black hat MIDDLE EAST AND AFRICA

ZERO-KNOWLEDGE PROOFS: SECURING THE FUTURE OF CRYPTO

15 – 17 NOVEMBER 2022

RIYADH FRONT EXHIBITION CENTRE

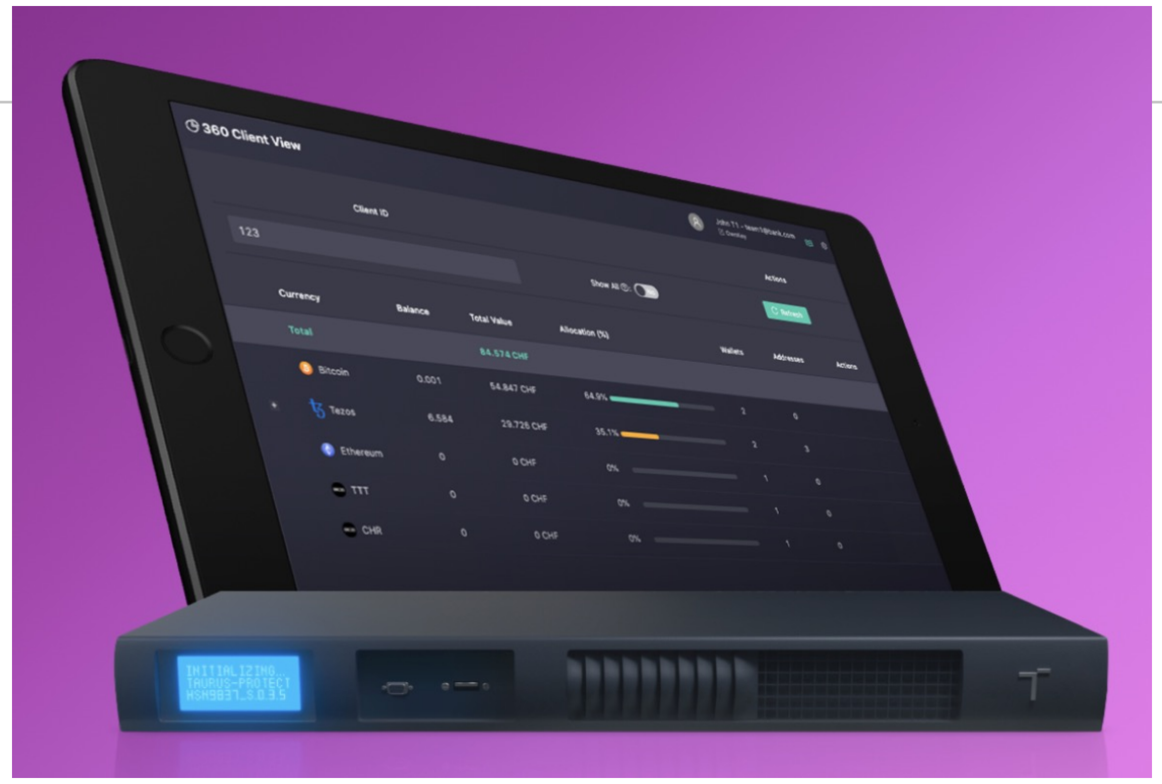
SAUDI ARABIA

JP Aumasson – CSO @ Taurus

/me

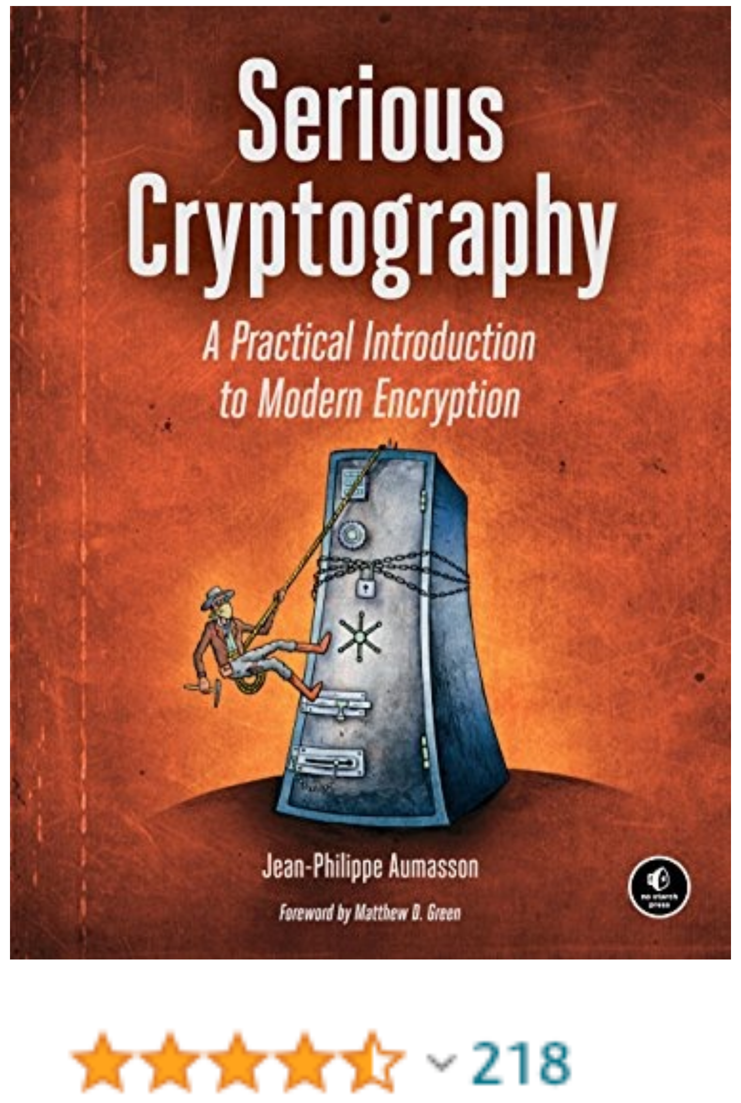
Co-founder & CSO of a Swiss fintech (**Taurus**)

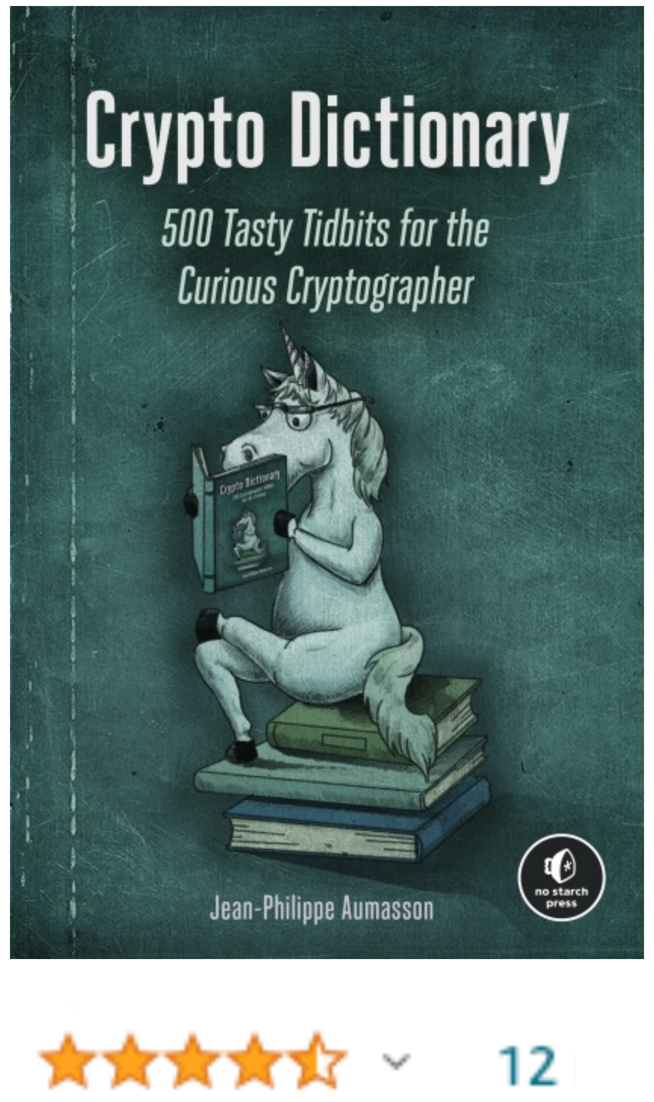
- High-assurance crypto custody tech https://taurushq.com
- Used by banks to protect and manage their BTC/ETH/etc.
- Running a regulated exchange https://t-dx.com



Cryptography and vulnerability research since ~2006

- Designed crypto in the Linux kernel, Bitcoin, etc. (SipHash, BLAKE2, BLAKE3)
- Wrote books about cryptography





https://aumasson.jp. https://twitter.com/veorq

Zero-knowledge proof?

NOT “zero-knowledge architecture” or“zero-trust”

NOT “zero-knowledge encryption” (marketing term for client-side encryption)

vixentael

Dec 28, 2017 · 7 min read · Listen

Zero Knowledge Architectures for Mobile Applications

WHAT IS ZERO-KNOWLEDGE ENCRYPTION, AND HOW DOES IT WORK?

WRITTEN BY MATT AHLGREN RESEARCHED BY WSR TEAM | JUNE 22, 2022 | IN CLOUD STORAGE, PASSWORD MANAGERS

Zero-knowledge encryption is arguably one of the **most secure ways of protecting your data.** In a nutshell, it means that cloud storage or backup providers know nothing (i.e. have “zero-knowledge”) about the data you store on their servers.

Zero-knowledge proof?

NOT “zero-knowledge architecture” or“zero-trust”

NOT “zero-knowledge encryption” (marketing term for client-side encryption)

A class of **cryptography protocols**…

- Between a *prover* and a *verifier*
- Which can be *non-interactive*
- Known since the 1980s, only recently used in practice at scale (*zkSNARKS*)

The Knowledge Complexity of Interactive Proof-Systems

(Extended Abstract)

Shafi Goldwasser	Silvio Micali	Charles Rackoff
MIT	MIT	University of Toronto

Zero-knowledge proof?

NOT “zero-knowledge architecture” or“zero-trust”

NOT “zero-knowledge encryption” (marketing term for client-side encryption)

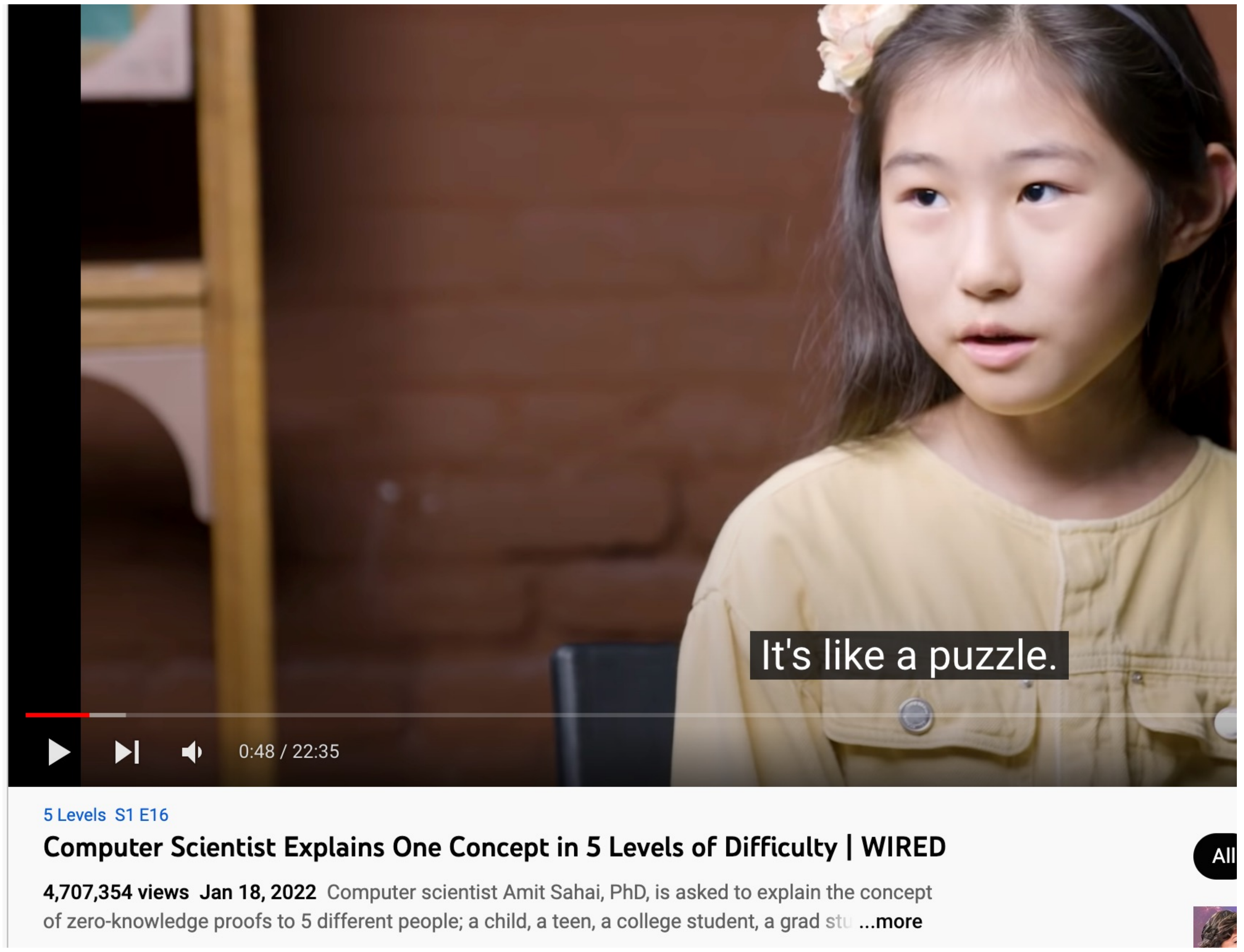
Zero-knowledge proof?

NOT "zero-knowledge architecture" or"zero-trust"

NOT "zero-knowledge encryption" (marketing term for client-side encryption)

A class of **cryptography protocols**…

- Between a *prover* and a *verifier*
- Which can be *non-interactive*
- Known since the 1980s, only recently used in practice at scale (*zkSNARKS*)



https://www.youtube.com/watch?v=fOGdb1CTu5c

The simplest ZK proof

Schnorr's proof of knowledge of discrete logarithm ($\mathbf{x}$ in $y = g^{\mathbf{x}} \bmod p$)

Prover

Verifier

Pick a random r, send $t = g^r \bmod p$ →

Send a random c ←

Send $s = r + c\mathbf{x} \bmod p$ →

Verify that $g^s = t \times y^c$

It works because $g^s = g^{r + c\mathbf{x}} = g^r \times (g^{\mathbf{x}})^c = t \times y^c$

Zero-knowledge proofs applications

- **Privacy** of payments, and of general computation (with "zkVMs")
- **Scalability** – via "**ZK rollups**", to preventing re-computation of smart contracts
- **Storage** proofs, as in Filecoin's proofs of spacetime

> Our proof-of-concept system allows the Police to prove to the public that the DNA profile of a Presidential Candidate does not appear in the forensic DNA profile database maintained by the Police. The proof, which is generated by the Police, relies on no external trusted party, and reveals no further information about the contents of the database, nor about the candidate's profile. In particular, no DNA information is disclosed to any party outside the Police. The proof is shorter than the size of the DNA database, and verified faster than the time needed to examine that database naïvely.

https://eprint.iacr.org/2018/046

Vibrant ecosystem

Examples of major projects in the ZK space, many other initiatives and research groups

aleo.org

anoma.network

arkworks.rs

aztec.network

celo.org

protocol.ai

starkware.co

z.cash

This talk

Focus on **zkSNARKs**, a class of zero-knowledge proof systems

- *Fully succinct* = ***constant*** proof size and ***linear*** verification time (wrt program size)

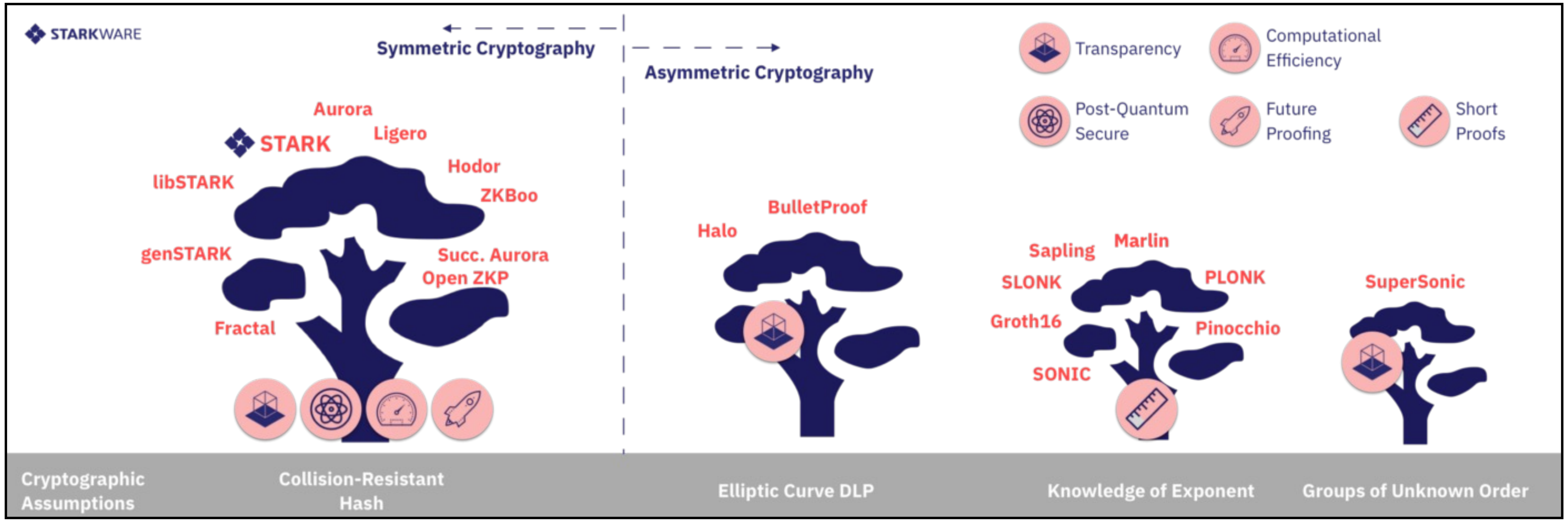
Based on my experience looking for bugs in

- **Groth16**, used in Zcash, Filecoin, and many others
- **Marlin**, a universal zkSNARK, used in Aleo
- **Circuits**, and in many other related crypto

zkSNARKs and friends

zkSNARKs are **not the only proof systems** used in practice

- STARKs: no trusted setup, proof size not constant, post-quantum (StarkWare)
- Bulletproofs: simpler, no trusted setup, but slower verification (Monero)



STARK = Scalable, Transparent ARgument of Knowledge

zkSNARKs' golden age: 2018-2020

Zero-knowledge proof (ZKP) systems

ZKP System	Publication year	Protocol	Transparent	Universal	Plausibly Post-Quantum Secure	Programming Paradigm
Pinocchio[31]	2013	zk-SNARK	No	No	No	Procedural
Geppetto[32]	2015	zk-SNARK	No	No	No	Procedural
TinyRAM[33]	2013	zk-SNARK	No	No	No	Procedural
Buffet[34]	2015	zk-SNARK	No	No	No	Procedural
ZoKrates[35]	2018	zk-SNARK	No	No	No	Procedural
xJsnark[36]	2018	zk-SNARK	No	No	No	Procedural
vRAM[37]	2018	zk-SNARG	No	Yes	No	Assembly
vnTinyRAM[38]	2014	zk-SNARK	No	Yes	No	Procedural
MIRAGE[39]	2020	zk-SNARK	No	Yes	No	Arithmetic Circuits
Sonic[40]	2019	zk-SNARK	No	Yes	No	Arithmetic Circuits
Marlin[41]	2020	zk-SNARK	No	Yes	No	Arithmetic Circuits
PLONK[42]	2019	zk-SNARK	No	Yes	No	Arithmetic Circuits
SuperSonic[43]	2020	zk-SNARK	Yes	Yes	No	Arithmetic Circuits
Bulletproofs[44]	2018	Bulletproofs	Yes	Yes	No	Arithmetic Circuits
Hyrax[45]	2018	zk-SNARK	Yes	Yes	No	Arithmetic Circuits
Halo[46]	2019	zk-SNARK	Yes	Yes	No	Arithmetic Circuits
Virgo[47]	2020	zk-SNARK	Yes	Yes	Yes	Arithmetic Circuits
Ligero[48]	2017	zk-SNARK	Yes	Yes	Yes	Arithmetic Circuits
Aurora[49]	2019	zk-SNARK	Yes	Yes	Yes	Arithmetic Circuits
zk-STARK[50]	2019	zk-STARK	Yes	Yes	Yes	Assembly
Zilch[30] [51]	2021	zk-STARK	Yes	Yes	Yes	Object-Oriented

https://www.wikiwand.com/en/Zero-knowledge_proof

Why study zkSNARKs security?

A critical component of decentralised platforms (L2 protocols, private transactions):

- Complexity + Novelty => Non-trivial **bugs**
- A lot **at stake** ($$$, user data, user privacy)

Why study zkSNARKs security?

A critical component of decentralised platforms (L2 protocols, private transactions):

- Complexity + Novelty => Non-trivial **bugs**
- A lot **at stake** ($$$, user data, user privacy)

As a cryptographer since ~2005, **the most interesting** crypto I've seen:

- Intricate constructions with non-trivial components
- "Simple but complex" – non-interactive, but many moving parts
- "Multidimensional" way to reason about security
- "Real-worldness": not just papers – "code is specs"

What's zkSNARKs security?

<u>Soundness</u>: Invalid proofs should always be rejected

- Most obvious attack, often the *highest risk* in practice:
- Forging, altering, replaying valid proofs should be impossible

What's zkSNARKs security?

Soundness: Invalid proofs should always be rejected

- Most obvious attack, often the *highest risk* in practice:
- Forging, altering, replaying valid proofs should be impossible

Zero-knowledge: Proofs should not leak secret information

- In practice, succinct proofs of large programs can leak only little data

What's zkSNARKs security?

<u>Soundness</u>: Invalid proofs should always be rejected

- Most obvious attack, often the *highest risk* in practice:
- Forging, altering, replaying valid proofs should be impossible

<u>Zero-knowledge</u>: Proofs should not leak secret information

- In practice, succinct proofs of large programs can leak only little data

<u>Completeness</u>: Valid proofs should always be accepted

- Often a DoS/usability risk that may be further exploited
- All programs/circuits supported should be correctly processed

Who can find bugs?

A. Developers of the code (manually or via testing)
B. Developers of other projects' code
C. External auditors of the code
D. Users of the code, accidentally 😇
E. External "attackers" 😈

Security goal: you want A|B|C to find bugs before D|E

Bug hunting challenges

Practical zkSNARKs are recent, thus auditors often have

- Limited **experience** auditing zkSNARKs
- Limited **knowledge** of the theory and of implementations' tricks
- Limited **"checklist"** of bugs and bug classes
- Limited **tooling** and methodologies
- Limited **documentation** from the projects

How to make useful work nonetheless?

New crypto, new approach

- More **collaboration** with the devs/designers (joint review sessions, Q&As, etc.)
- More **threat analysis**, to understand the application's unique/novel risks
- Practical **experience**: writing PoCs, circuits, proof systems, etc.
- Learn **previous failures**, for example from…
  - Public disclosures and exploits
  - Other audit reports
  - Issue trackers / PRs
  - Community

General workflow, and failure *examples*

Computation

↓

Circuit definition

↓

Arithmetization

↓

Non-interactive proof

↓

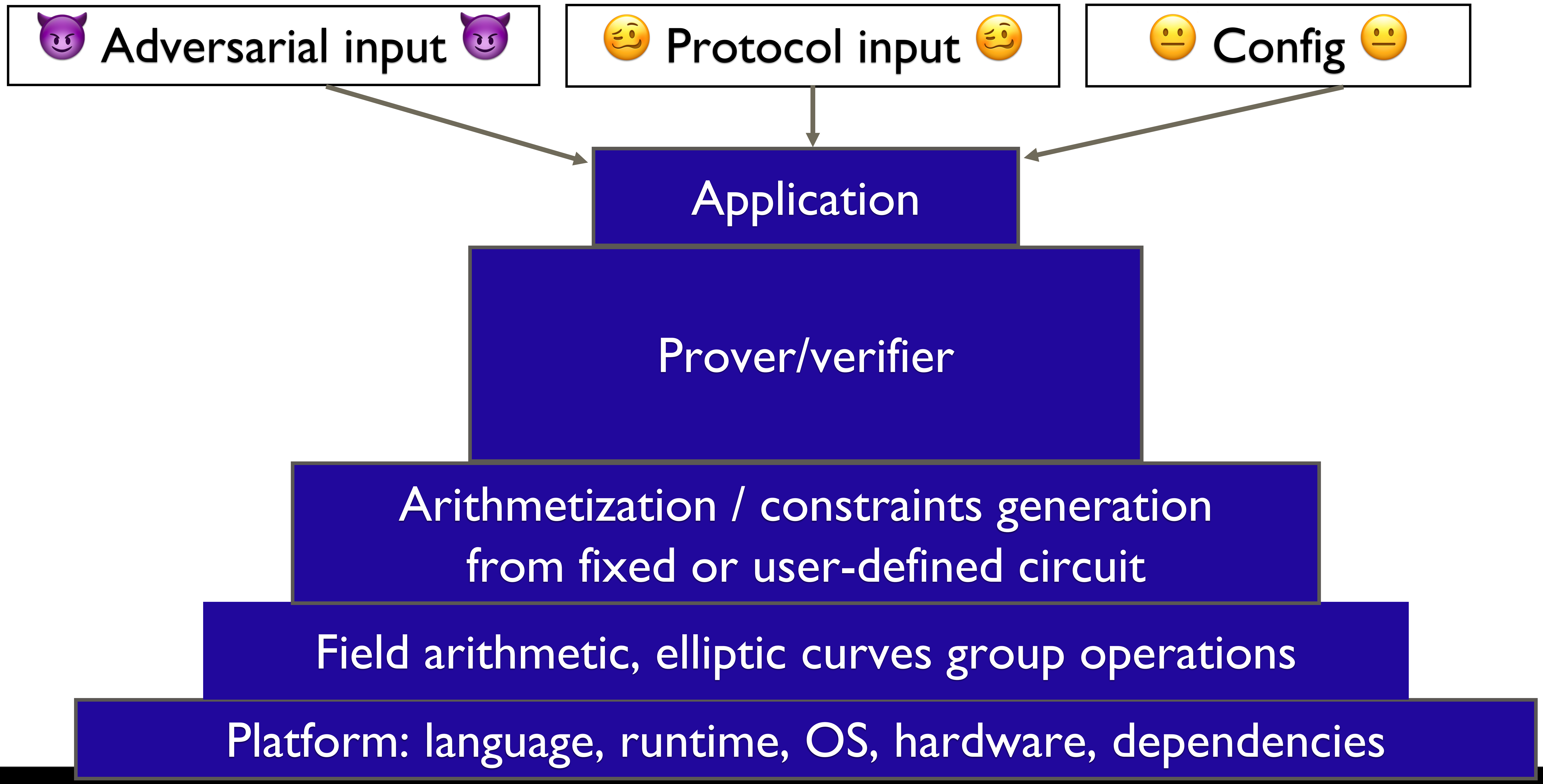
Usage in applications

General workflow, and failure *examples*

Computation	The program's logic is not secure
↓ Circuit definition	The circuit is not equivalent to the program
↓ Arithmetization	The constraint system fails to enforce a constraint
↓ Non-interactive proof	Insecure choice of primitives/parameters/properties
↓ Usage in applications	The application allows replays of previous proofs

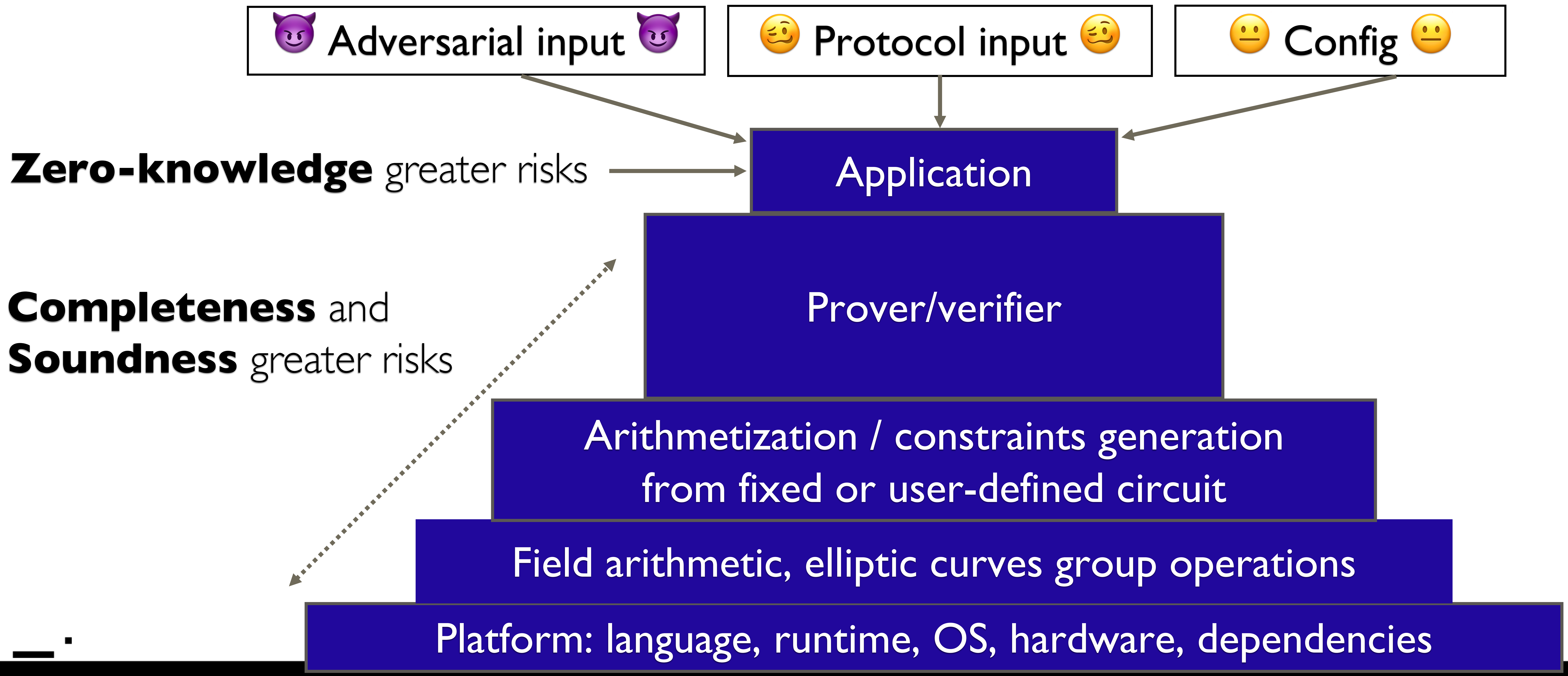
Need structure/methodology..

A failure in a **lower layer** can jeopardise the security of all upper layers



What to look for, and where?

A failure in a **lower layer** can jeopardise the security of all upper layers



Divide and conquer..

A failure in a **subcomponent** can jeopardise the security of all upper layers

Understand composability conditions..

Security 101: **Input validation** must be defined, implemented, and tested

😈 Adversarial input 😈 | 🥴 Protocol input 🥴 | 😐 Config 😐

Key management, Testing **Application** Interface, Side channels

Contracts between components must be defined to prevent **insecure composition**

Example: which component is responsible for **group membership** checks?

Elliptic curves, Pairings, Hash functions, PRF, Algebraic commitment
Randomness, Merkle trees **Prover/verifier** Linear algebra, Multi-exp.
Polynomial commitments, Fiat-Shamir transforms, etc. etc.

Real-word crypto bugs..

Soundness – Field arithmetic (1/n)

Vulnerability allowing double spend #16

Closed **poma** opened this issue on 26 Jul 2019 · 2 comments

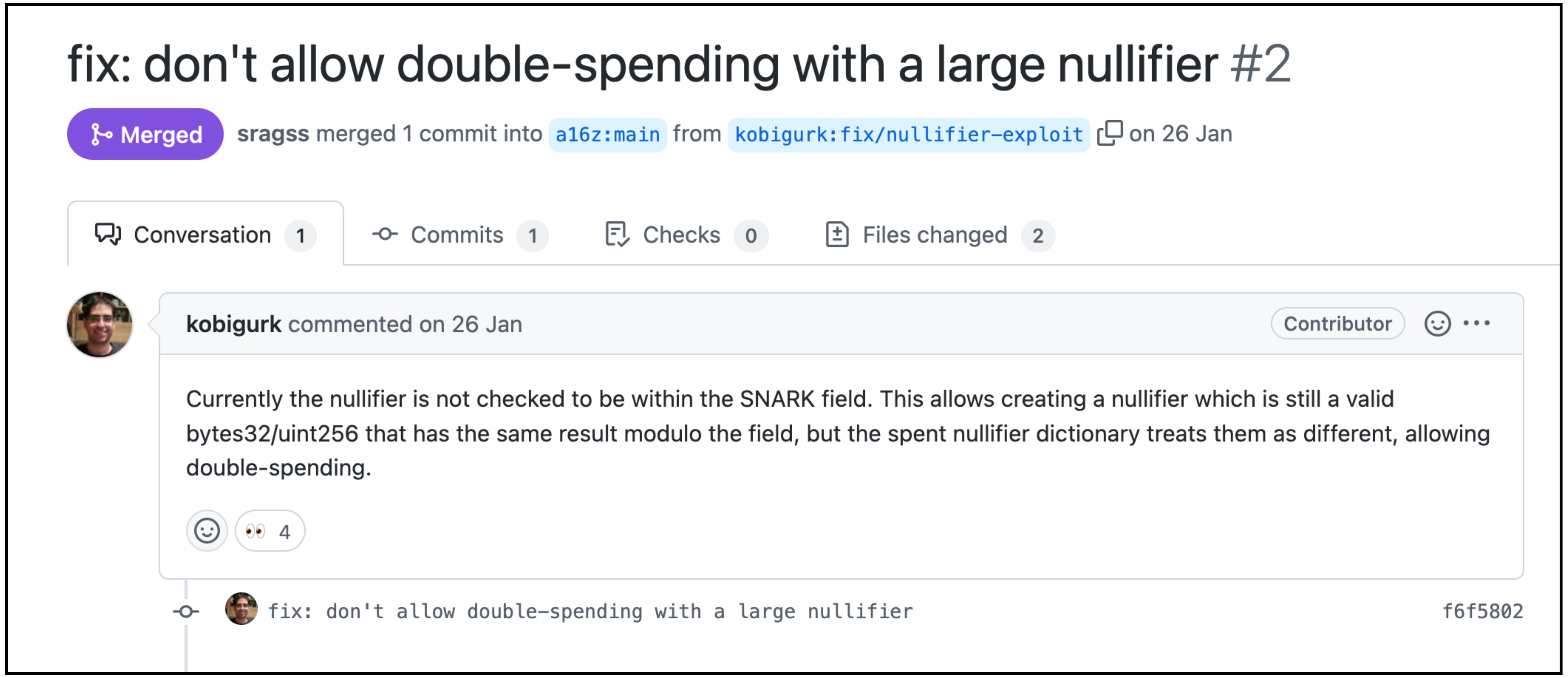
poma commented on 26 Jul 2019 • edited

Looks like in Semaphore.sol#L83 we don't check that nullifier length is less than field modulus. So `nullifier_hash + 21888242871839275222246405745257275088548364400416034343698204186575808495617` will also pass snark proof verification if it fits into uint256, allowing double spend.

Root cause: Missing overflow check of a nullifier (~ unique ID of a shielded payment)

https://github.com/appliedzkp/semaphore/issues/16

Soundness – Field arithmetic (2/n)



Root cause: Missing overflow check of a nullifier (~ unique ID of a shielded payment)

https://github.com/a16z/zkp-merkle-airdrop-contracts/pull/2

Soundness – Field arithmetic (3/n)

Potential security bug with the zk-SNARK verifier

Closed **weijiekoh** opened this issue on 21 Mar 2020 · 2 comments · Fixed by #43

weijiekoh commented on 21 Mar 2020

Expected Behavior

The `Verifier.verify()` function, not the function that calls it (i.e. `Shield.createMSA()` and `Shield.createPO()`, should require that each public input to the snark is less than the scalar field:

Missing overflow check (of a public circuit input)

https://github.com/eea-oasis/baseline/issues/34

Soundness – Field arithmetic (4/n)

```diff
-     // If the values are not in the correct range, the pairing check will fail.
+     // If the values are not in the correct range, the pairing check will fail
+     // because by EIP197 it verfies all input.
      Proof memory proof;
      proof.A = Pairing.G1Point(a[0], a[1]);
      proof.B = Pairing.G2Point([b[0][0], b[0][1]], [b[1][0], b[1][1]]);
@@ -219,7 +221,7 @@ contract Verifier {
      if (input.length + 1 != vk.IC.length) revert Pairing.InvalidProof();
      Pairing.G1Point memory vk_x = vk.IC[0];
      for (uint256 i = 0; i < input.length; i++) {
-       if (input[i] >= Pairing.SCALAR_MODULUS) revert Pairing.InvalidProof();
+       // By EIP196 the scalar_mul verifies it's input is in the correct range.
        vk_x = Pairing.addition(vk_x, Pairing.scalar_mul(vk.IC[i + 1], input[i]));
```

Missing overflow check (of a public circuit input)

https://github.com/appliedzkp/semaphore/pull/96/

Soundness – R1CS

Discuss: enforce `mul_by_inverse` #70

Merged **weikengchen** merged 7 commits into `master` from `fix-mul-by-inverse` on 6 Jul

Conversation 12 | Commits 7 | Checks 5 | Files changed 3

weikengchen commented on 4 Jul 2021 • edited — Member

Description

It seems that the `mul_by_inverse` implementation has a soundness issue that the newly allocated `d_inv` does not need to be the inverse of `d` but could be any value. This can be a soundness issue as the `poly` gadgets have used this API.

```
fn mul_by_inverse(&self, d: &Self) -> Result<Self, SynthesisError> {
    let d_inv = if self.is_constant() || d.is_constant() {
        d.inverse()?
    if self.is_constant() || d.is_constant() {
        let d_inv = d.inverse()?;
        Ok(d_inv * self)
    } else {
```

RUSTSEC-2021-0075 History

Flaw in `FieldVar::mul_by_inverse` allows unsound R1CS constraint systems

Field element inverse property not enforced by the constraint system

https://github.com/arkworks-rs/r1cs-std/pull/70

Soundness – Trusted setup (paper)

Background

On March 1, 2018, Ariel Gabizon, a cryptographer employed by the Zcash Company at the time, discovered a subtle cryptographic flaw in the [BCTV14] paper that describes the zk-SNARK construction used in the original launch of Zcash. The flaw allows an attacker to create counterfeit shielded value in any system that depends on parameters which are generated as described by the paper.

This vulnerability is so subtle that it evaded years of analysis by expert cryptographers focused on zero-knowledge proving systems and zk-SNARKs. In an analysis [Parno15] in 2015, Bryan Parno from Microsoft Research discovered a different mistake in the paper. However, the vulnerability we discovered appears to have evaded his analysis. The vulnerability also appears in the subversion zero-knowledge SNARK scheme of [Fuchsbauer17], where an adaptation of [BCTV14] inherits the flaw. The vulnerability also appears in the ADSNARK construction described in [BBFR14]. Finally, the vulnerability evaded the Zcash Company's own cryptography team, which includes experts in the field that had identified several flaws in other parts of the system.

Theoretical flaw in the paper's setup description (sensitive values not cleared)

https://electriccoin.co/blog/zcash-counterfeiting-vulnerability-successfully-remediated/

Soundness – Fiat-Shamir (code and papers)

Coordinated disclosure of vulnerabilities affecting Girault, Bulletproofs, and PlonK

POST APRIL 13, 2022 LEAVE A COMMENT

By Jim Miller

- ZenGo's zk-paillier
- ING Bank's zkrp (deleted)
- SECBIT Labs' ckb-zkp
- Adjoint, Inc.'s bulletproofs
- Dusk Network's plonk
- Iden3's SnarkJS
- ConsenSys' gnark

The Problem

Why is this type of vulnerability so widespread? It really comes down to a combination of ambiguous descriptions in academic papers and a general lack of guidance around these protocols.

The vulnerabilities in one of these proof systems, Bulletproofs, stem from a mistake in the original academic paper, in which the authors recommend an insecure Fiat-Shamir generation. In addition to disclosing these issues to the above repositories, we've also reached out to the authors of Bulletproofs who have now fixed the mistake.

Incomplete Fiat-Shamiring of protocol transcript

https://blog.trailofbits.com/2022/04/13/part-1-coordinated-disclosure-of-vulnerabilities-affecting-girault-bulletproofs-and-plonk/

Zero-knowledge – Application (Zcash, Monero)

Remote Side-Channel Attacks on Anonymous Transactions

Florian Tramèr*
Stanford University
tramer@cs.stanford.edu

Dan Boneh
Stanford University
dabo@cs.stanford.edu

Kenneth G. Paterson
ETH Zürich
kenny.paterson@inf.ethz.ch

We exploit the fact that the time to produce a proof is correlated with the value of the prover's witness. As the witness contains the transaction amount, we expect this amount to be correlated with the proof time. For example, Zcash's proofs decompose the transaction amount into bits and compute an elliptic curve operation for each *non-zero* bit. The proof time is thus strongly correlated with the Hamming weight of the transaction amount, which is in turn correlated with its value.

Abstract: Privacy-focused crypto-currencies, such as Zcash or Monero, aim to provide strong cryptographic guarantees for transaction confidentiality and unlinkability. In this paper, we describe side-channel attacks that let remote adversaries bypass these protections. We present a general class of timing side-channel and traffic-analysis attacks on receiver privacy. These attacks enable an active remote adversary to identify the (secret) payee of any transaction in Zcash or Monero. The attacks violate the privacy goals of these crypto- currencies by exploiting side-channel information leaked by the implementation of different system components. Specifically, we show that a

Timing dependencies exploited to leak secrets and obtain oracles

https://eprint.iacr.org/2020/627.pdf

Conclusions

😌 **Why not be too scared?**

- Robust code and frameworks (e.g. Rust projects such as arkworks and zkcrypto)
- Safe code easier to write with DSLs (Cairo, Leo, Lurk, Noir, etc.)
- Improvement in SDLC security (e.g. slsa.dev, GitHub Advanced Security)
- Relatively narrow attack surface in practice

Conclusions

😌 **Why not be too scared?**

- Robust code and frameworks (e.g. Rust projects such as arkworks and zkcrypto)
- Safe code easier to write with DSLs (Cairo, Leo, Lurk, Noir, etc.)
- Improvement in SDLC security (e.g. slsa.dev, GitHub Advanced Security)
- Relatively narrow attack surface in practice

😱 **Why be scared?**

- Few people understand zkSNARKs, even fewer can find bugs
- Limited maturity level in many ZK/blockchain projects' SDLC
- Lack of tooling (testing, fuzzing, verification)
- More ZK usage => more $$$ at stake => greater RoI for attackers

Conclusions

Learning resources and projects:

- zkproof.org community and events
- zkhack.dev virtual event (next on ***Nov 22***)
- zkvalidator.com initiative
- zeroknowledge.fm podcast
- zkStudyClub video series http://youtu.be/playlist?list=PLj80z0cJm8QHm_9BdZ1BqcGbgE-BEn-3Y
- Bugs writeups such as https://blog.trailofbits.com/2022/04/13/part-1-coordinated-disclosure-of-vulnerabilities-affecting-girault-bulletproofs-and-plonk/

zk-SNARKs: A Gentle Introduction

Anca Nitulescu

https://www.di.ens.fr/~nitulesc/files/Survey-SNARKs.pdf

شكراً

Thank you

JP Aumasson
@veorq

Big thank yous for their help and feedback to:
Aleo, Protocol Labs, Kobi Gurkan, Adrian Hamelink, Daira Hopwood, Daniel Jacob Bilar, David Wong, Lúcás Meier, Mathilde Raynal, Anna Rose