privacy by ensuring that no information

Bonus: the security of SNARKs in the presence of signing oracles

Proof of Knowledge

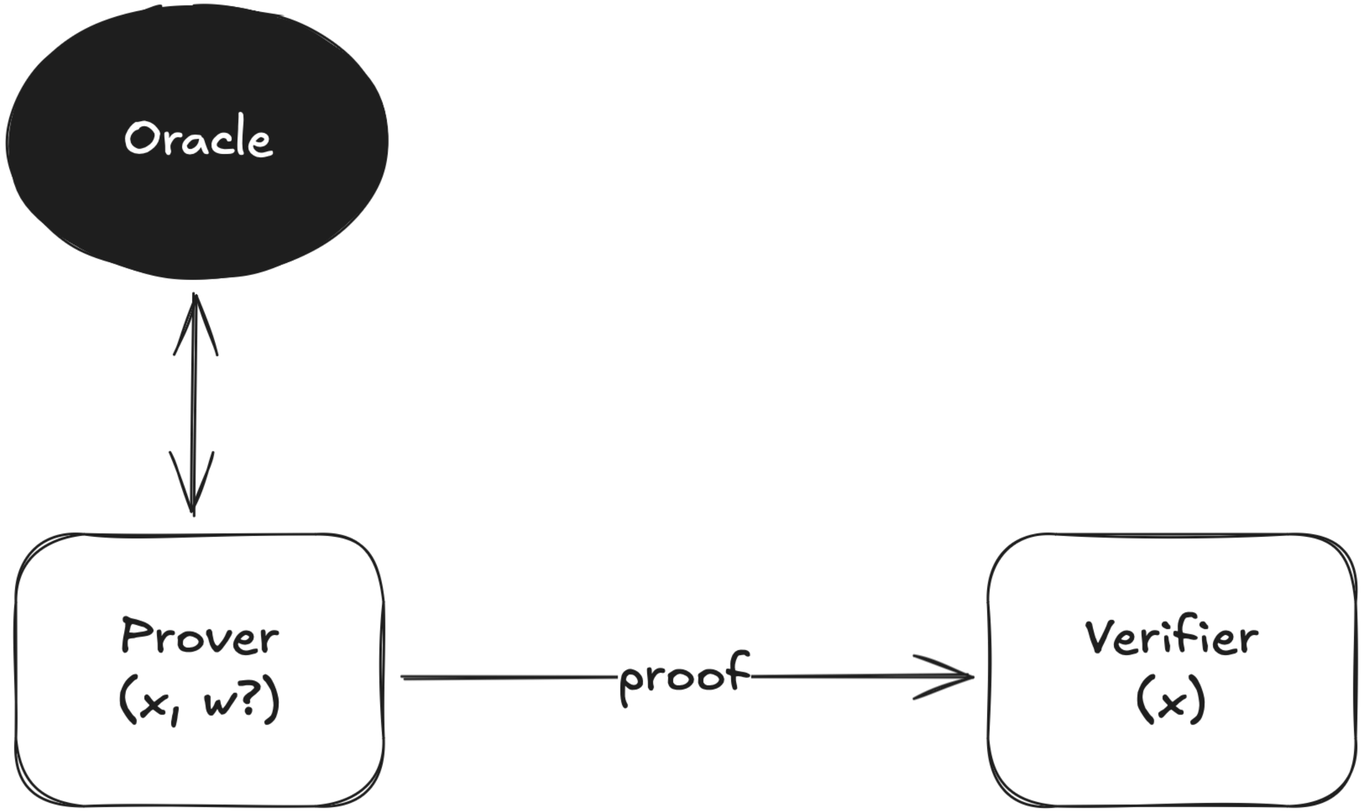


Proof of Knowledge



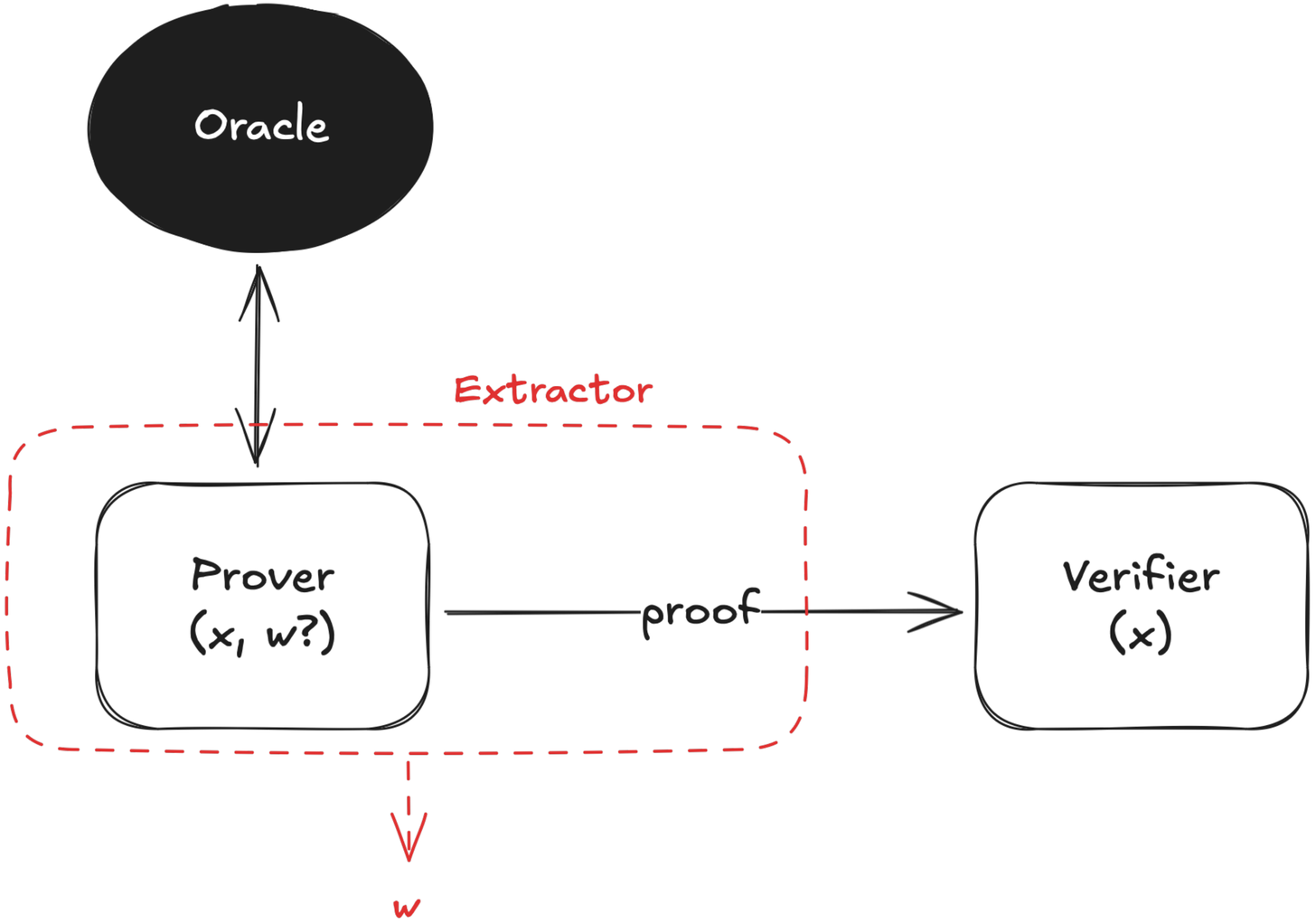
Proof of Knowledge

in the presence of oracles



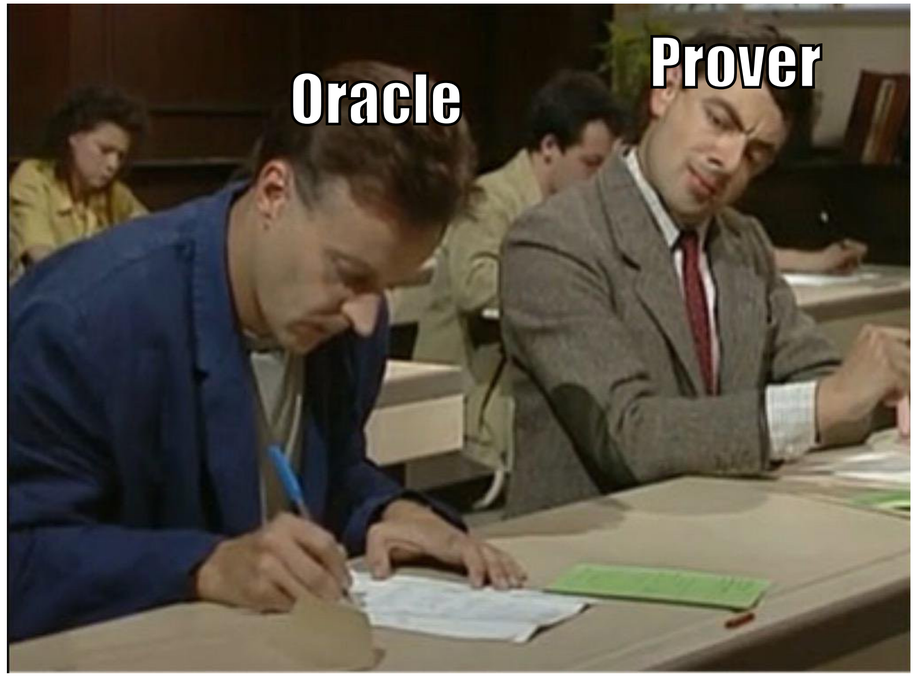
Proof of Knowledge

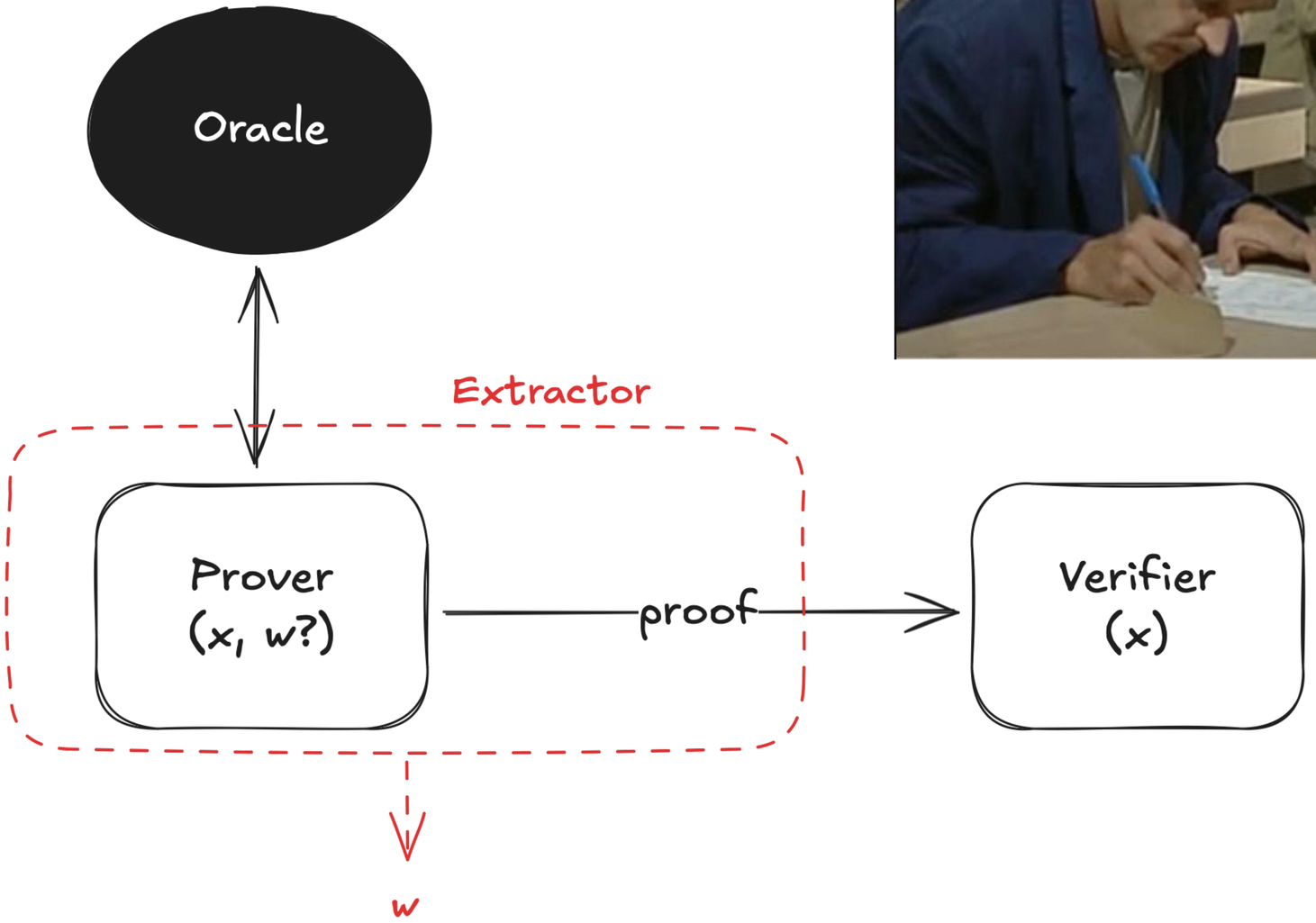
in the presence of oracles



Proof of Knowledge

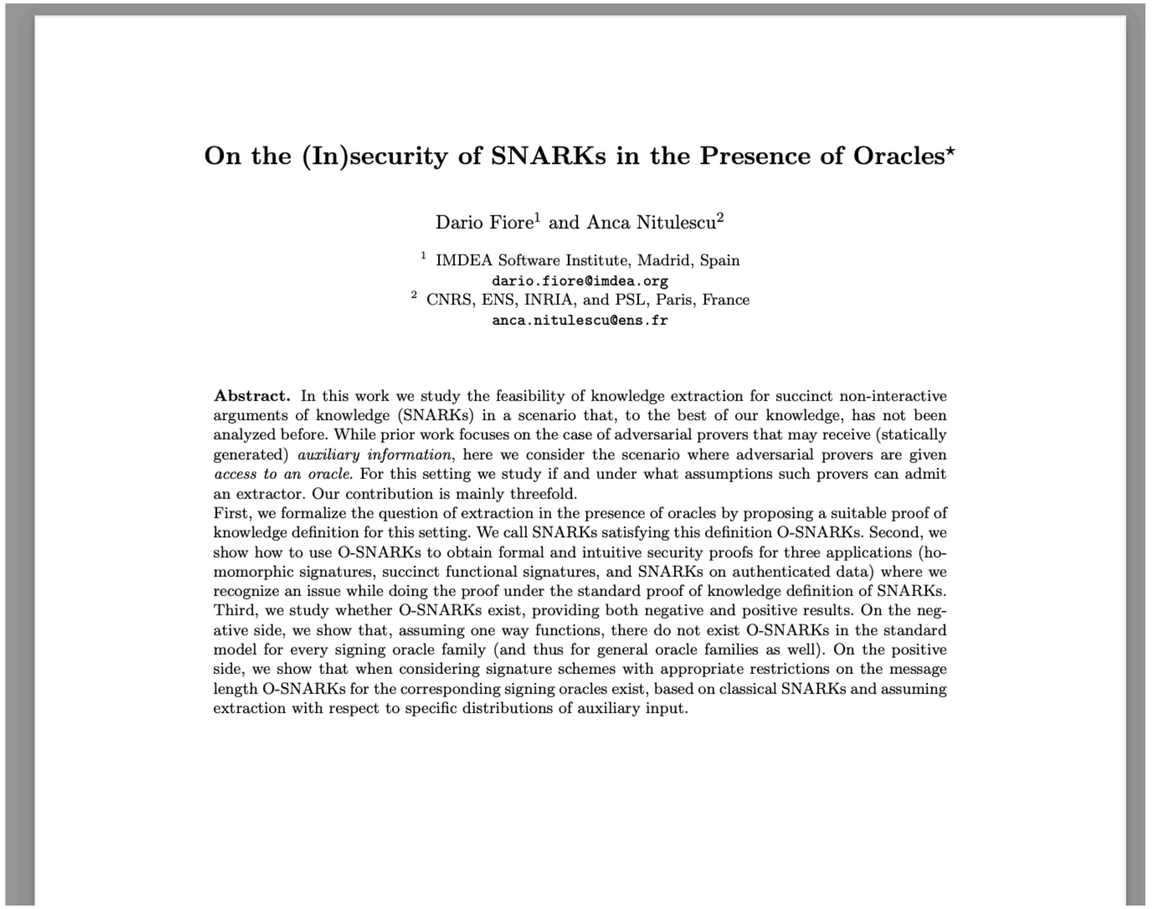
in the presence of oracles





O-SNARKs

- O-SNARK for 🅾 = SNARK that is secure **in the presence of an oracle** from the family 🅾
  - e.g., SNARK in the presence of a proving oracle (= simulation extractability), or SNARK in the presence of a signing oracle
- Notion introduced by Fiore and Nitulescu (TCC ’16)



On the (In)security of SNARKs in the Presence of Oracles*

Dario Fiore[1] and Anca Nitulescu[2]

[1] IMDEA Software Institute, Madrid, Spain
dario.fiore@imdea.org
[2] CNRS, ENS, INRIA, and PSL, Paris, France
anca.nitulescu@ens.fr

Abstract. In this work we study the feasibility of knowledge extraction for succinct non-interactive arguments of knowledge (SNARKs) in a scenario that, to the best of our knowledge, has not been analyzed before. While prior work focuses on the case of adversarial provers that may receive (statically generated) *auxiliary information*, here we consider the scenario where adversarial provers are given *access to an oracle*. For this setting we study if and under what assumptions such provers can admit an extractor. Our contribution is mainly threefold.
First, we formalize the question of extraction in the presence of oracles by proposing a suitable proof of knowledge definition for this setting. We call SNARKs satisfying this definition O-SNARKs. Second, we show how to use O-SNARKs to obtain formal and intuitive security proofs for three applications (homomorphic signatures, succinct functional signatures, and SNARKs on authenticated data) where we recognize an issue while doing the proof under the standard proof of knowledge definition of SNARKs. Third, we study whether O-SNARKs exist, providing both negative and positive results. On the negative side, we show that, assuming one way functions, there do not exist O-SNARKs in the standard model for every signing oracle family (and thus for general oracle families as well). On the positive side, we show that when considering signature schemes with appropriate restrictions on the message length O-SNARKs for the corresponding signing oracles exist, based on classical SNARKs and assuming extraction with respect to specific distributions of auxiliary input.

O-SNARKs

- Bad news: O-SNARK do not exist in the standard model for ALL oracles.
- However, some oracles can be shown to be benign
  - Schnorr signatures in “Giving Extractors what they need” by Ariel Gabizon
- Good news in other models:
  - in the ROM, SNARKs from the Micali and BCS transform are O-SNARKs
  - in the AGM/GGM, we need to assume that the oracle does not interfere with the group
  - for knowledge assumptions (e.g., as in Groth16), we can formulate the same assumption in the presence of an oracle from the family 🔮

Summary

- We are all building the same circuit: verify signature + match criteria
- Next up: caching credentials + using these proofs to bootstrap systems
- O-SNARKs

References 1/2

- ZK x ID applications
  - ZK Email https://prove.email/
  - Web Proofs https://docs.pluto.xyz/web-proofs
  - OpenPubKey [HMF+24] https://eprint.iacr.org/2023/296
  - zkLogin [BCJ+24] https://arxiv.org/abs/2401.11735
  - Aptos Keyless https://github.com/aptos-labs/aptos-core/tree/main/keyless/circuit
  - OpenPassport https://www.proofofpassport.com/
  - verITAS [DCB24] https://eprint.iacr.org/2024/1066
  - Reef [AIM+23] https://eprint.iacr.org/2023/1886

References 2/2

- TLS Oracles
  - DECO [SKM+19] https://arxiv.org/abs/1909.00938
  - TLS Notary https://docs.tlsnotary.org/
  - Proxying is Enough [LJSK24] https://eprint.iacr.org/2024/733
- What's next
  - zk-creds [RWGM22] https://eprint.iacr.org/2022/878
  - Arke: Scalable and Byzantine Fault Tolerant Privacy-Preserving Contact Discovery [**M**SGJ23] https://eprint.iacr.org/2023/1218
- O-SNARKs
  - On the (in)-security of SNARK in the presence of oracles [FN16] https://eprint.iacr.org/2016/112
  - Giving extractors what they need [Gab18] https://medium.com/@arielgabizon/giving-extractors-what-they-need-d17b7b698b0a
  - zkSNARKs in the ROM with Unconditional UC-Security [CF24] https://eprint.iacr.org/2024/724