|galois|

ZKPs for Trust in Software and Hardware

James Parker **(james@galois.com), Alex Malozemoff, David Archer**

Galois, Inc.

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0085. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

Background: SIEVE

- Over the past four years, Galois and other teams have been working on the DARPA SIEVE program
- Goal is to improve the utility and efficiency of zero-knowledge proofs
- Teams are split into two categories:
  - Frontends - Produce ZK statements for different applications
  - Backends - Efficiently verify ZK statements generated by frontends
- SIEVE IR - Standardized circuit representation of ZK proofs

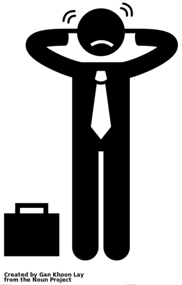
Overview

- Summarize applications we’ve developed during SIEVE
  - Disclosing software vulnerabilities in ZK
  - Improving supply chain security by proving secret, proprietary hardware is correct
  - Using ZK to prove compliance with tax incentives
- Reflect on how these technologies may impact future policies

Challenges of vulnerability disclosure

White Hat

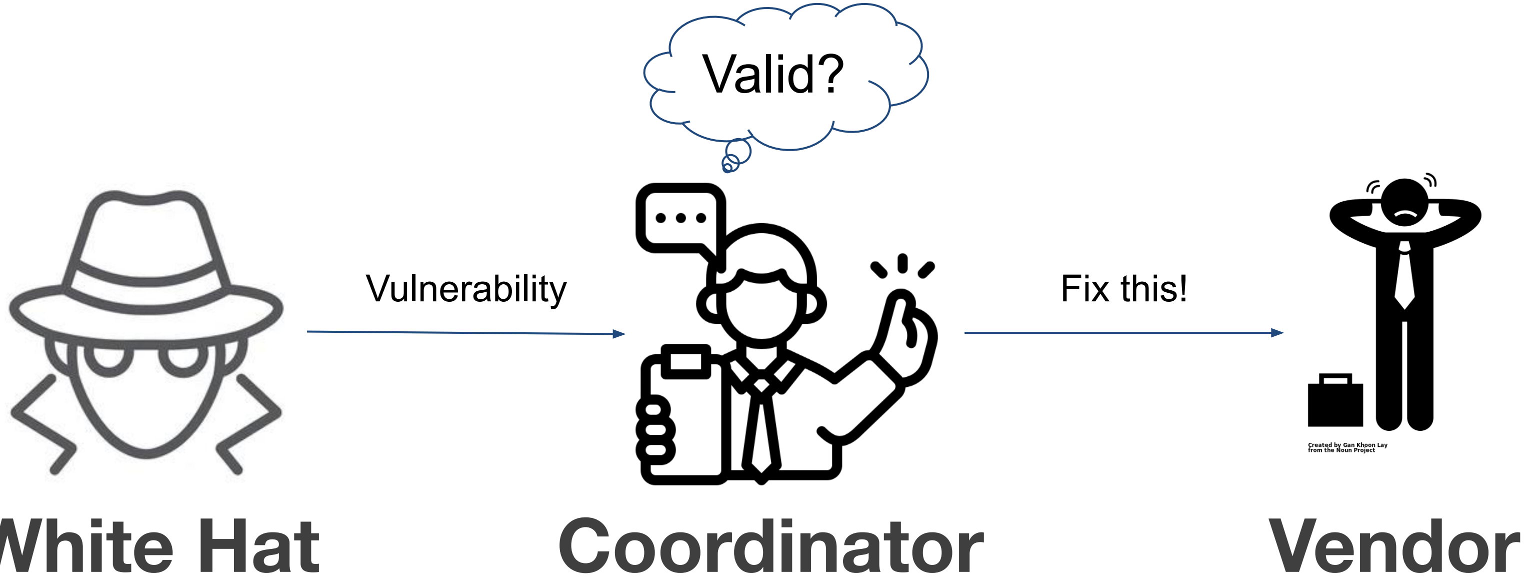
Users



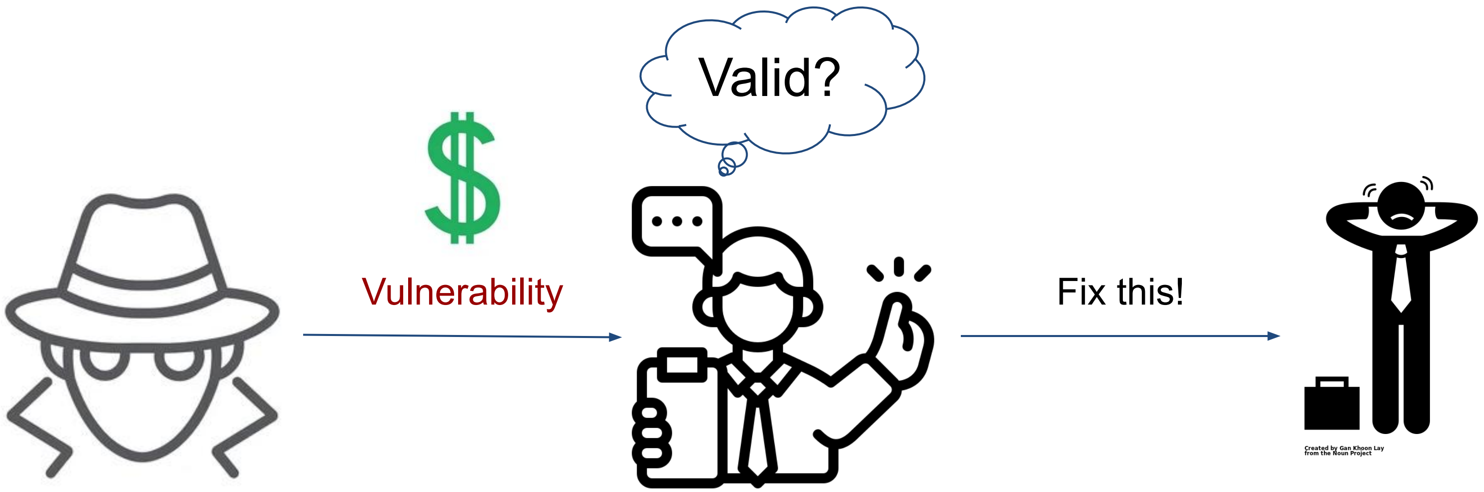
Vendors

Bad actors

Traditional Disclosure Process

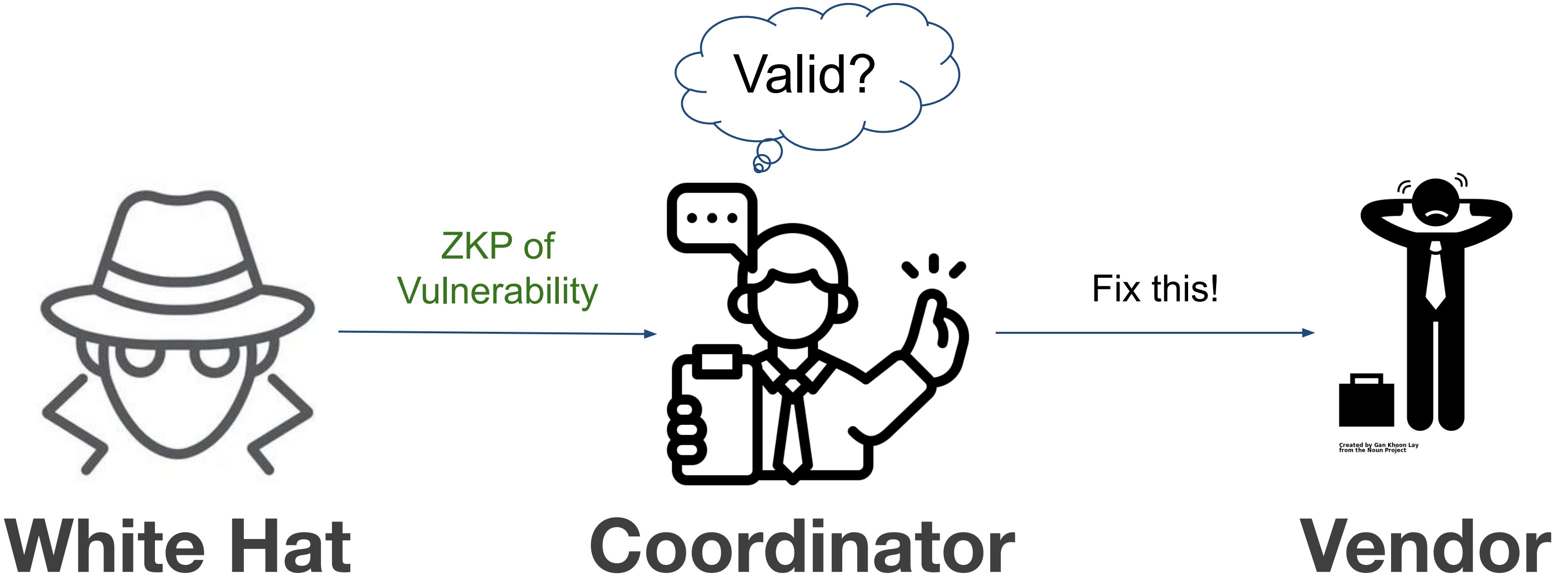


Traditional Disclosure Process

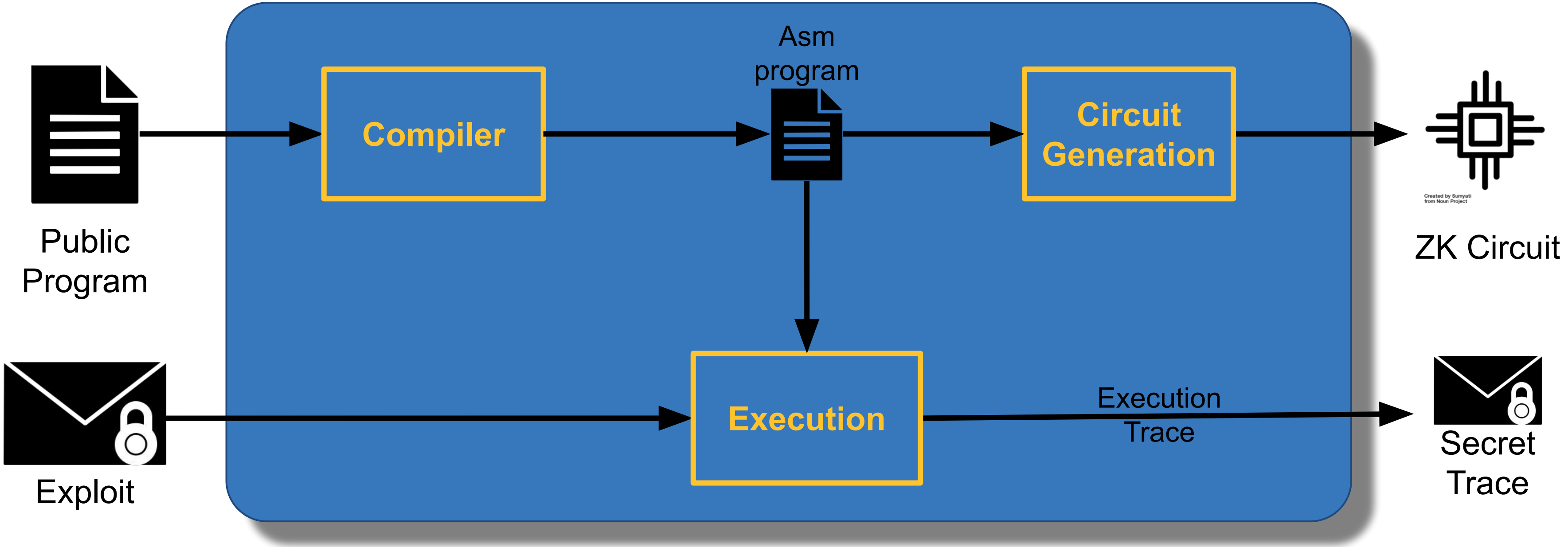


White Hat **Coordinator** **Vendor**

Improved Disclosure Process using ZKPs



Cheesecloth: Zero-Knowledge Proofs of Real World Vulnerabilities



Cheesecloth: Zero-Knowledge Proofs of Real World Vulnerabilities[1]

- Compiles the execution of arbitrary LLVM programs (C, C++, Rust) to ZK statements
- Types of vulnerabilities supported:
  - Memory errors
  - Leaks of information (Taint analysis)
  - Cryptographic protocol bugs (Model network interactions between two parties)
  - Denial of service (Symbolic execution)

1. Cuéllar et al. Cheesecloth: Zero-Knowledge Proofs of Real World Vulnerabilities. Usenix 2023

Real world case studies

Vulnerability class	Example Program	Trace length	Gates ($\mathbb{F}$ 2)	ZK Runtime†	ZK Network†
Buffer overflow	GBA Raster Image Transmogrifier (GRIT) 	5K	324M	1m 10s	416 MB
Out-of-bounds array access	FFmpeg (CVE-2013-0864)	79K	8B	29m	10 GB
Information leakage	OpenSSL Heartbleed (CVE-2014-0160)	1.3M	159B	9h 45m	203 GB
Cryptographic protocol bugs	Scuttlebutt (CVE-2020-4045)	4.5M	502B	-	-
Denial of Service (DOS)	OpenSSL Infinite Loop (CVE-2022-0778, in progress)	10T*	-	-	-

† Run on the Diet Mac'n'Cheese ZKP protocol implementation

Diet Mac'n'Cheese

ZK backend: VOLE-based *interactive* proof system

Features:

- Many fields: F_2, F_p for many p, $GF[2^n]$
- Field switching
- Permutation checks
- RAM operations
- Free disjunctions (Dora[1])
- Browser integration ("Web Mac'n'Cheese")

https://github.com/GaloisInc/swanky

1. Goel et al. Dora: Processor Expressiveness is (Nearly) Free in Zero-Knowledge for RAM Programs.

ZKPs for supply chain security

Problem: FPGA vendors typically encrypt their hardware implementations using the IEEE-1735-2014 v2 standard

- Prevents customers from inspecting FPGA implementations to verify the correctness of the FPGAs
- Customers cannot check for supply chain attacks in the FPGAs

Solution: Use formal methods (FM) to mathematically prove that FPGA implementations are correct

- Convert these FM proofs to ZKPs to prove that secret, encrypted FPGAs are correct

Proving properties about FPGAs

	ZK Runtime
Translation to SAT (Cheesecloth)	23h 43m
ZKUnsat	3m 14s

- Example FPGAs: GE Research has developed FPGA hardware that filters malformed network data
- Encode expected behavior of these FPGAs as inductive properties using state machines
- Convert the formal method proof to a ZKP
  - Convert the FPGA implementation to SAT in ZK with Cheesecloth
  - Use zkUnsat[1] to efficiently validate the SAT statement in ZK

1. Luo et al. Proving UNSAT in Zero Knowledge. CCS 2022

Proving properties about DFDL FPGAs

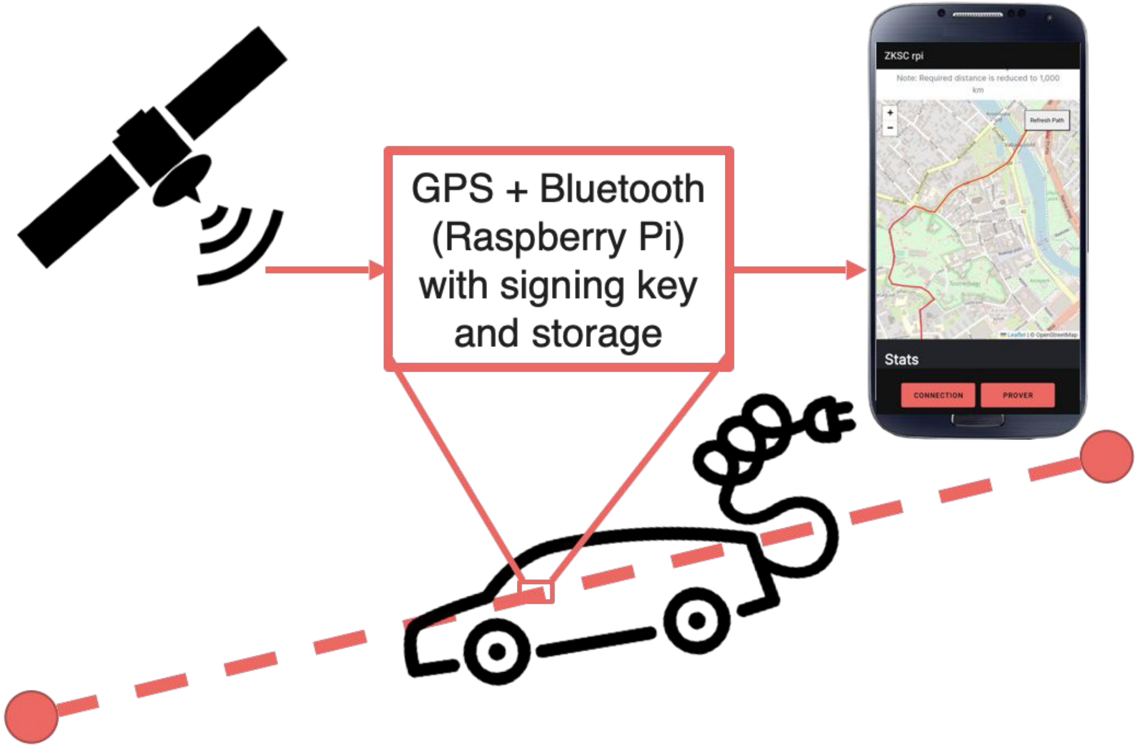
	ZK Runtime
Translation to SAT (Cheesecloth)	23h 43m
ZKUnsat	3m 14s

- Shown that it is feasible to use ZKPs to improve supply chain security
  - Ongoing work to improve the translation to SAT runtime
- ZKPs allow vendors to keep their proprietary FPGA implementations secret while customers can verify correctness properties about the implementations

ZKPs of EV tax incentive compliance

- Estonia offers a tax break on electric vehicles, as long as most of the driving time is done in Estonia
- How do you prove compliance without providing GPS driving logs to the government?
  - Our collaborator, Cybernetica, has constructed ZKP statements to accomplish this!
  - Run Diet Mac’n’Cheese for their ZK backend

ZKPs of EV tax incentive compliance

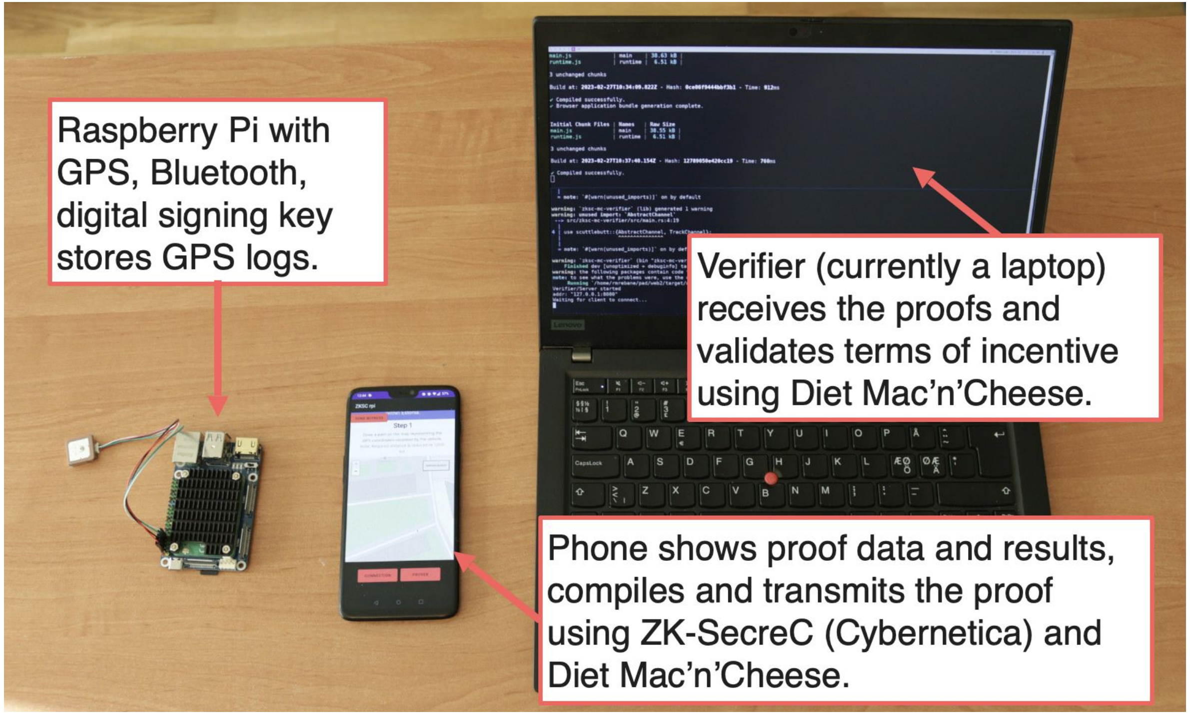


1. **Device** in vehicle stores GPS data
2. **Device** digitally signs GPS trail and sends to phone
3. **Phone** compiles a ZKP on mileage in a given country and the signature (terms of the EV subsidy!)
4. **Driver** preserves privacy of locations!

ZKPs of EV tax incentive compliance

Runtime:

- 17s with a phone and WASM
- 2s on a laptop



Other compliance applications

Many compliance applications could utilize ZKPs to improve privacy:

- Car insurance incentives: Offer discounts for tracking speed, acceleration, and location data
- Tax compliance: Instead of reporting your income to the government, provide a ZKP that your tax bill is correct
- Medical compliance: Prove that you are up to date on vaccines without sharing your entire health history

Are there policy changes that would encourage the adoption of ZKPs to improve user privacy for these applications?

Summary

Over the course of DARPA SIEVE, we have pushed the boundaries of ZKPs, demonstrating that they are practical beyond traditional blockchain use cases.

- Improving vulnerability disclosure using ZKPs
- Securing hardware supply chains by proving secret, proprietary FPGA implementations are correct
- Proving compliance with tax incentives while keeping location data secret

If you are interested in using these technologies to solve hard problems or discussing how policies might encourage their adoption, please reach out!

Resources:

- Cheesecloth: https://github.com/GaloisInc/cheesecloth/
- Diet Mac’n’Cheese: https://github.com/GaloisInc/swanky
- SIEVE IR: https://github.com/sieve-zk/ir/