| galois |

High Assurance Cryptography and Verifiable Elections: Three Galois Projects

Daniel M. Zimmerman, Prinicipal Researcher

VeTSS Workshop on Verified Software — 24 September 2019
Isaac Newton Institute for Mathematical Sciences, Cambridge, United Kingdom

Some of this material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract Nos. HR0011-19-C-0070 and HR0011-18-C-0013. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of DARPA.

About Galois

- established in 1999 to apply functional programming and formal methods to the problem of information assurance
- 3 offices across the U.S. (Portland, OR / Arlington, VA / Dayton, OH)
- ~115 people working on ~60 active projects for a wide variety of clients, including U.S. government (DARPA, IARPA, AFRL, others) and commercial (Amazon, Microsoft, others)

About Galois

- over the years, we have substantially broadened our scope to *high assurance everything*
- until recently, *everything* basically meant *software*, but we've started doing hardware work as well — about which more later
- in practice, R&D in programming languages, code and binary analysis, verification, security, cryptography (incl. secure computation), novel hardware design techniques, and more
- our toolbox: symbolic execution, model checking, interactive theorem provers, SMT solvers, functional programming… multiple core tools developed in-house
- we are organized rather uniquely—https://lifeatgalois.com/ (it's all true!)

Galois R&D Model

- big research projects from U.S. government agencies
- smaller, commercial research projects from Amazon, Facebook, Microsoft, others
- many collaborations with academic partners
- spinout companies commercializing research technology (Tangram Flex, MuseDev, Free & Fair)

DARPA

amazon web services

Microsoft

TANGRAMFLEX

FREE & FAIR

Galois Tools

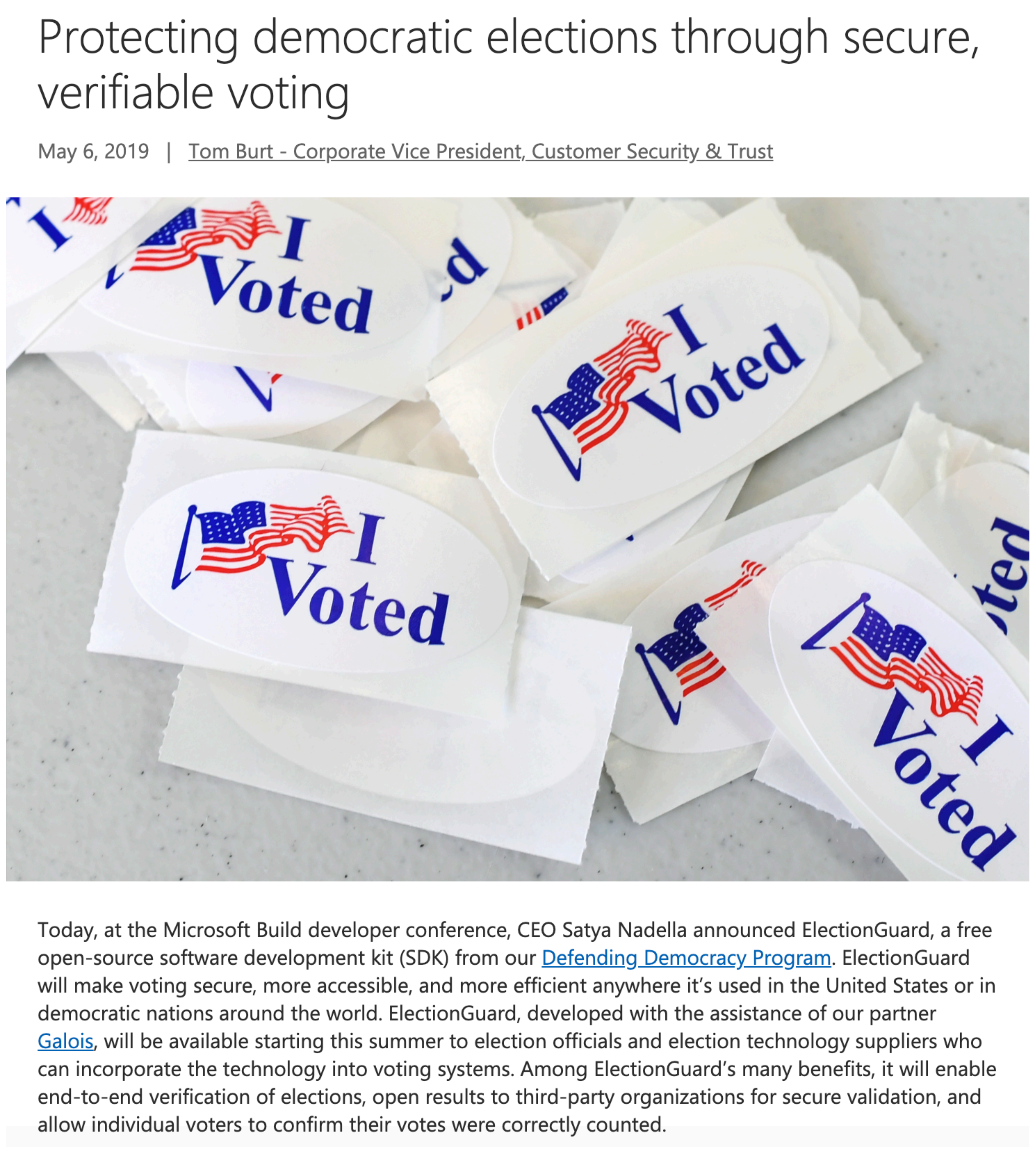
- we generally provide open source releases of as much of our core tooling as our clients allow:
  - *Cryptol* - domain-specific language and verification system for cryptography (cryptol.net)
  - the *Software Analysis Workbench (SAW)* - reasoning tool for models of software behavior (saw.galois.com)
  - *Crucible* - library for forward symbolic execution of imperative programs (github.com/GaloisInc/crucible)
  - *Macaw* - binary analysis framework (github.com/GaloisInc/macaw)
  - and over 200 other public repositories on GitHub (over 300, including forks of others' tools/libraries)

Remainder of This Talk

- three Galois projects touching upon high assurance cryptography and verifiable elections
    - software: Microsoft ElectionGuard
    - hardware: 21st Century Cryptography
    - both: BESSPIN

Project: Microsoft ElectionGuard

- commercial project designing and implementing a cryptographic SDK for *end-to-end verifiable (E2E-V) elections*
- based on cryptographic protocols developed by Microsoft Research
- goal: enable existing vendors of election technology to (dramatically!) improve their systems by incorporating the SDK



Protecting democratic elections through secure, verifiable voting

May 6, 2019 | Tom Burt - Corporate Vice President, Customer Security & Trust

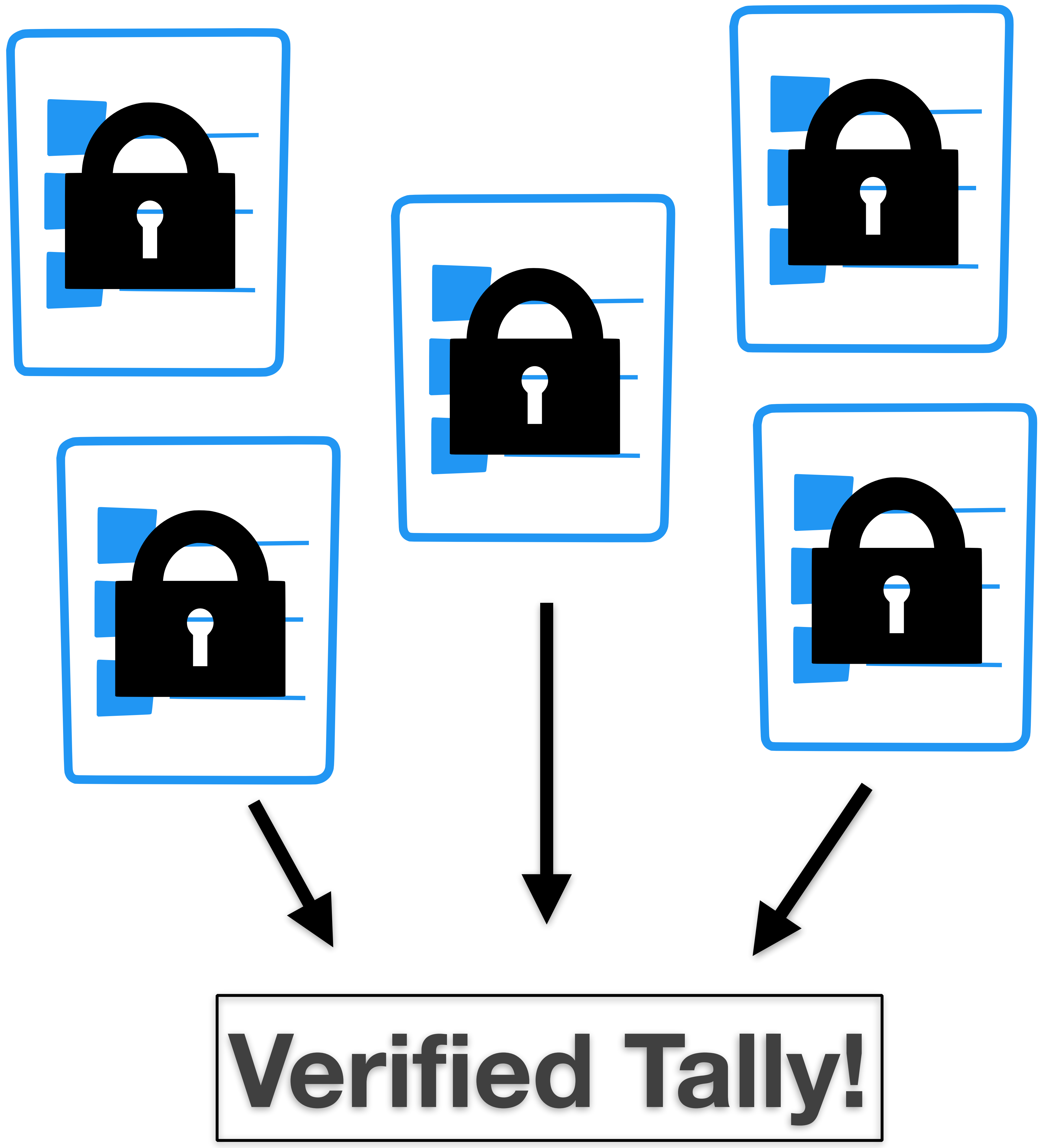
Today, at the Microsoft Build developer conference, CEO Satya Nadella announced ElectionGuard, a free open-source software development kit (SDK) from our Defending Democracy Program. ElectionGuard will make voting secure, more accessible, and more efficient anywhere it's used in the United States or in democratic nations around the world. ElectionGuard, developed with the assistance of our partner Galois, will be available starting this summer to election officials and election technology suppliers who can incorporate the technology into voting systems. Among ElectionGuard's many benefits, it will enable end-to-end verification of elections, open results to third-party organizations for secure validation, and allow individual voters to confirm their votes were correctly counted.

End-to-End Verifiable Elections

- any voter can obtain proof that their ballot was included as intended in the final tally
- any observer can check that the tally was carried out correctly
- it is impossible for a voter to prove to someone else how they voted (to prevent coercion and vote buying/selling)

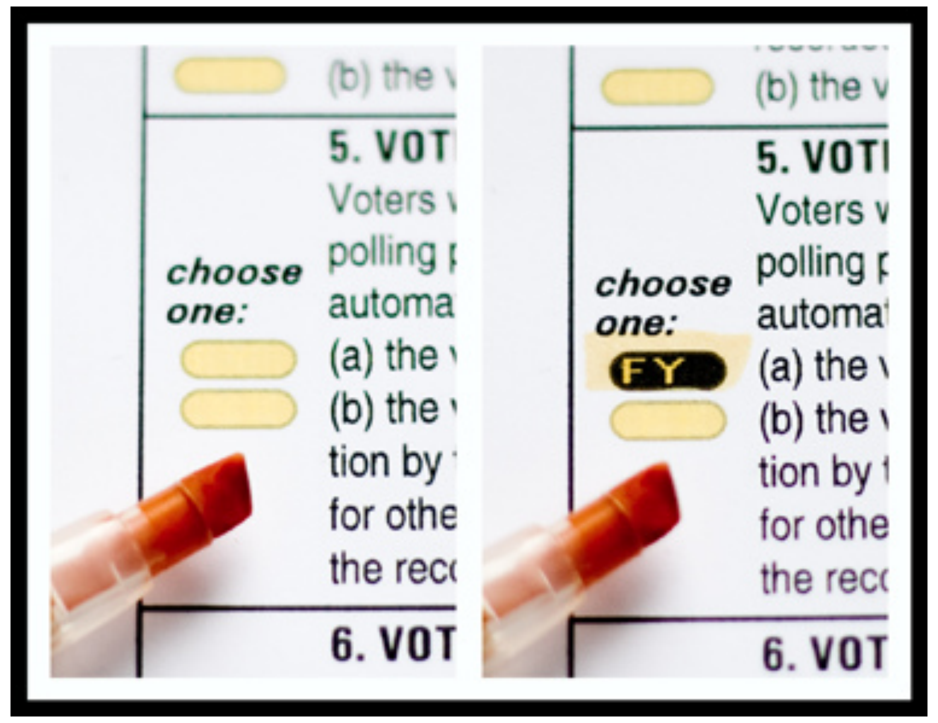
End-to-End Verifiable Elections

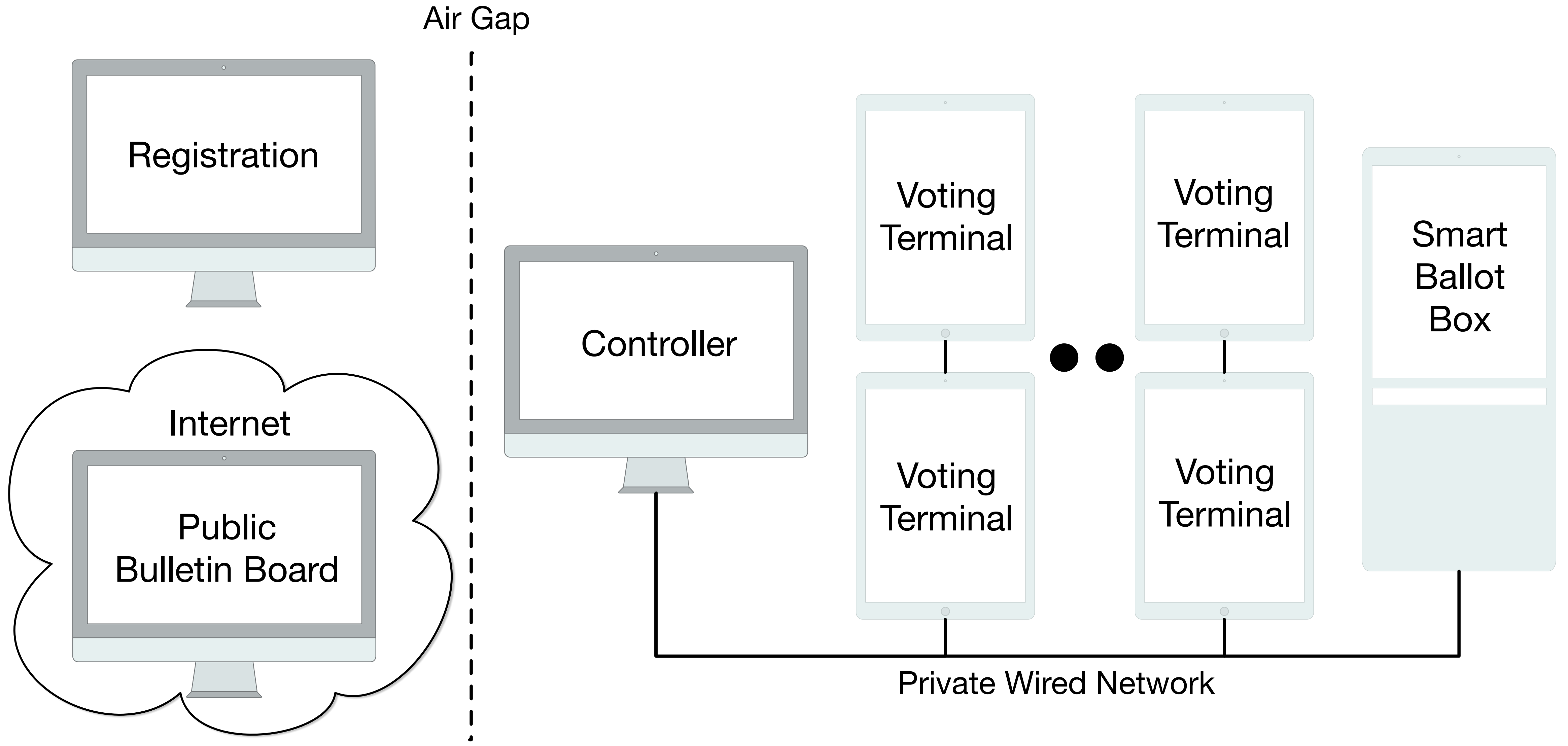
- voters prepare and cast encrypted ballots, which can be made public—even linked to voters' identities—once they are encrypted
- the encrypted ballots are processed to generate a *tally* —the election result—and a *proof* that the tally matches the set of cast ballots



E2E-V Systems

- several E2E-V systems have been proposed or developed:
  - Scantegrity/Scantegrity II
  - Helios (Internet-based)
  - Prêt À Voter
  - STAR-Vote
  - others…
- all rely on sophisticated cryptography: secret sharing schemes, homomorphic tallying, mix-nets…

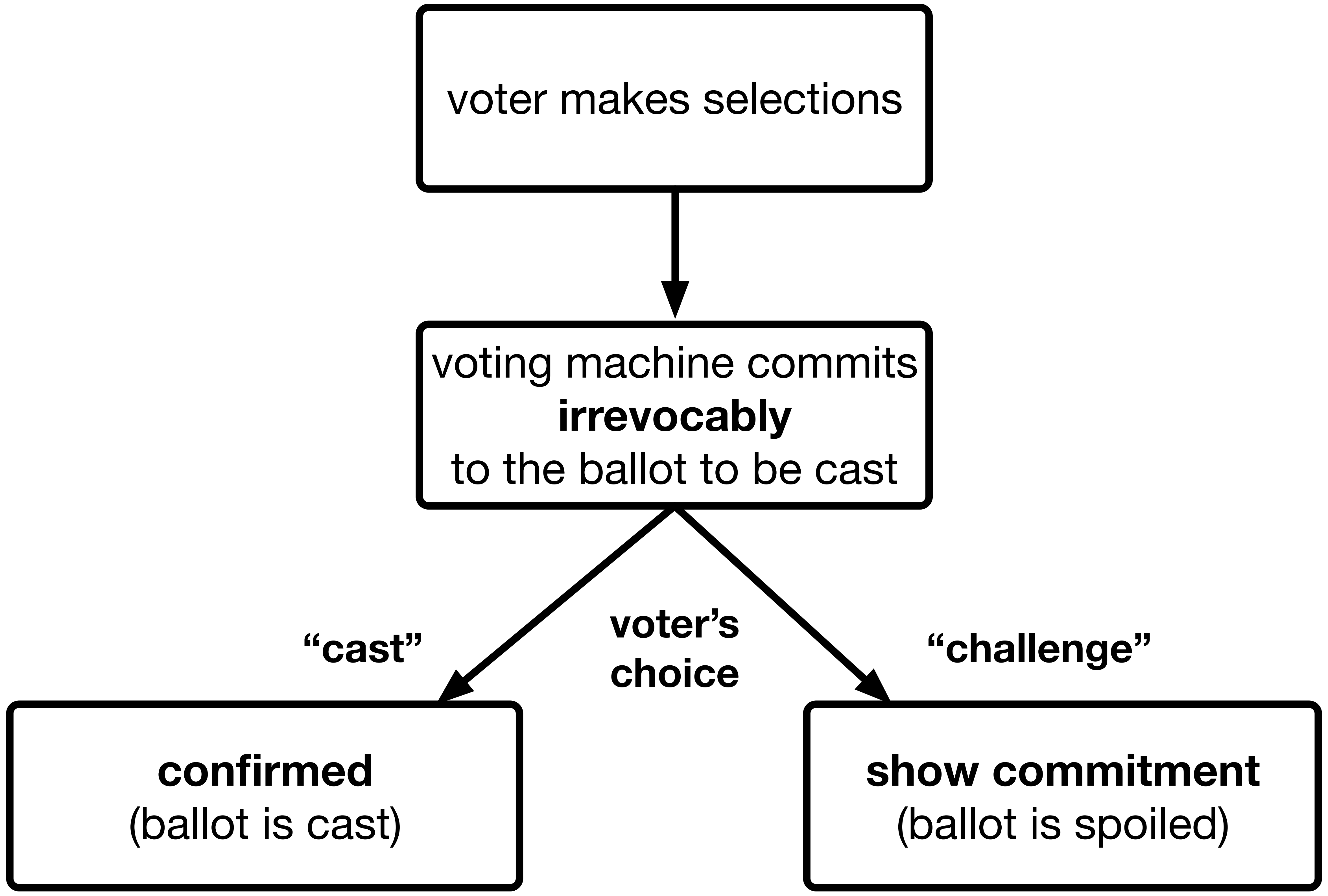




ElectionGuard Cryptography

- ElectionGuard is based on the concept of a *Benaloh challenge*
- threshold cryptography (ballot decryption requires the cooperation of multiple *trustees*)
- homomorphic tallying

Benaloh Challenges (2006)

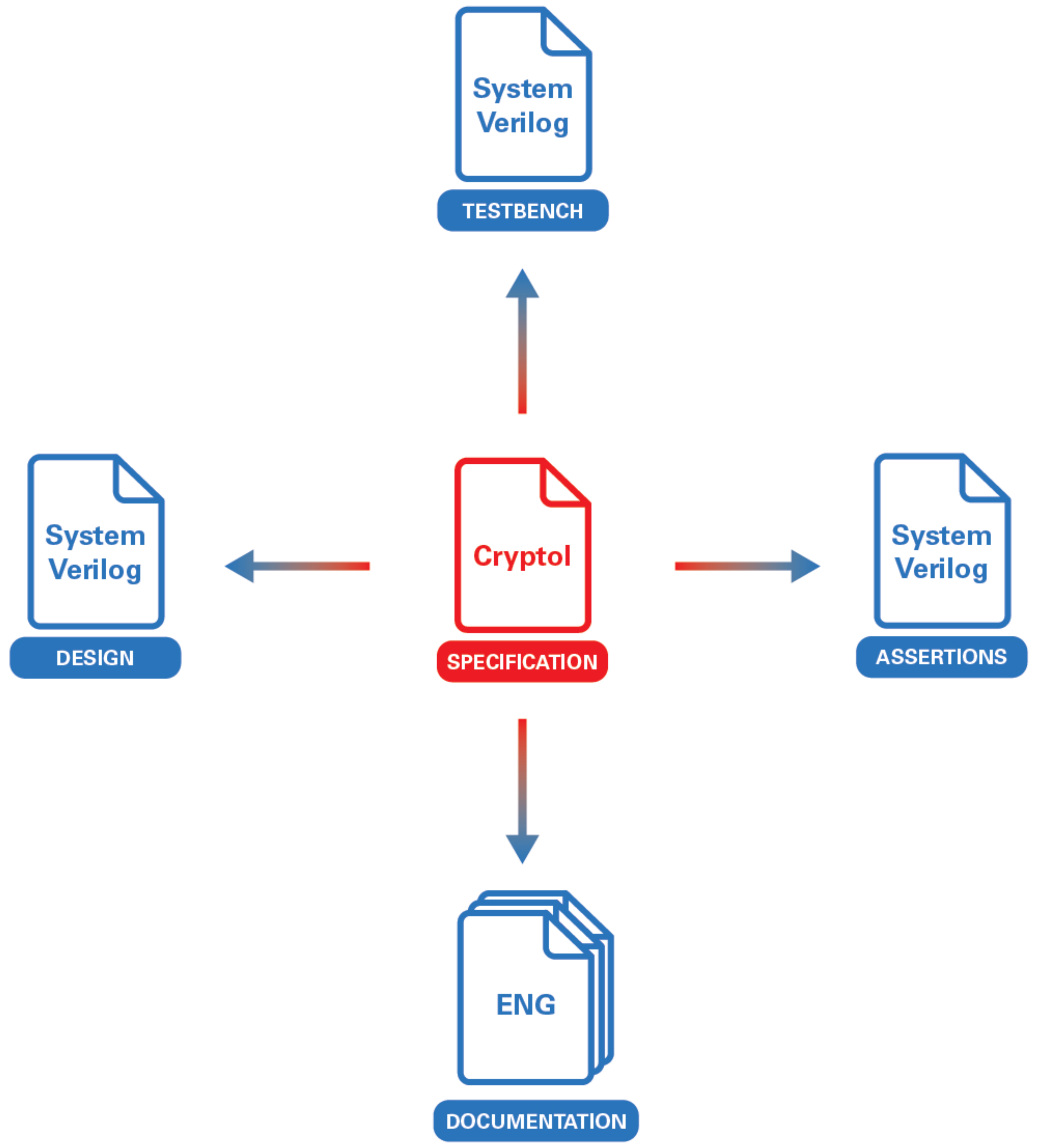

voter makes selections
voting machine commits **irrevocably** to the ballot to be cast
"cast"
voter's choice
"challenge"
confirmed (ballot is cast)
show commitment (ballot is spoiled)

ElectionGuard Assurance/Verification

- initial spec of protocol in Coq
- Cryptol implementations of protocol and cryptographic primitives
- C implementation of SDK (that's what election vendors use!)
- **status:** Cryptol and C implementations complete, several properties proven about Cryptol implementation
- **aspirational goal:** prove equivalence between Cryptol and C implementations using SAW
- open source release coming soon: https://github.com/microsoft/ElectionGuard-SDK

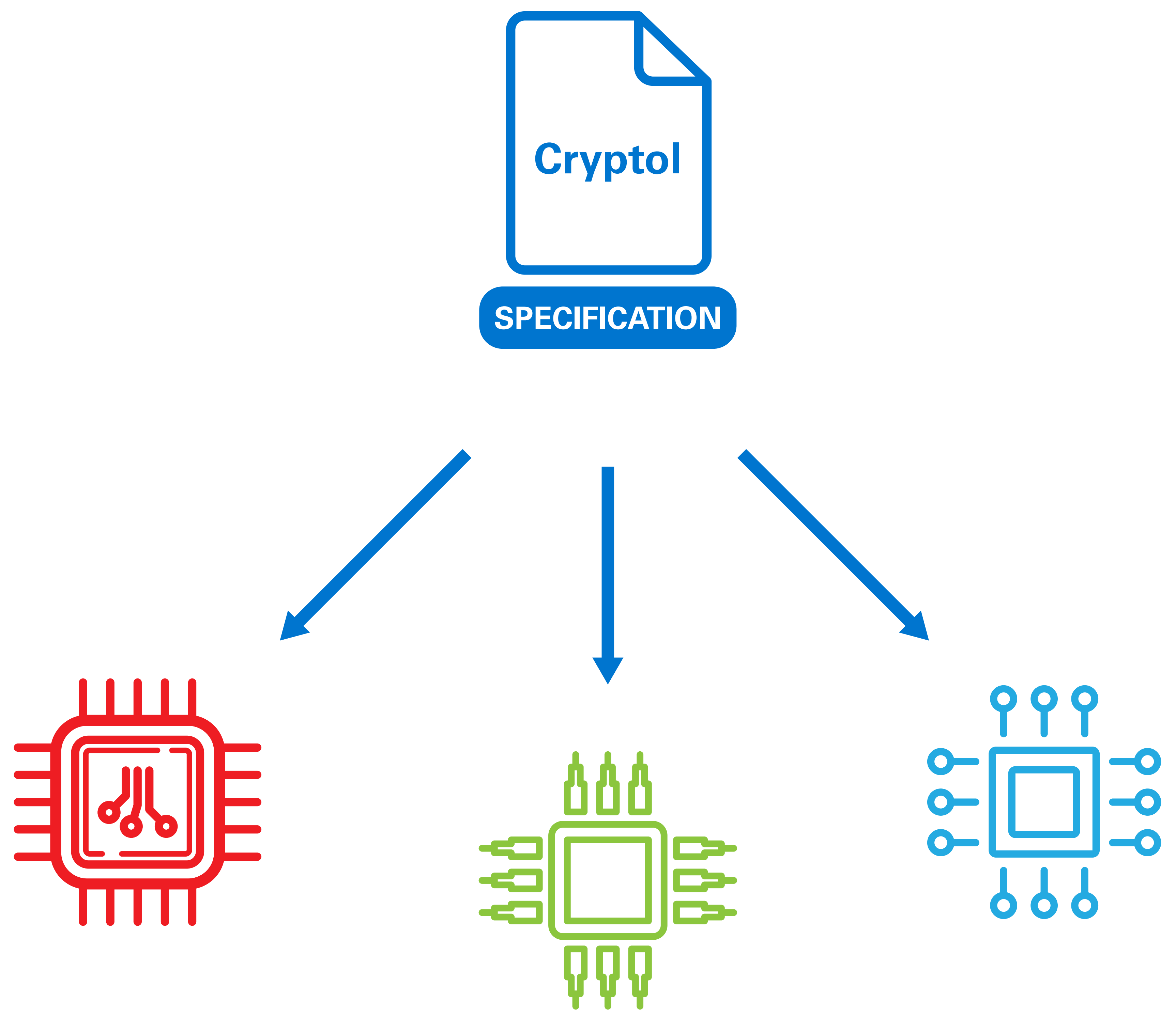
Project: 21st Century Cryptography

- 16-month DARPA project to design, fabricate, and evaluate an asynchronous cryptographic ASIC with novel side channel mitigation techniques
- *synthesis* of cryptographic circuits and supporting artifacts directly from verified Cryptol implementations



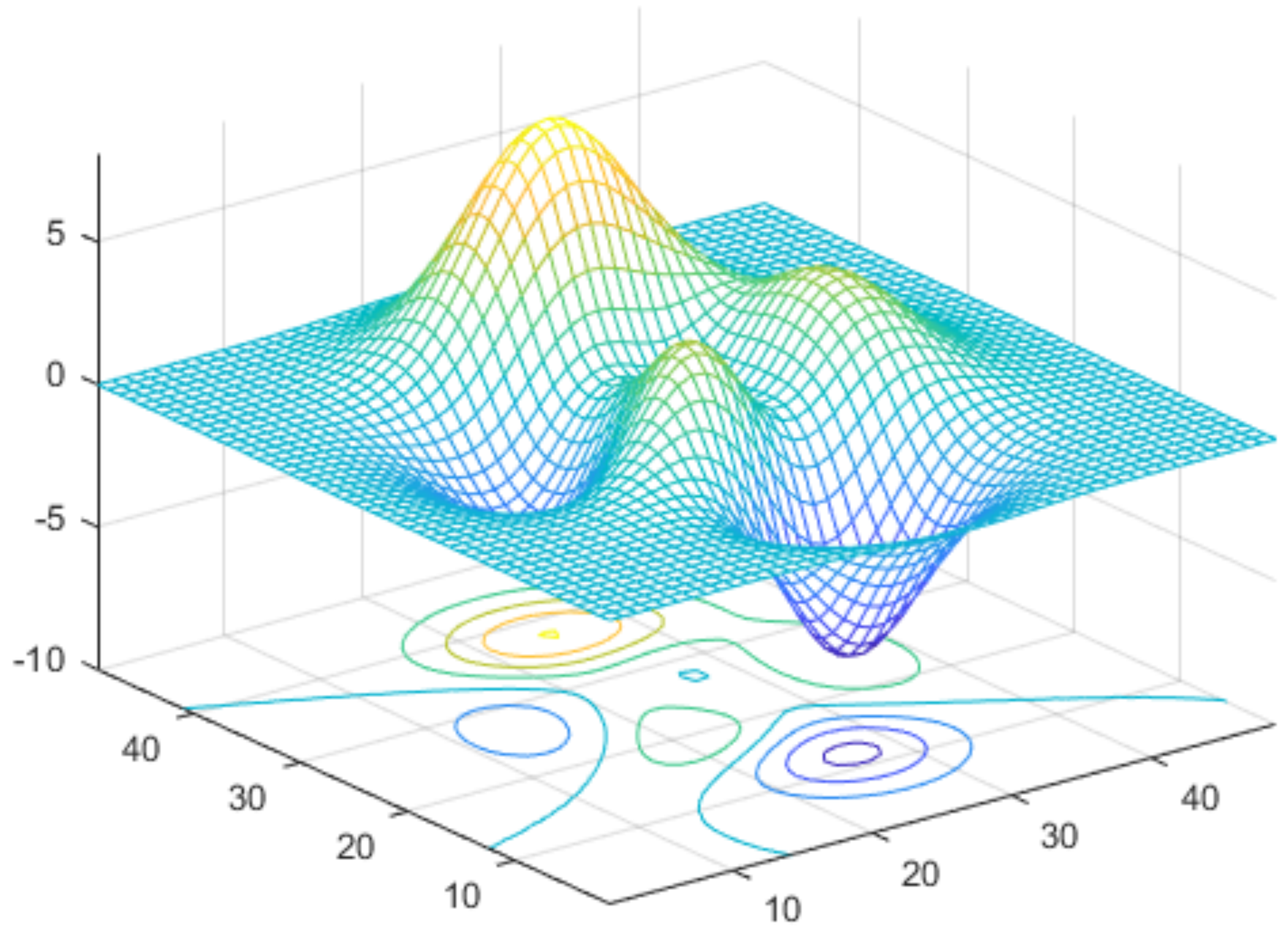
Product Line Synthesis

- synthesize not just a single instance of each circuit, but an entire *product line*
  - variation based on cryptographic parameters (e.g., key sizes)
  - variation based on performance, pipelining, synchronous vs. asynchronous implementation…



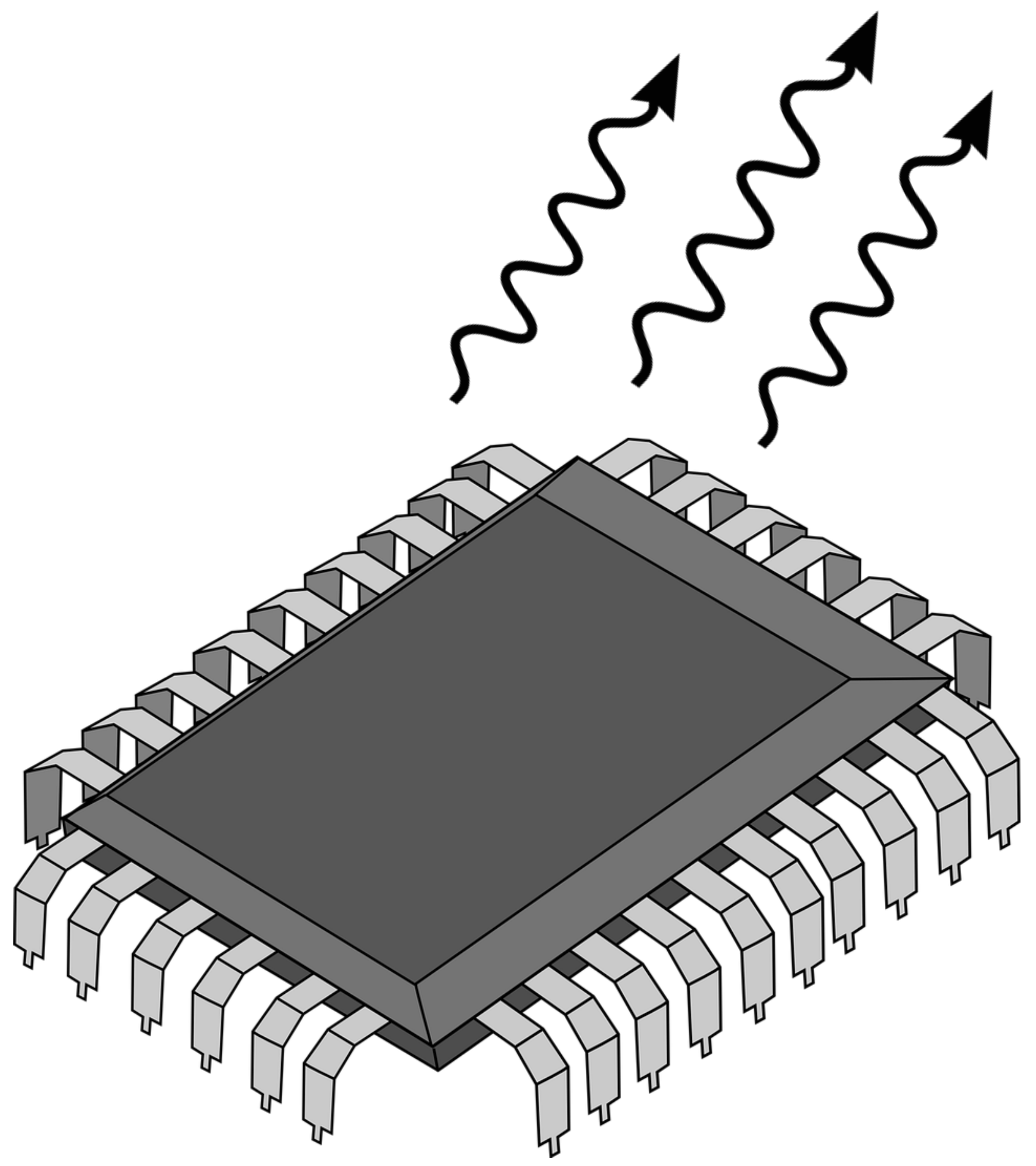
Product Line Synthesis

- synthesizing a product line enables *tradeoffs* among **p**ower, **p**erformance, **a**rea, and **s**ecurity (PPAS)
- can pick the "right" (or "best") circuit for the job at hand



Side Channels

- synthesis from verified Cryptol lets us rule out implementation errors for the “S” in PPAS… but what about side channels?



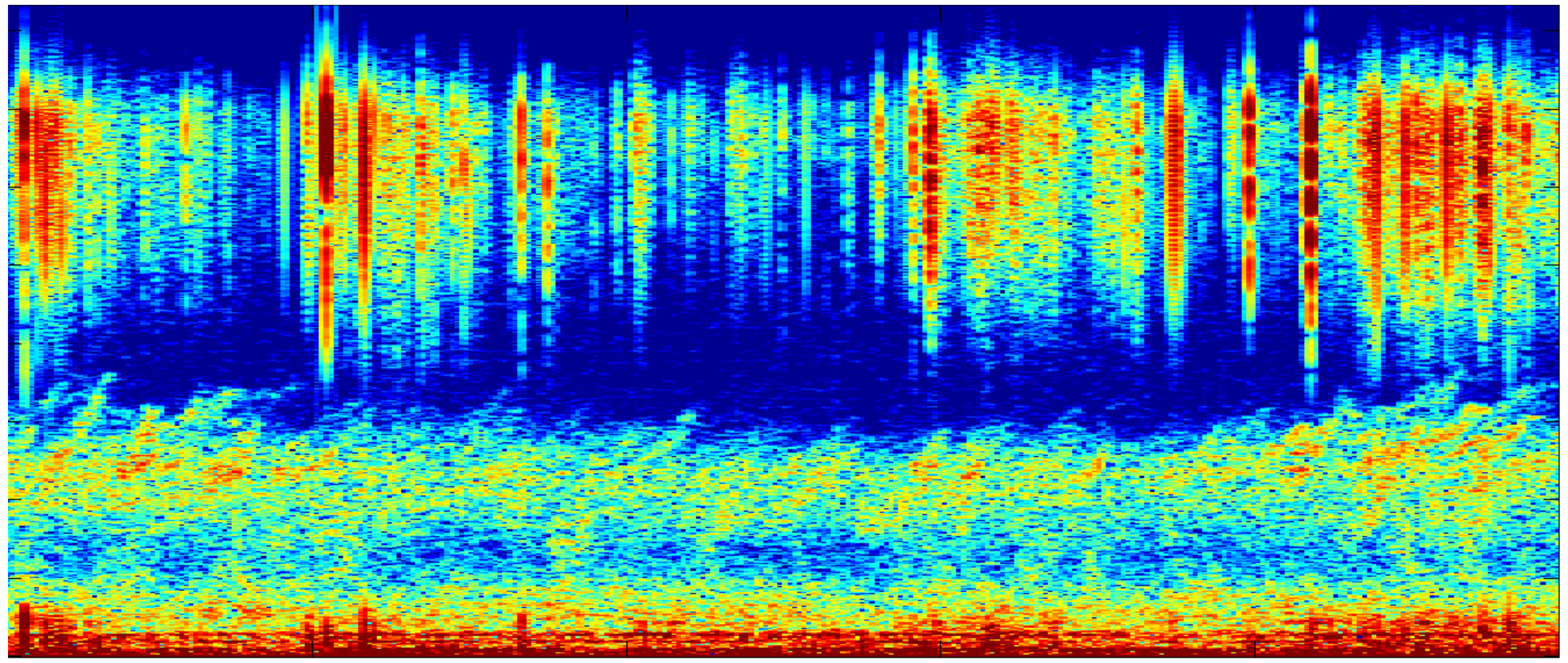
Side Channels: Timing

- timing side channels leak information by taking different amounts of time to process a 1 or 0 bit

```
if (key_bit == 0)
  { do_something_simple }
else
  { do_something_complex }
```

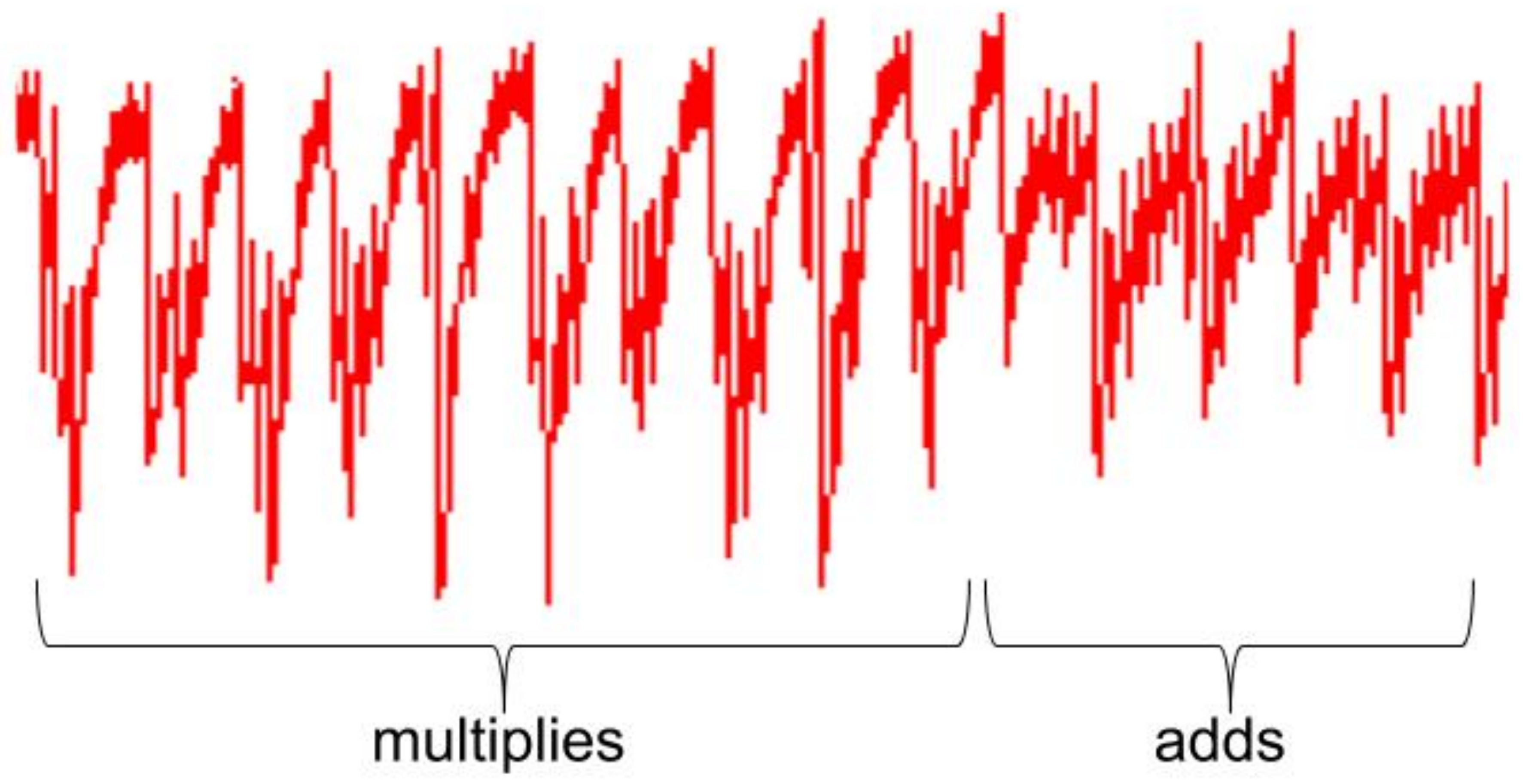
Side Channels: Electromagnetic

- when power draw changes, current flow changes, and changing current generates EM fields
- constant current at the boundary of the chip can hide changing currents within the chip that still create distinct EM signatures



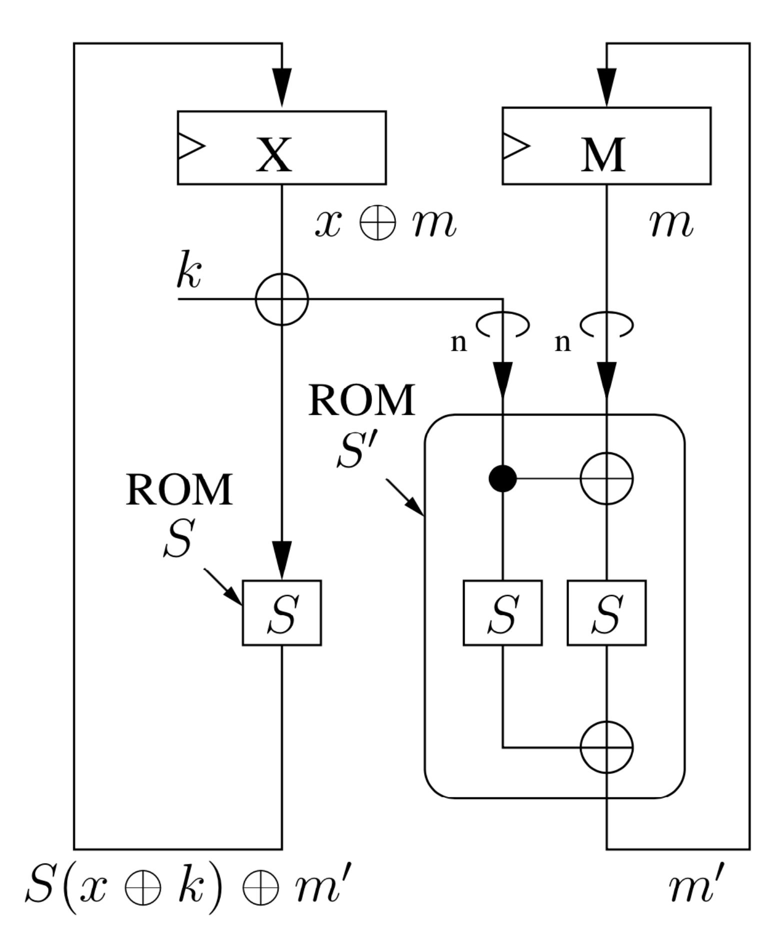
Side Channels: Power

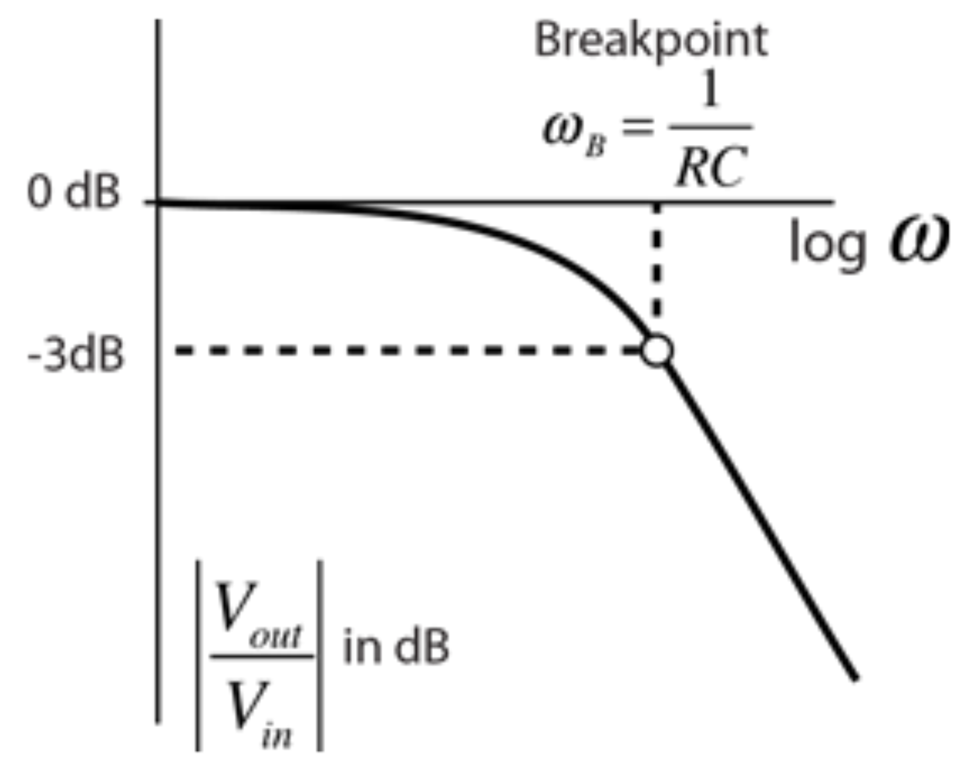
- power side channels leak information
- processing a 1 or a 0 can consume different amounts of power
- different operations can consume different amounts of power

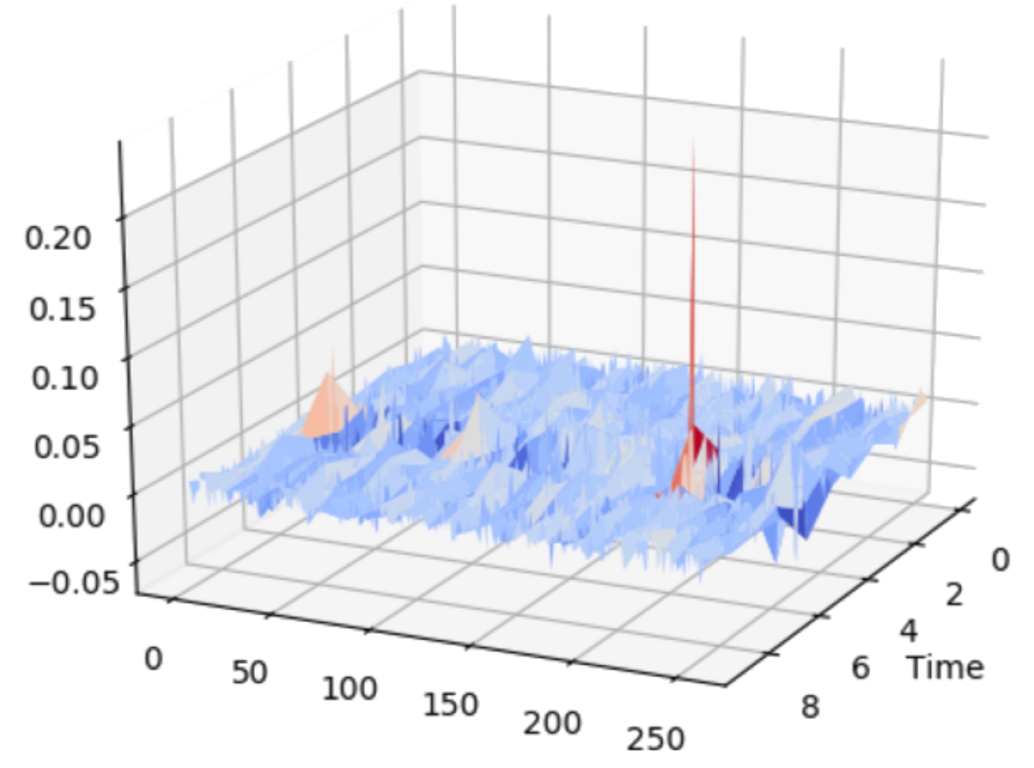


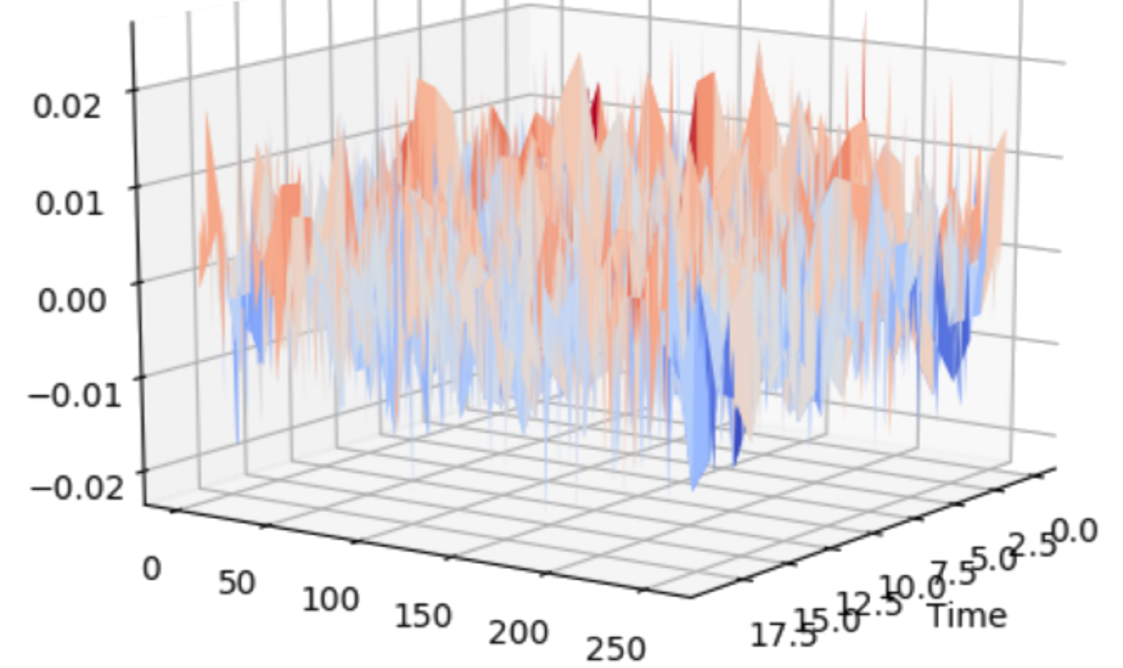
Power Side Channel Resistance

- 21CC is primarily concerned with power side channels
- lots of ongoing research on mitigations
- some well-researched existing techniques:
  - logic masking
  - current smoothing
  - noise generation



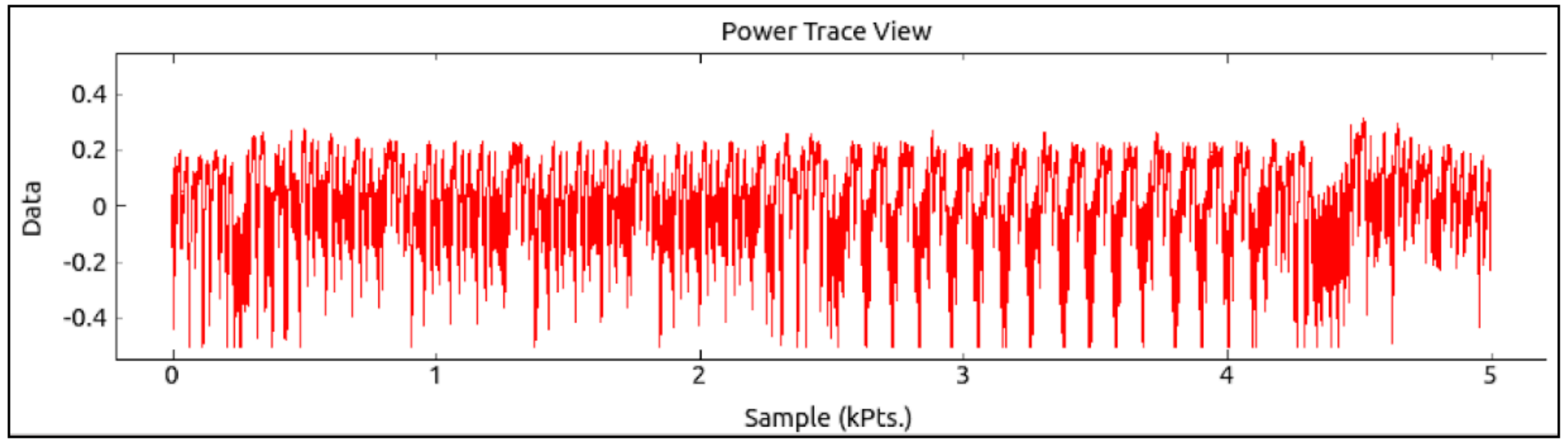






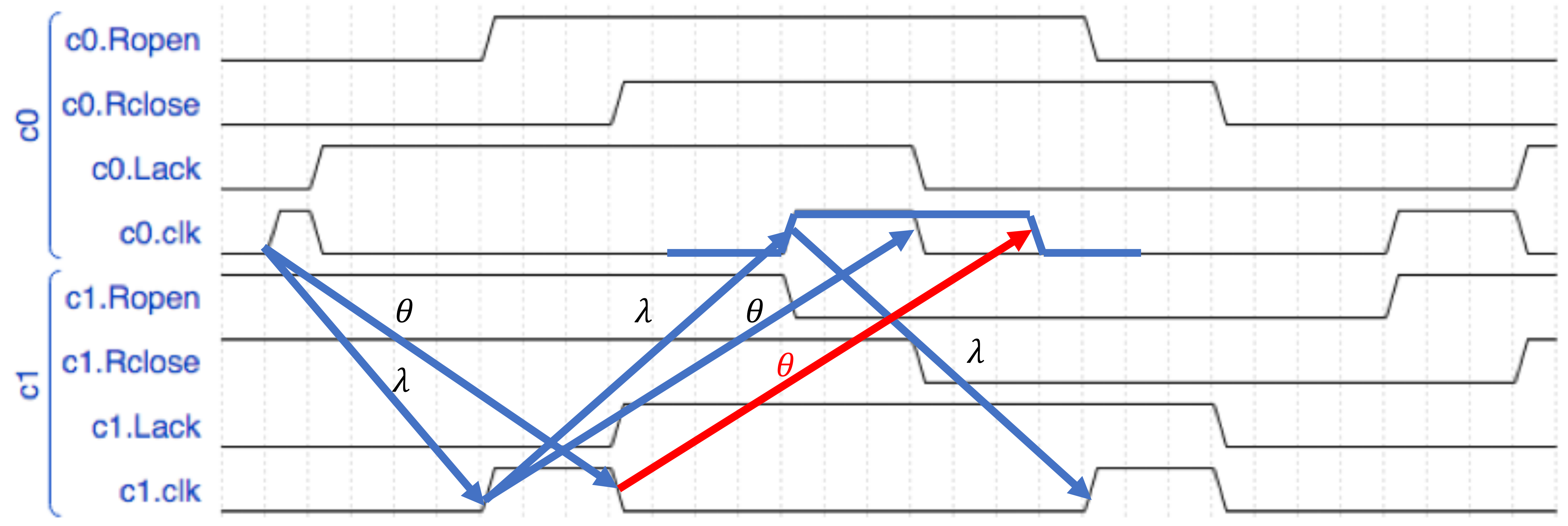
Mitigation Technique: Randomization

- *randomization* has appeared in some literature, but not extensively
- the idea: confuse alignment of power/EM traces and thereby hide/de-correlate signal
- expected to have lower cost than some other techniques $\Rightarrow$ *good PPAS tradeoff*



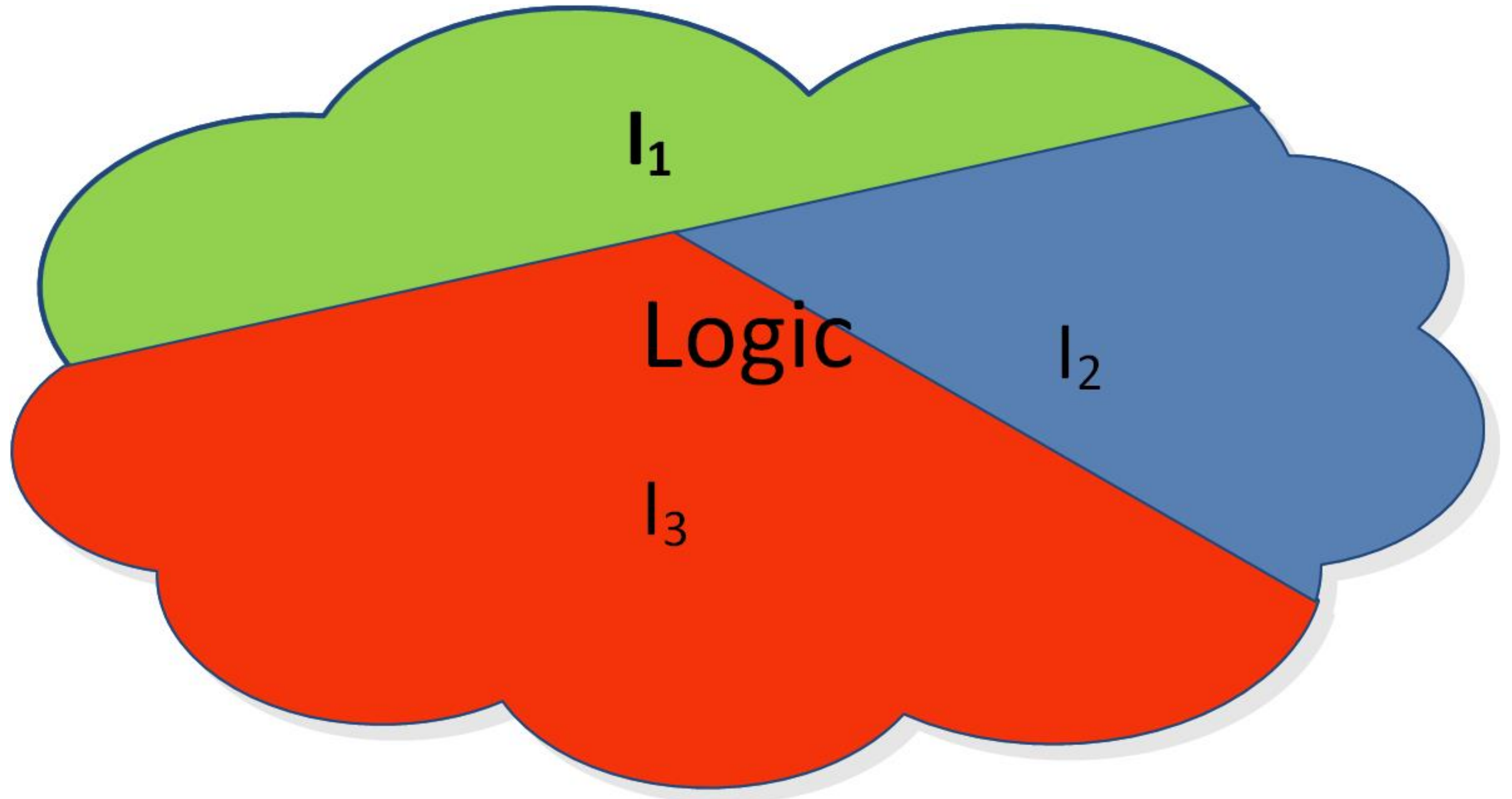
Timing Resilient Design

- our circuit implementation approach is *asynchronous*
- allows for random events (like voltage changes) to upset *timing* while maintaining *correct operation*



Multiple Independent Voltage "Islands"

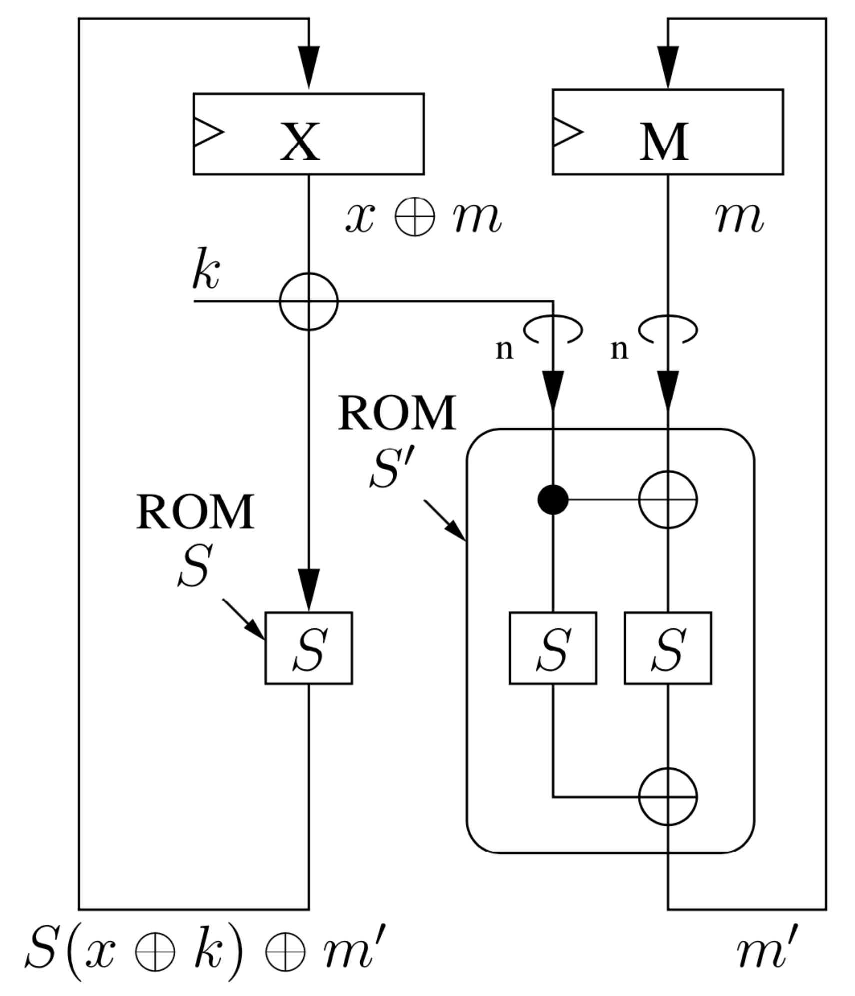
- timing resilience enables multiple independent "islands" of varying voltages across a chip, not just one supply voltage
- each island runs at a different (changing) speed, but they still work together properly
- essentially performs both randomization and noise generation



🟩 Voltage Island 1
🟦 Voltage Island 2
🟥 Voltage Island 3

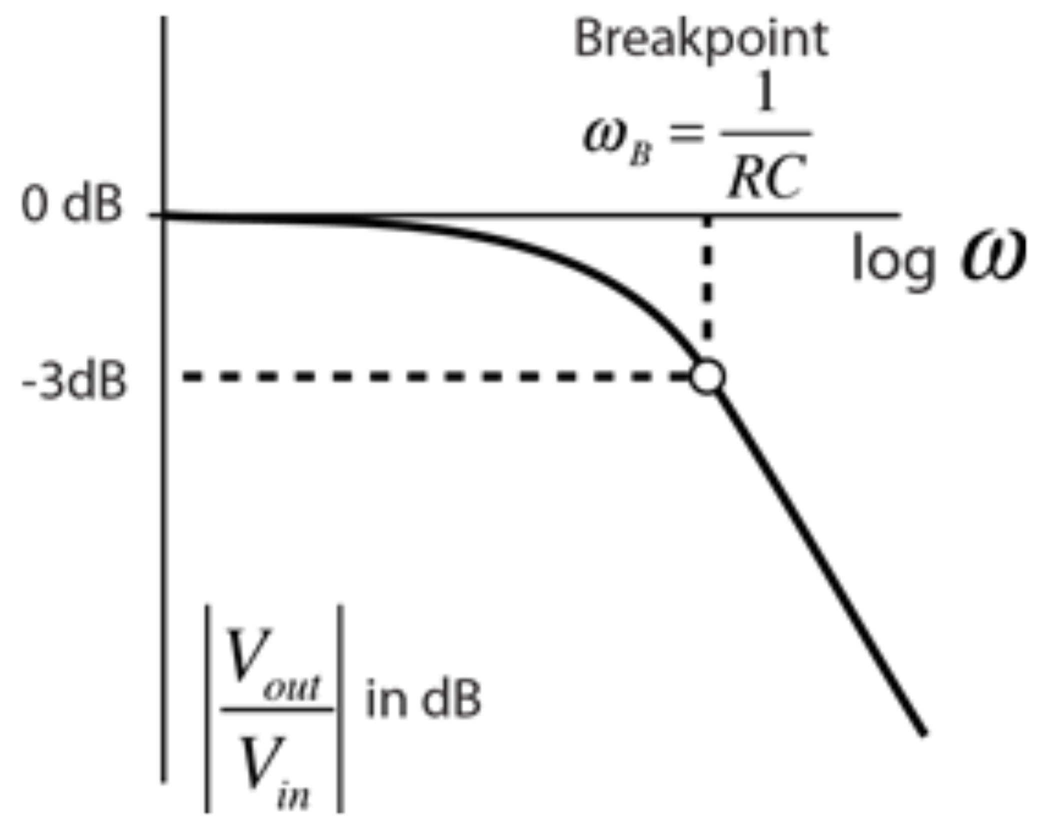
Randomization & Others

- randomization is orthogonal to many logic masking techniques and current smoothing techniques
- multiple mitigation techniques can be combined to achieve different PPAS tradeoffs



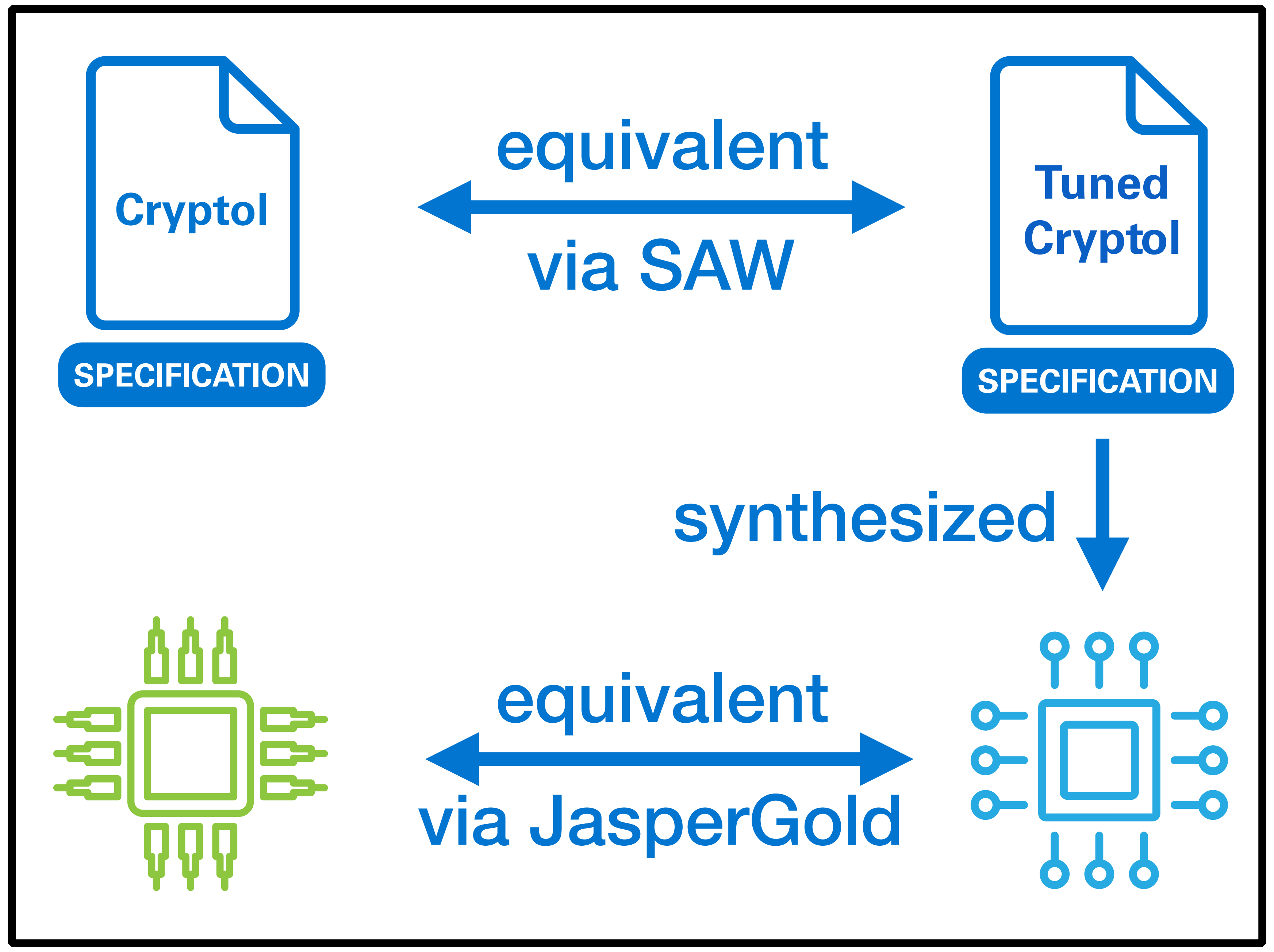
+

+



21CC Assurance/Verification

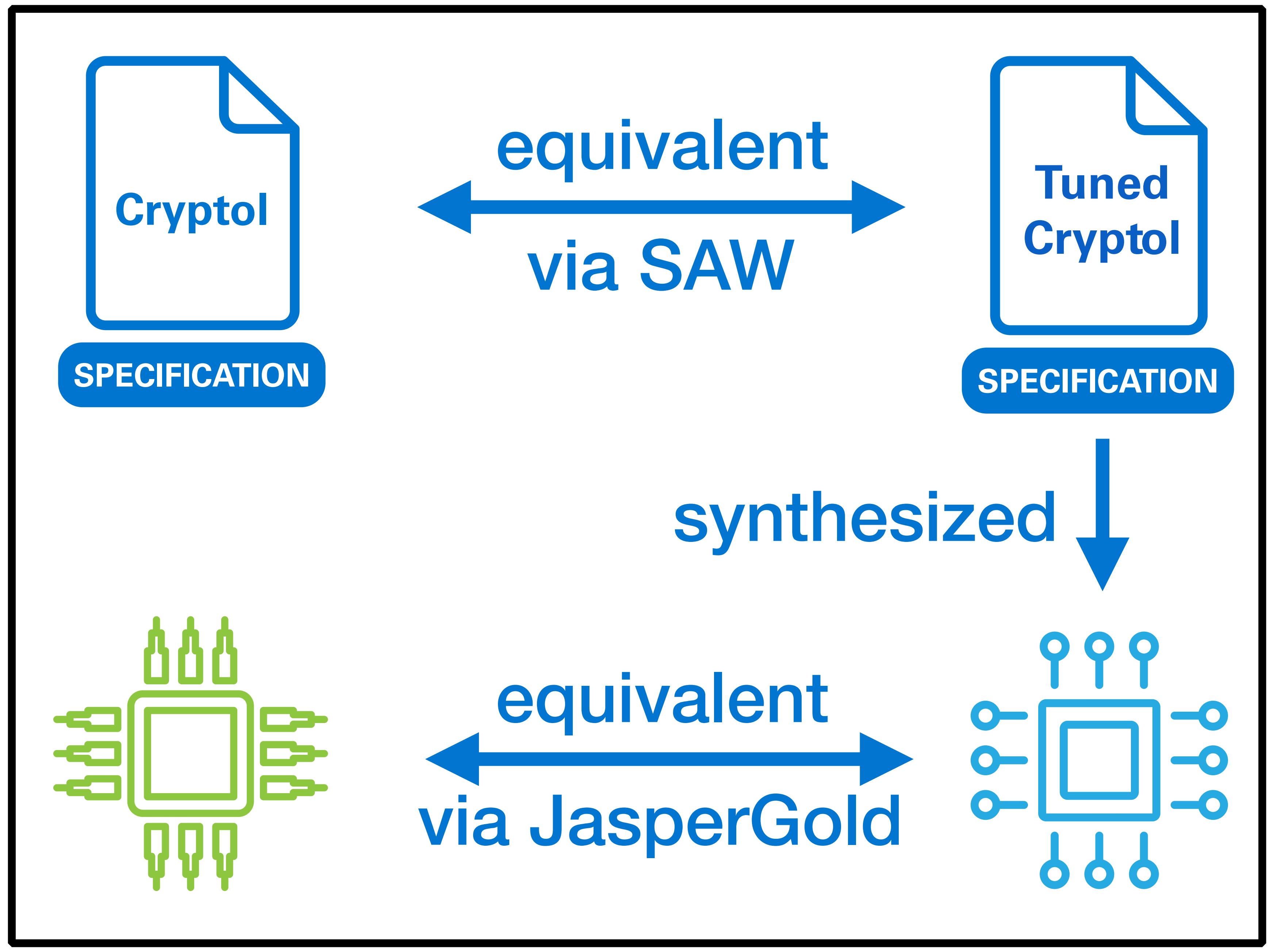
- initially using SAW to prove equivalence of “reference” Cryptol implementations (based on NIST standards) and Cryptol tuned for hardware synthesis
- initially using industrial hardware verification tools to prove equivalence of our synthesized hardware descriptions and “reference” circuits

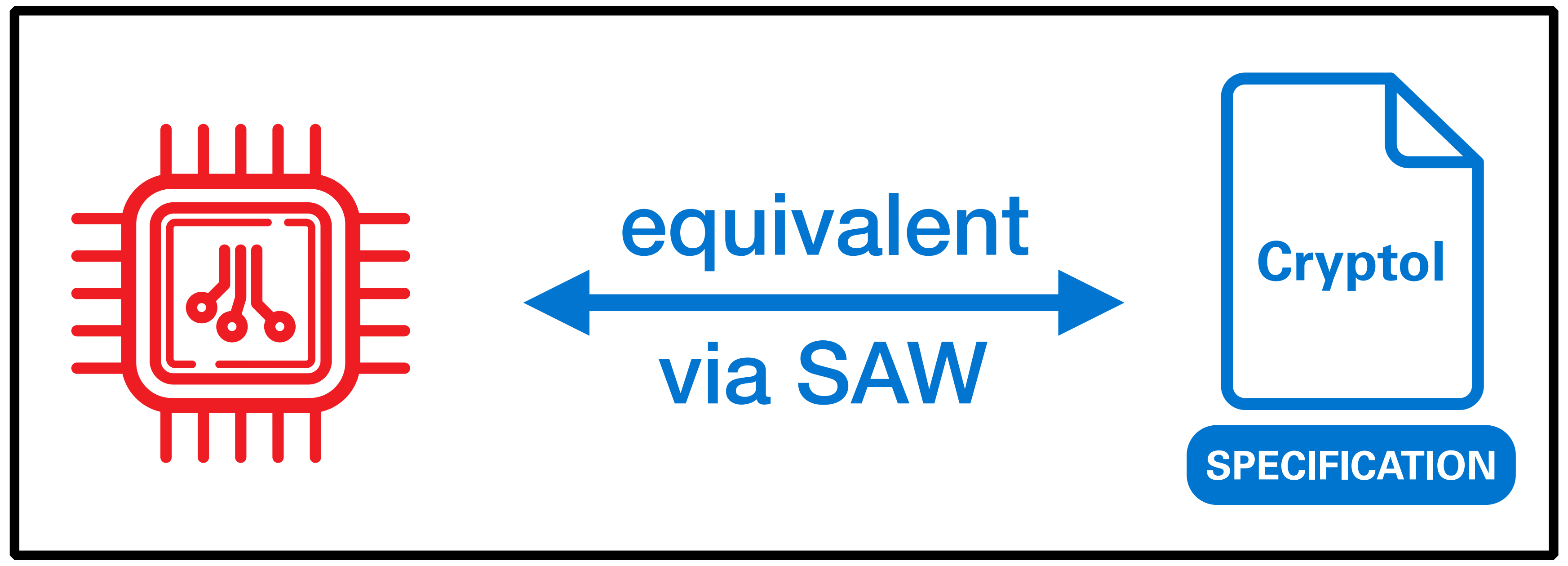


21CC Assurance/Verification

- initially using SAW to prove equivalence of “reference” Cryptol implementations (based on NIST standards) and Cryptol tuned for hardware synthesis
- initially using industrial hardware verification tools to prove equivalence of our synthesized hardware descriptions and “reference” circuits

- **aspirational goal**: use SAW to *directly* prove equivalence of hardware descriptions to implementations in any language understood by SAW





Project: BESSPIN

- **B**alancing **E**valuation of **S**ystem **S**ecurity **P**roperties with **I**ndustrial **N**eeds
- a project for DARPA MTO SSITH (**S**ystem **S**ecurity **I**ntegrated **t**hrough **H**ardware and Firmware)

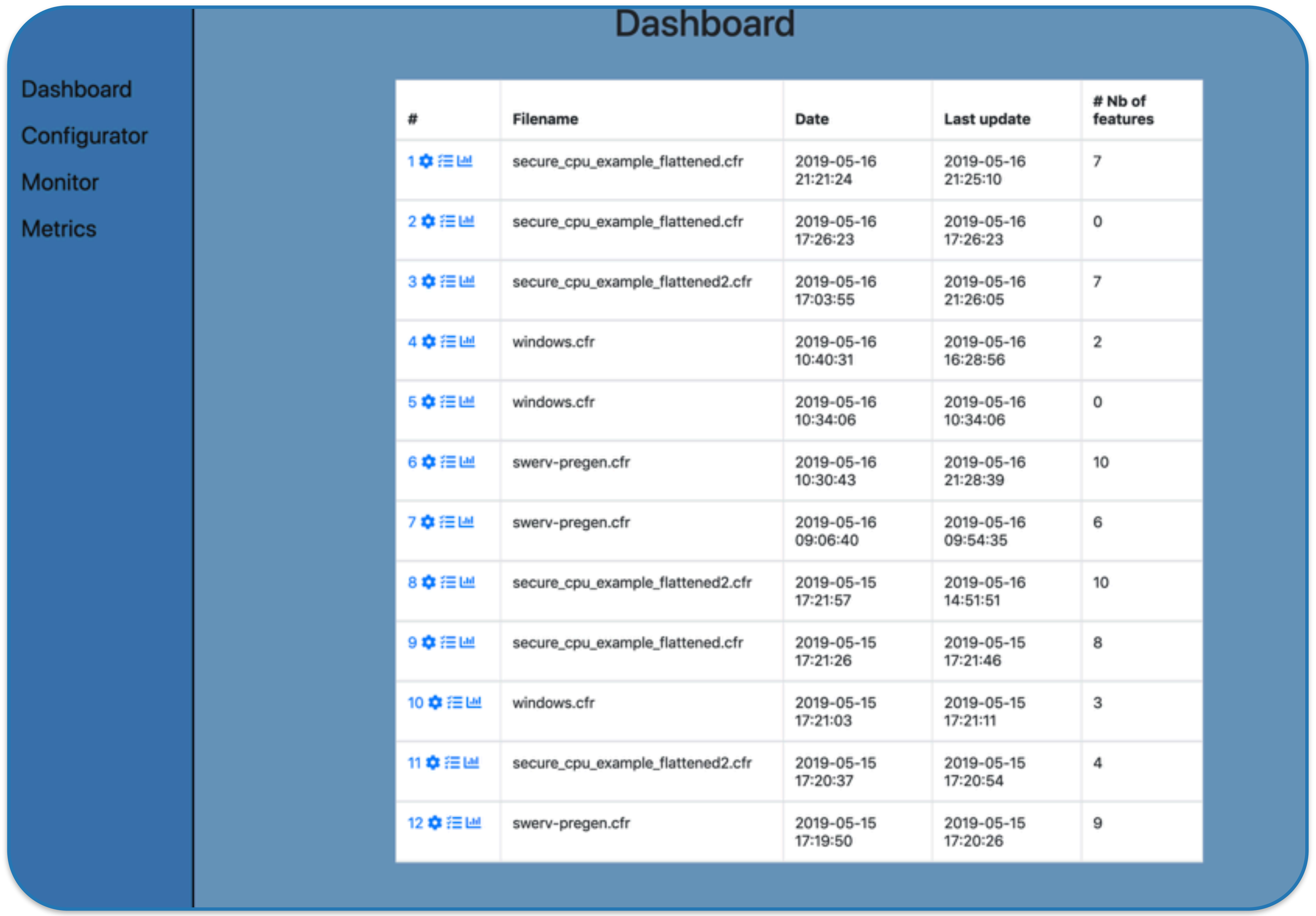
"Cloud City on Bespin", © Adrian Mark Gillespie
Used with Permission

SSITH Overview

- ~4 year program to develop hardware security architectures and associated design tools to protect systems against 7 classes of hardware vulnerabilities exploited through software
  - *buffer errors*
  - *permissions/privileges/access control*
  - *resource management*
  - *information leakage*
  - *numeric errors*
  - *cryptographic errors*
  - *code injection*

BESSPIN’s Role in SSITH

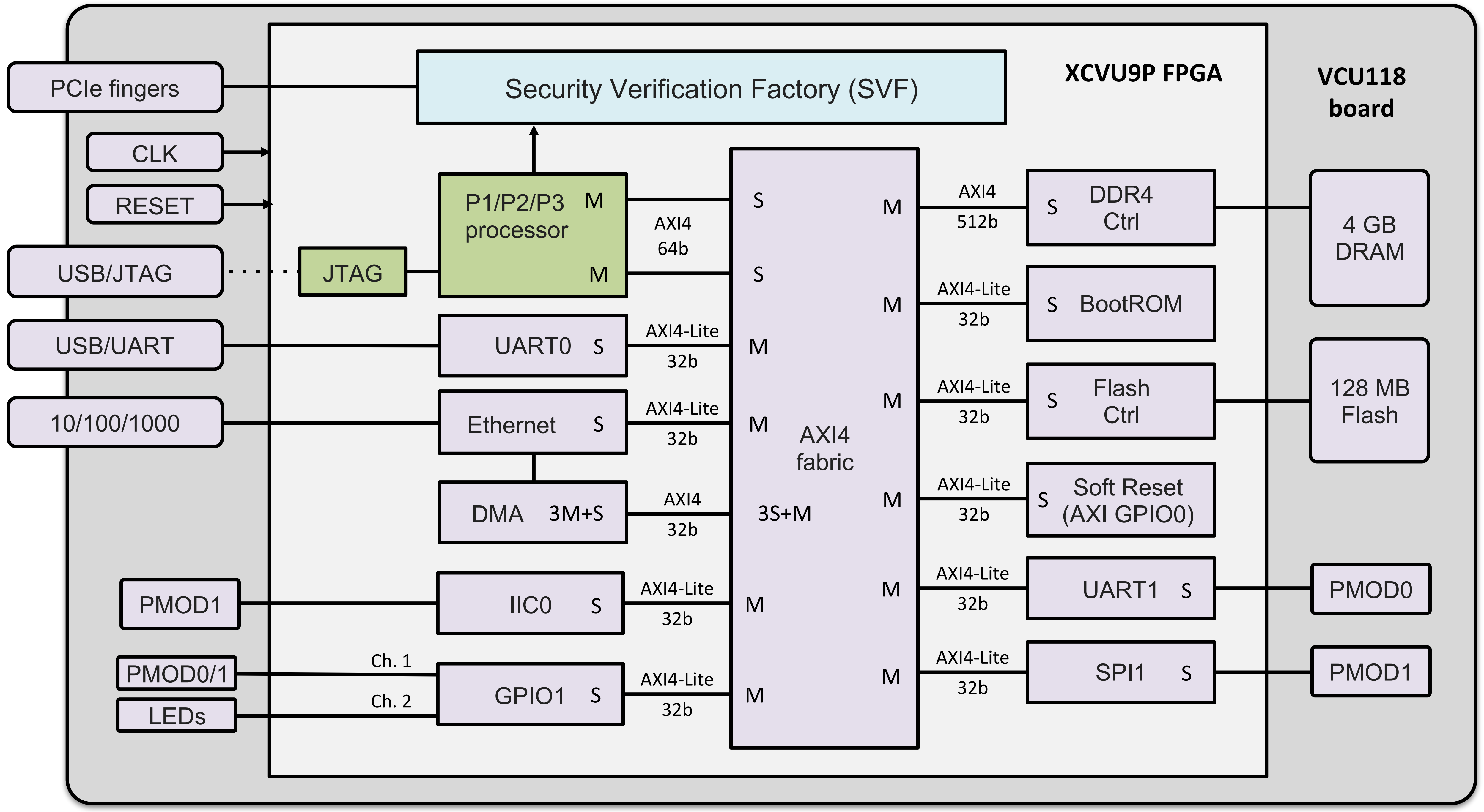
- precisely define the seven SSITH vulnerability classes
- provide tools that integrate with normal design flows and generate evidence that a processor is correct and secure
- create new tools that enable hardware designers to objectively trade off security against traditional hardware metrics (power, performance, area)



#	Filename	Date	Last update	# Nb of features
1	secure_cpu_example_flattened.cfr	2019-05-16 21:21:24	2019-05-16 21:25:10	7
2	secure_cpu_example_flattened.cfr	2019-05-16 17:26:23	2019-05-16 17:26:23	0
3	secure_cpu_example_flattened2.cfr	2019-05-16 17:03:55	2019-05-16 21:26:05	7
4	windows.cfr	2019-05-16 10:40:31	2019-05-16 16:28:56	2
5	windows.cfr	2019-05-16 10:34:06	2019-05-16 10:34:06	0
6	swerv-pregen.cfr	2019-05-16 10:30:43	2019-05-16 21:28:39	10
7	swerv-pregen.cfr	2019-05-16 09:06:40	2019-05-16 09:54:35	6
8	secure_cpu_example_flattened2.cfr	2019-05-15 17:21:57	2019-05-16 14:51:51	10
9	secure_cpu_example_flattened.cfr	2019-05-15 17:21:26	2019-05-15 17:21:46	8
10	windows.cfr	2019-05-15 17:21:03	2019-05-15 17:21:11	3
11	secure_cpu_example_flattened2.cfr	2019-05-15 17:20:37	2019-05-15 17:20:54	4
12	swerv-pregen.cfr	2019-05-15 17:19:50	2019-05-15 17:20:26	9

BESSPIN's Expanded Role in SSITH

- develop the baseline setup (FPGA boards, I/O interfaces, processors, OSes, drivers, etc.) that 6 other teams will extend with security enhancements
- develop two versions of each of three "levels" of RISC-V processor:
  - 32-bit, no MMU (P1)
  - 64-bit, MMU (P2)
  - 64-bit, MMU, OOO (P3)



Red Teaming SSITH Security

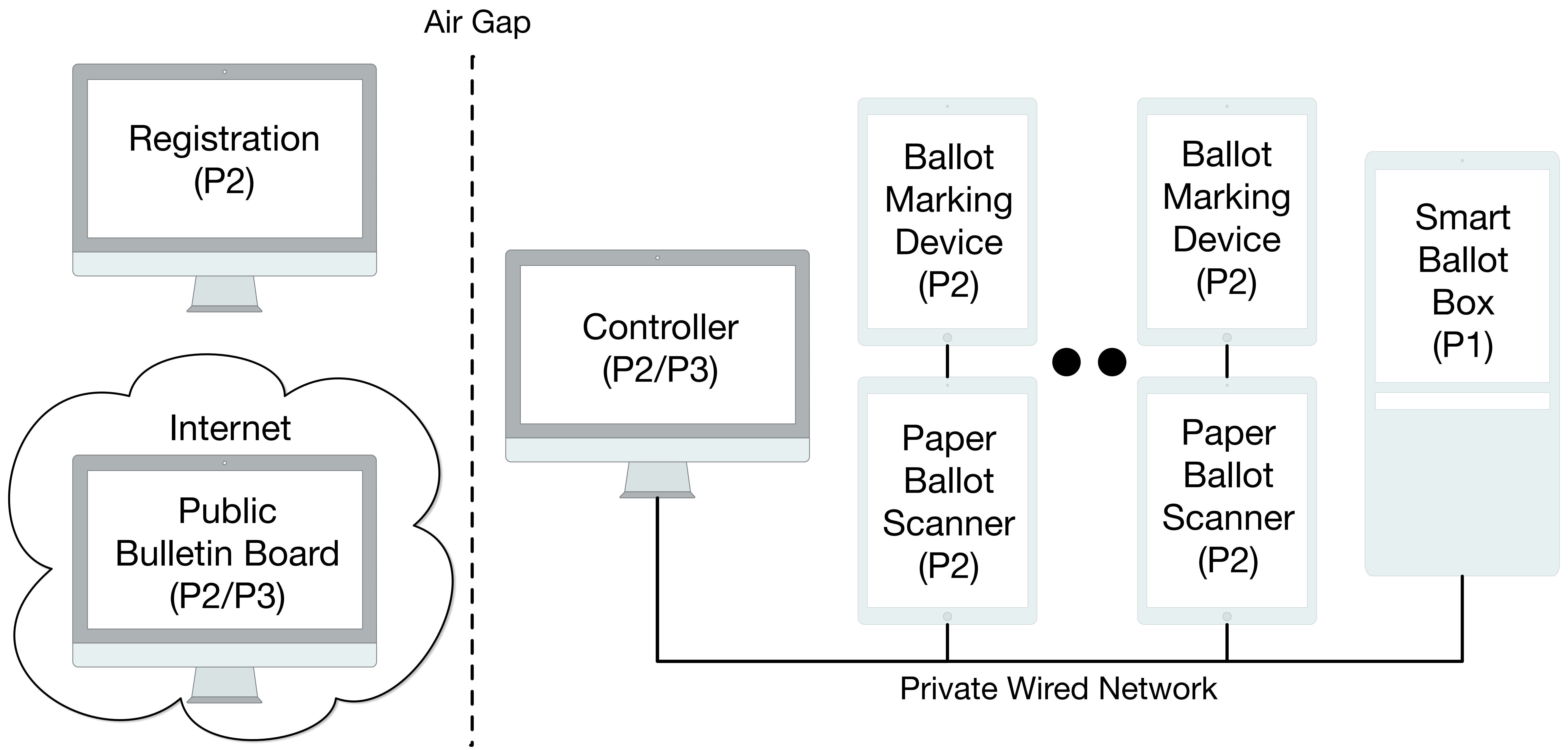
- how can DARPA evaluate the security of 6 teams' work including 18 SoCs across 6 architectures, 3 OSs, and 6 compilers?
- these systems are meant to be secure…
  - *even when their entire design and implementation is public…*
  - *even when the system's network is compromised…*
  - *even when the adversary has a beachhead and can install malware!*
- a set of red teams would take years to accomplish this evaluation, be very expensive, and likely provide little useful output

Red Teaming SSITH Security

- the answer: *a public red team exercise with a compelling demonstration project…*

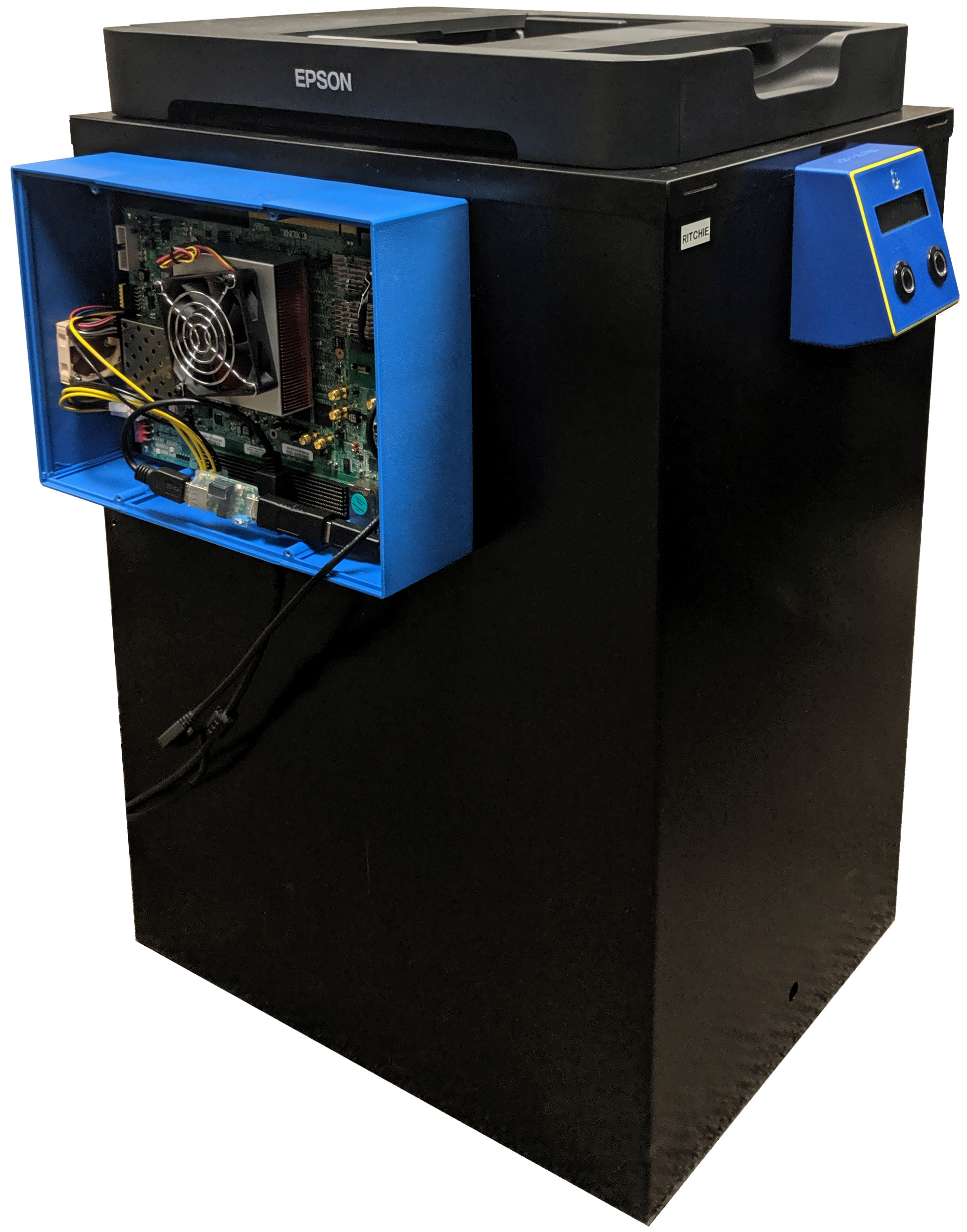
Red Teaming SSITH Security

- the answer: *a public red team exercise with a compelling demonstration project… an E2E-V voting system!*


Air Gap
Registration (P2)
Internet
Public Bulletin Board (P2/P3)
Controller (P2/P3)
Ballot Marking Device (P2)
Paper Ballot Scanner (P2)
Ballot Marking Device (P2)
Paper Ballot Scanner (P2)
Smart Ballot Box (P1)
Private Wired Network

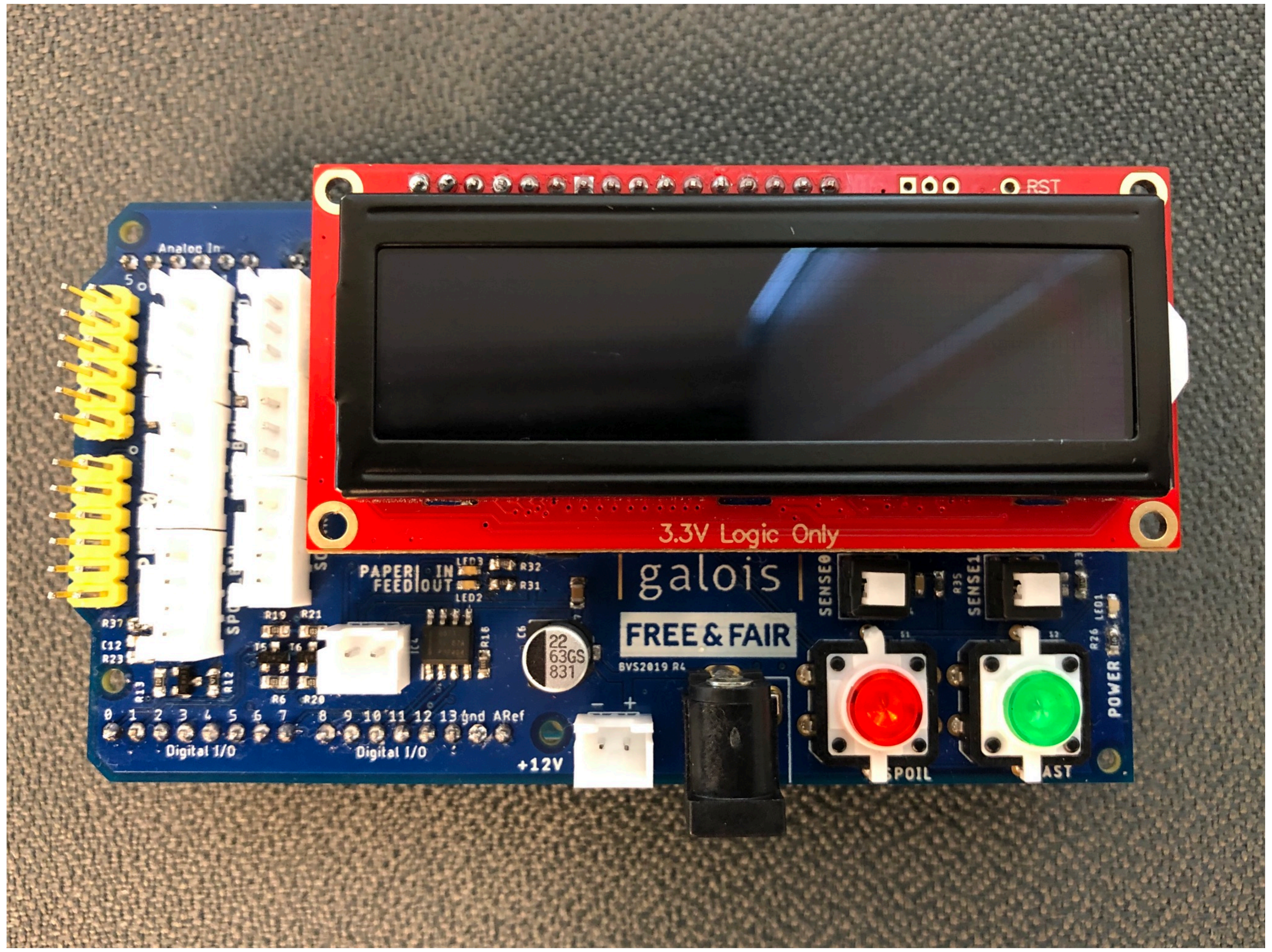
BESSPIN Voting System

- demonstrations at DEF CON 2019 and (planned) 2020
- 2019 smart ballot boxes were:
    - a cheap filing cabinet with a hole cut in the top
    - an expensive FPGA board for emulating a SSITH processor (left)
    - a custom control panel (right)
    - a disembodied Epson printer/scanner lid for paper feeding (top)



BESSPIN Voting System

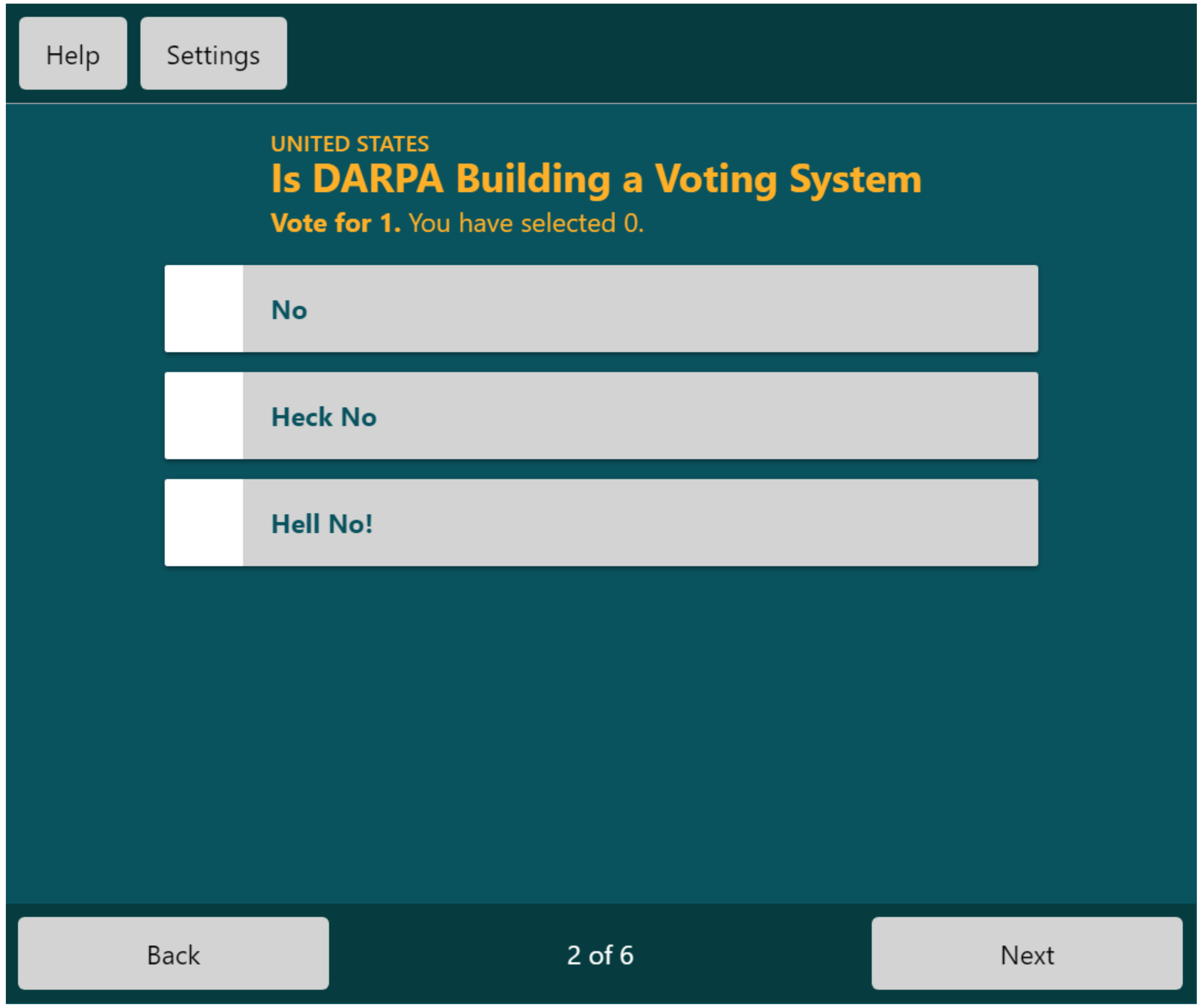
- in addition to a system with all the E2E-V components on the full SSITH hardware platforms, we're developing a low-cost hardware platform (CASCADES) and pedagogical materials to facilitate worldwide democratized red teaming



www.crowdsupply.com/free-and-fair/cascades

BESSPIN Voting System Assurance

- E2E-V systems don't *need* to be verified because they're *software independent* – but it helps things run smoothly!
- implemented in a verifiable subset of C, with Cryptol implementations for protocols and cryptography and ACSL specifications for C functions
- will include 21CC cryptographic circuits as part of system hardware
- using Frama-C tool suite, Cryptol, and SAW for assurance

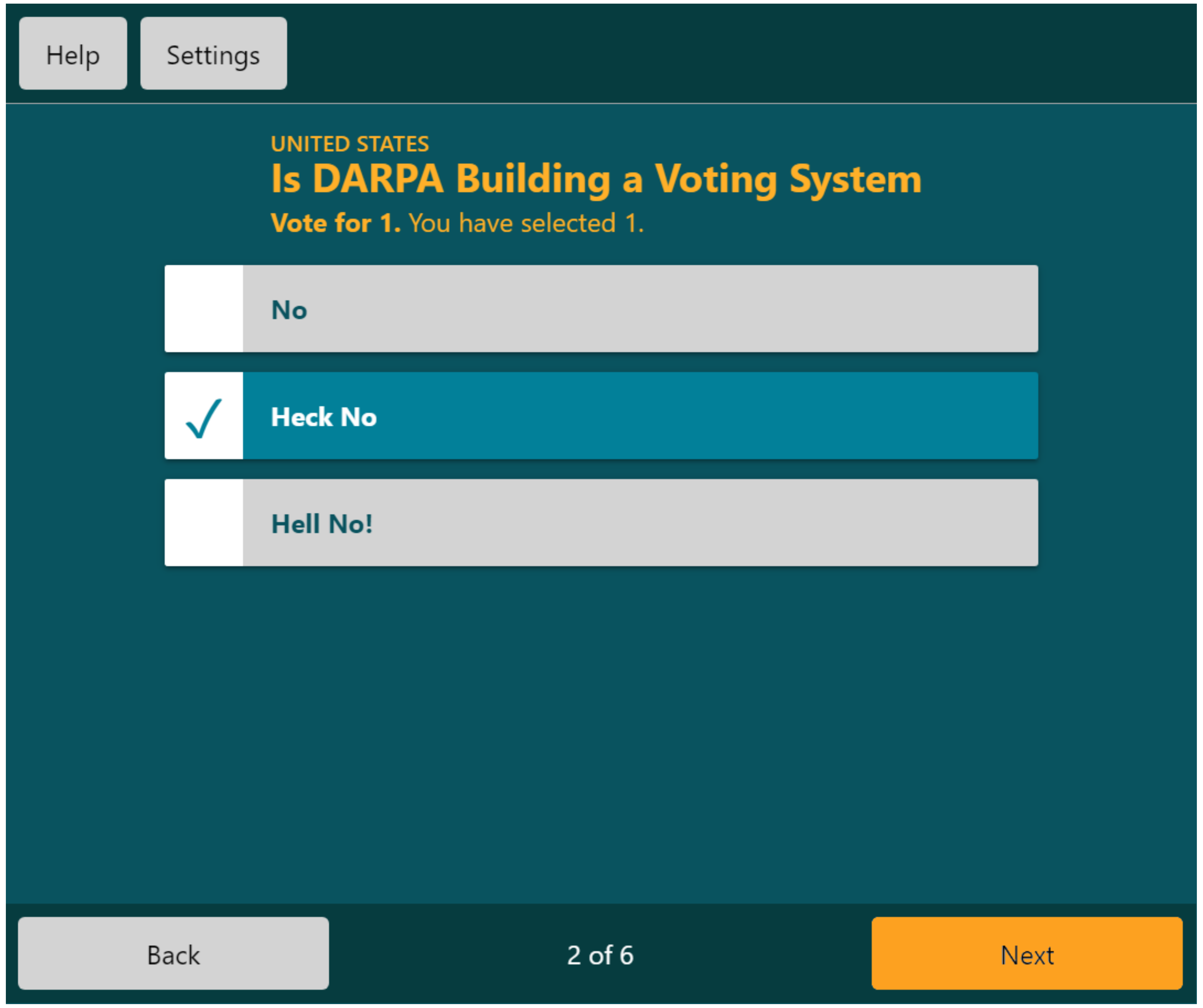


UNITED STATES
Is DARPA Building a Voting System
Vote for 1. You have selected 0.

- No
- Heck No
- Hell No!

BESSPIN Voting System Assurance

- while DARPA is *not* building a voting system for use in real elections…

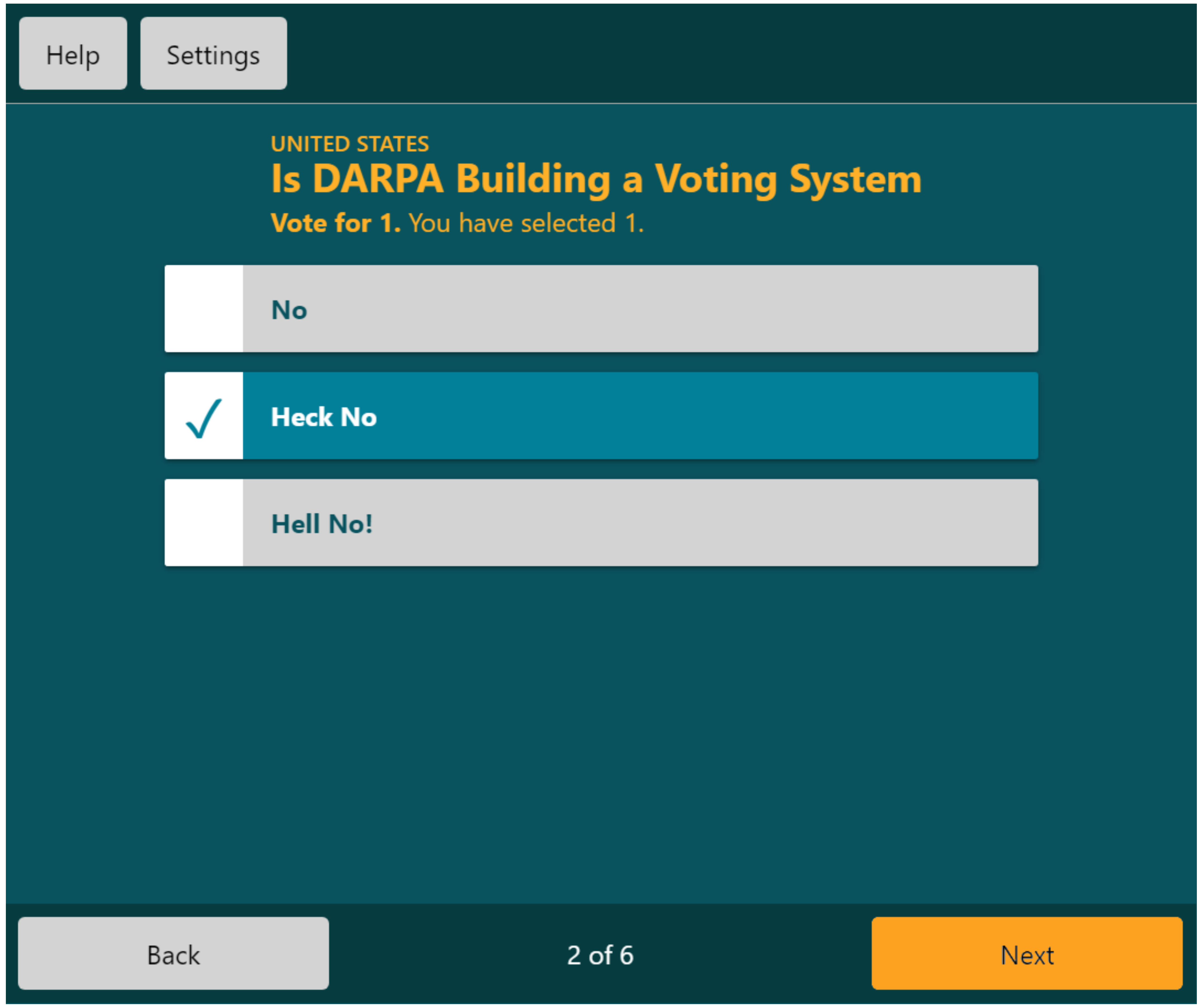


UNITED STATES
Is DARPA Building a Voting System
Vote for 1. You have selected 1.

No
[✓ Heck No]
Hell No!

BESSPIN Voting System Assurance

- while DARPA is *not* building a voting system for use in real elections…
- we're doing our best to make the BVS a role model for the companies that *are*!
- it's all open source: securehardware.org



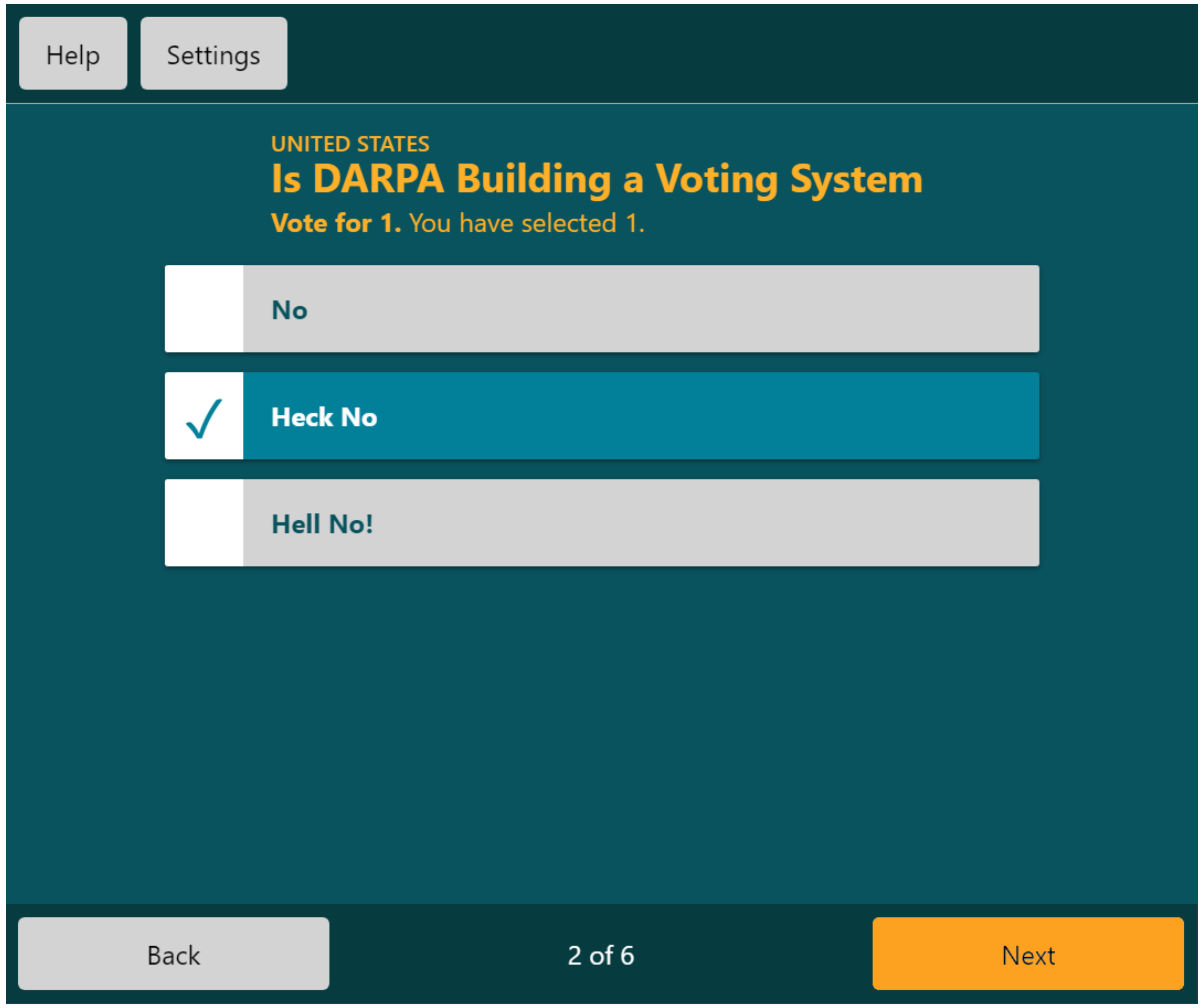
UNITED STATES
Is DARPA Building a Voting System
Vote for 1. You have selected 1.

No
[✓ Heck No]
Hell No!

BESSPIN Voting System Assurance

- while DARPA is *not* building a voting system for use in real elections…
- we're doing our best to make the BVS a role model for the companies that *are*!
- it's all open source: securehardware.org

- Questions?



UNITED STATES
Is DARPA Building a Voting System
Vote for 1. You have selected 1.

No
[✓ Heck No]
Hell No!