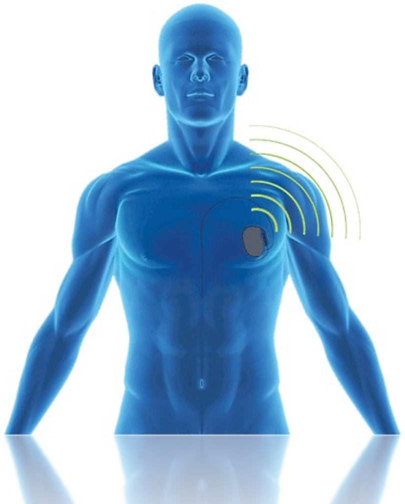
Towards An Open-Source, Formally-Verified Secure Enclave

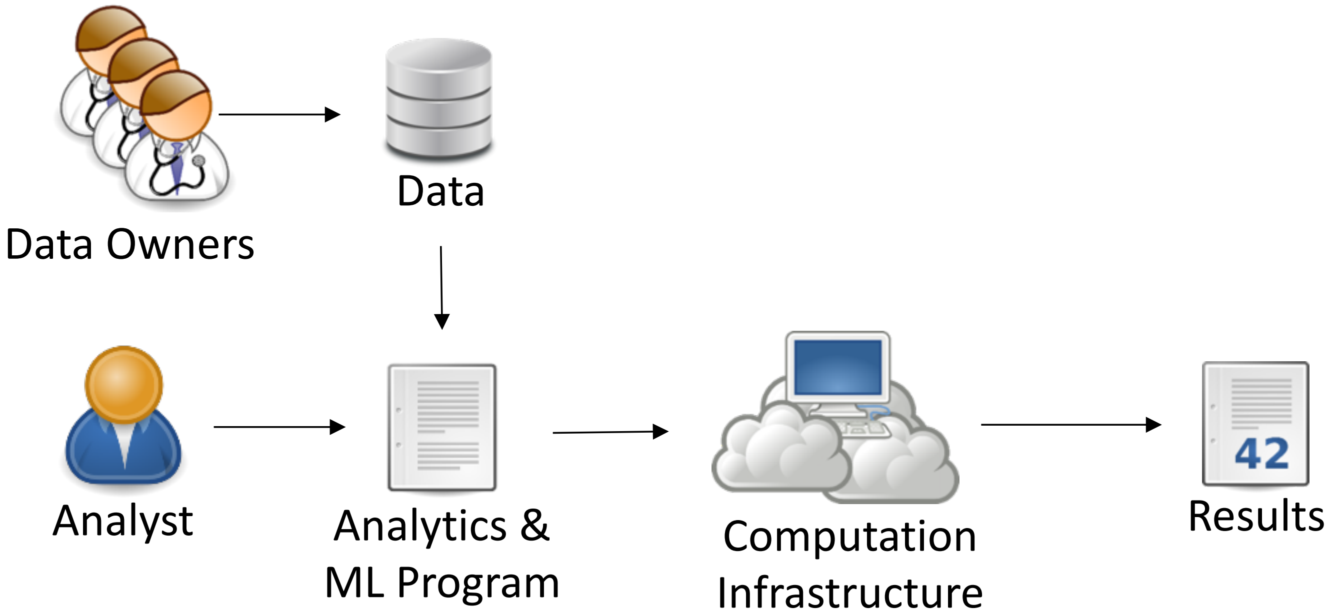
Dawn Song
UC Berkeley

The Age of Big Data

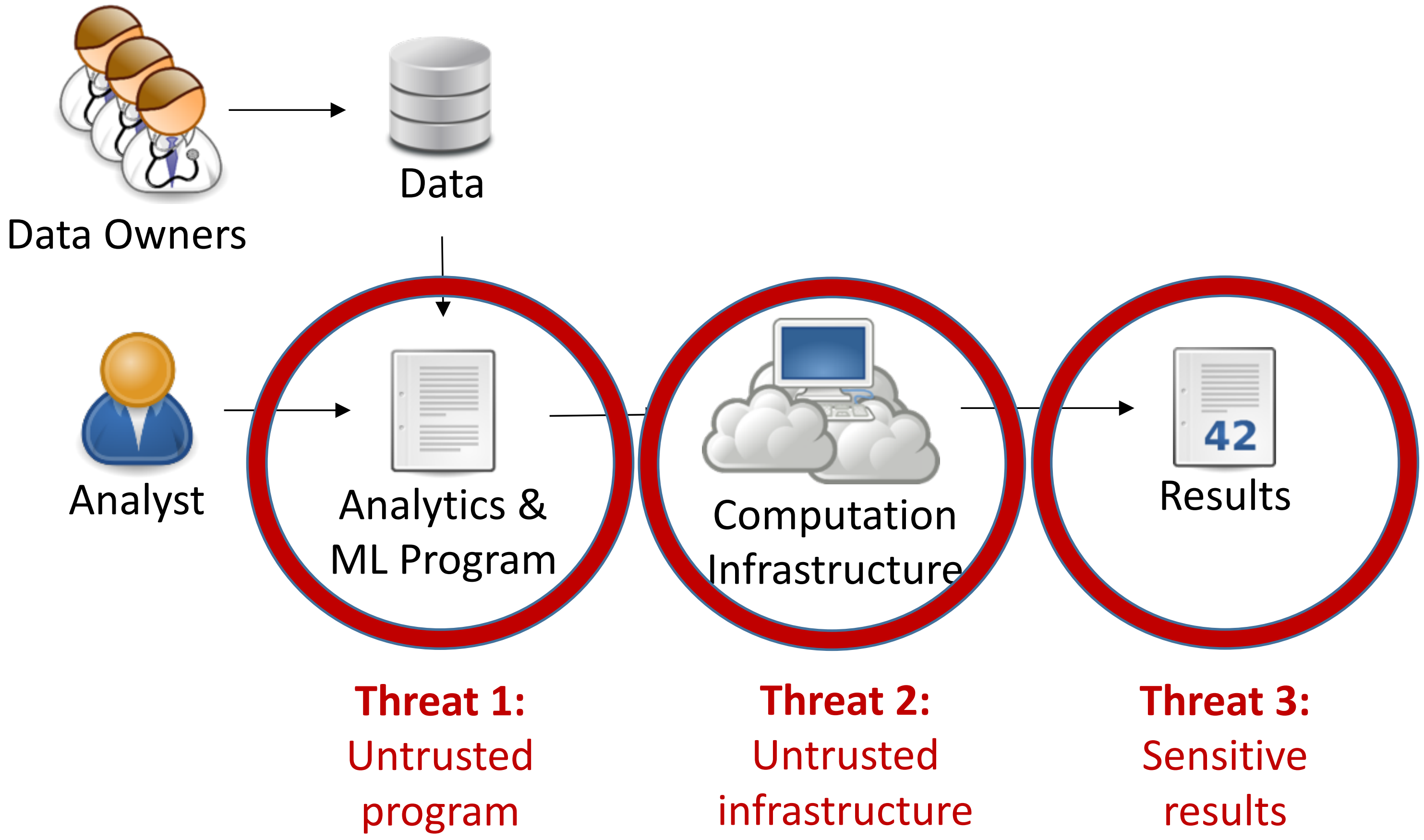


Plentiful, and Private

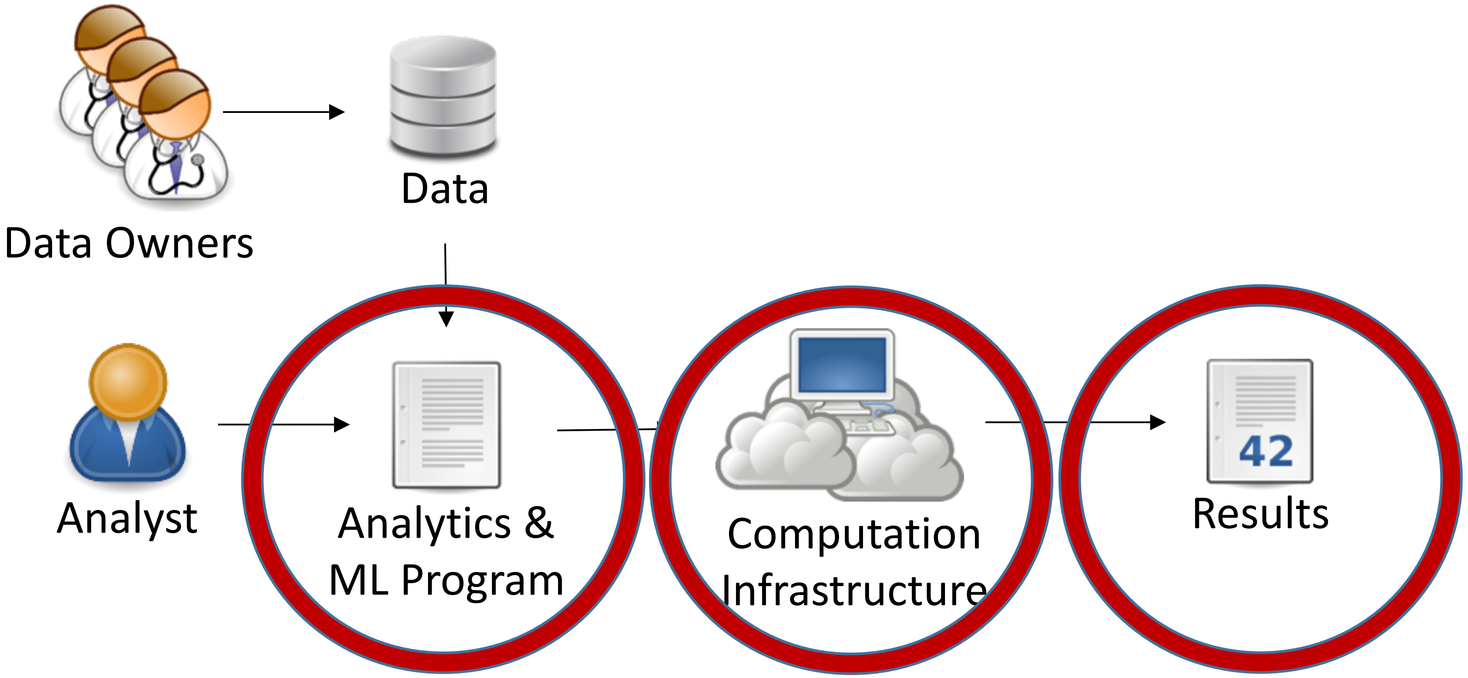
Current Frameworks for Data Analytics & Machine Learning



Current Frameworks Insufficient

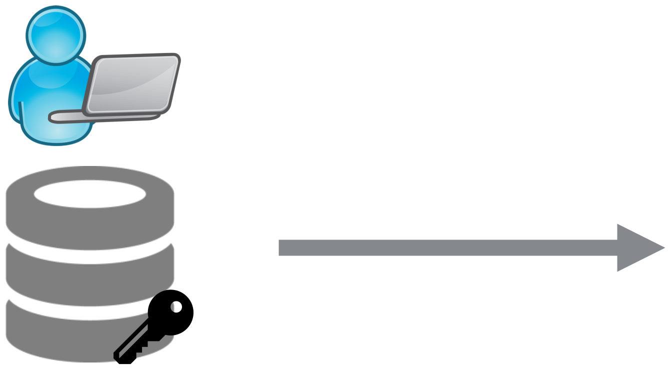


Desired Solutions for Confidentiality/Privacy



Threats	**Threat 1:** Untrusted program	**Threat 2:** Untrusted infrastructure	**Threat 3:** Sensitive results
Desired Solutions	**Program Rewriting & Verification**	**Secure Computation**	**Differential Privacy**

Secure Computation: Simulating Trusted Third Party



- Does my secret data remain secret?
- Does the program execute as it is supposed to?
- Is the right program executed?

Secure Computation

- Example:
  - Build a word-embedding from everyone’s text messages on their phones
- Challenge:
  - Text messages are highly sensitive
  - Computation infrastructure may not be trusted
- Solutions:
  - Crypto-based approach:
    - Non-interactive: Fully-homomorphic encryption (FHE)
    - Interactive: Multi-party computation (MPC)
  - Hardware-based approach:
    - Secure enclave provides isolation & remote attestation

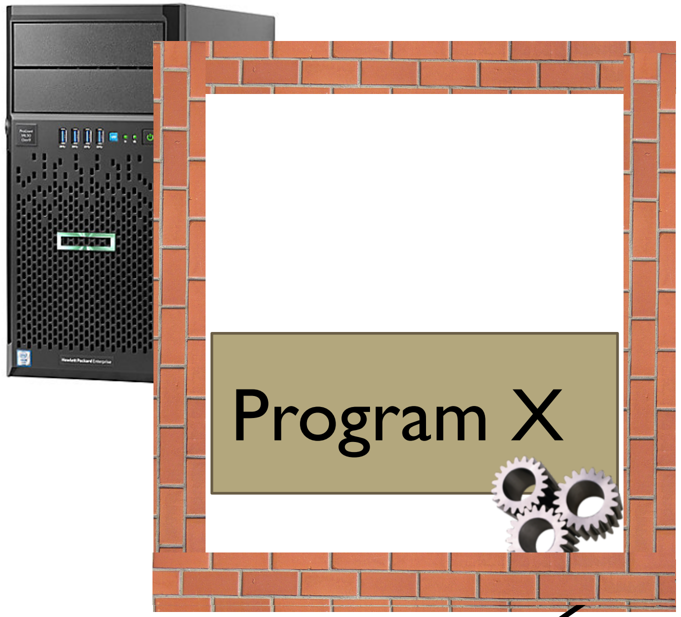
Crypto-based Secure Computation

- Fully-homomorphic encryption (FHE)
  - Given E(x), f, compute E(f(x))
  - Support general secure computation with strong security
  - High performance overhead: 10^6
  - Example: CryptoNet [Dowlin et al.]
    - Classification of an encrypted image using neural networks
    - On MNIST:
      - 51000 predictions per hour on a single PC
      - 579 seconds latency per image
- Multi-party computation (MPC)
  - Trust model: at most t out of k parties are malicious
  - Require many rounds of communication among different parties

Hardware-based secure computation

- Trusted Execution Environment (e.g., Intel SGX)
  - Secure enclave: isolation & attestation
    - Protect against malicious OS
  - Enable practical secure computation over encrypted data
    - In contrast to fully-homomorphic encryption (FHE) with 10^6 performance overhead
  - Many other security applications

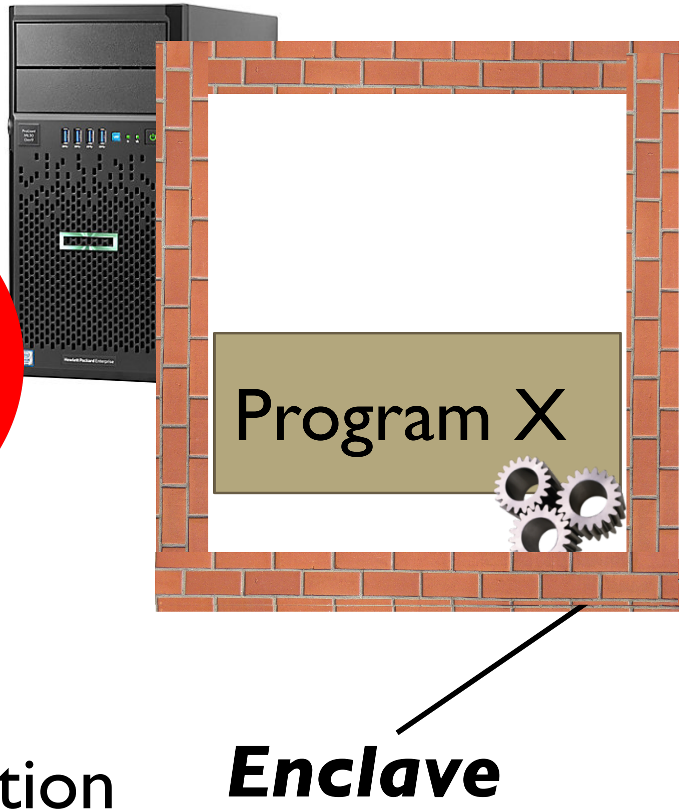
Trusted Execution Environment (TEE)



Enclave

Trusted Execution Environment (TEE)

Integrity

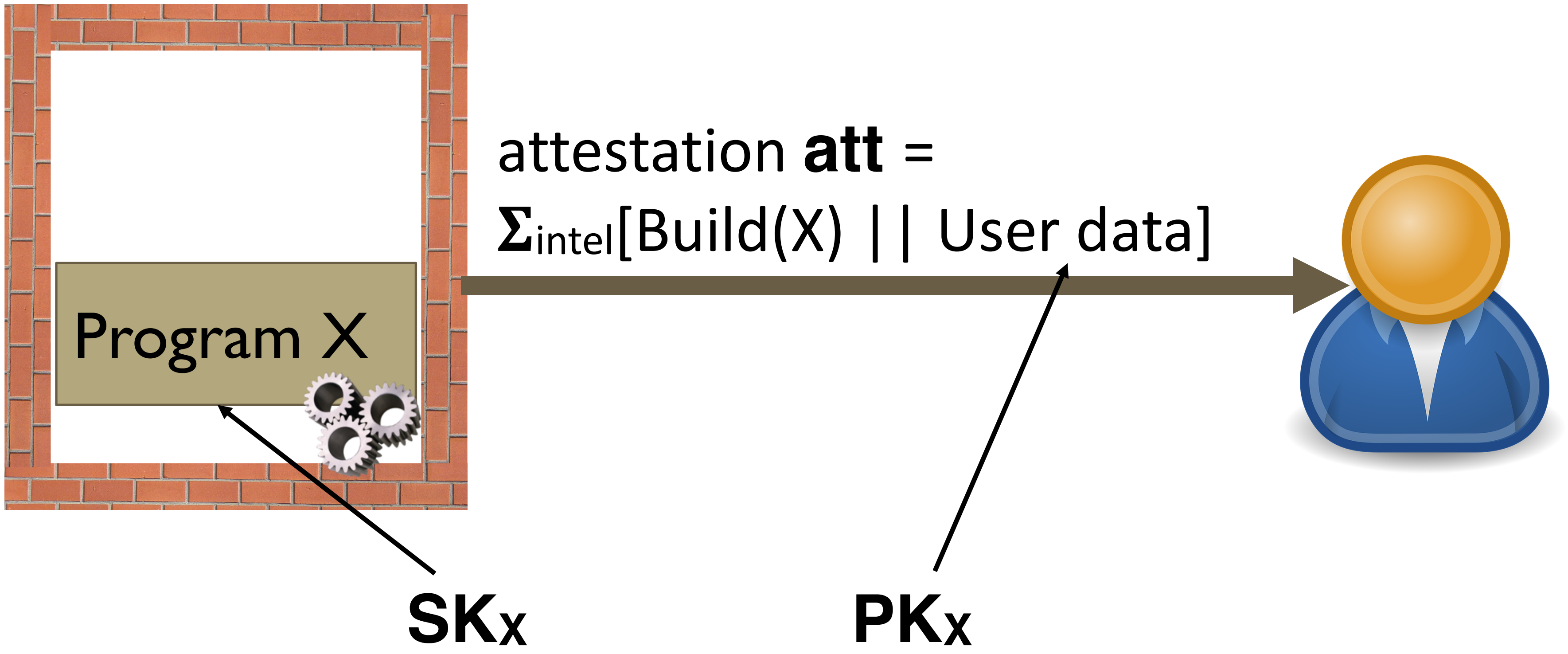


Confidentiality

OS and other processes cannot tamper with execution of X.

OS and other processes cannot learn state of X.*

Remote attestation



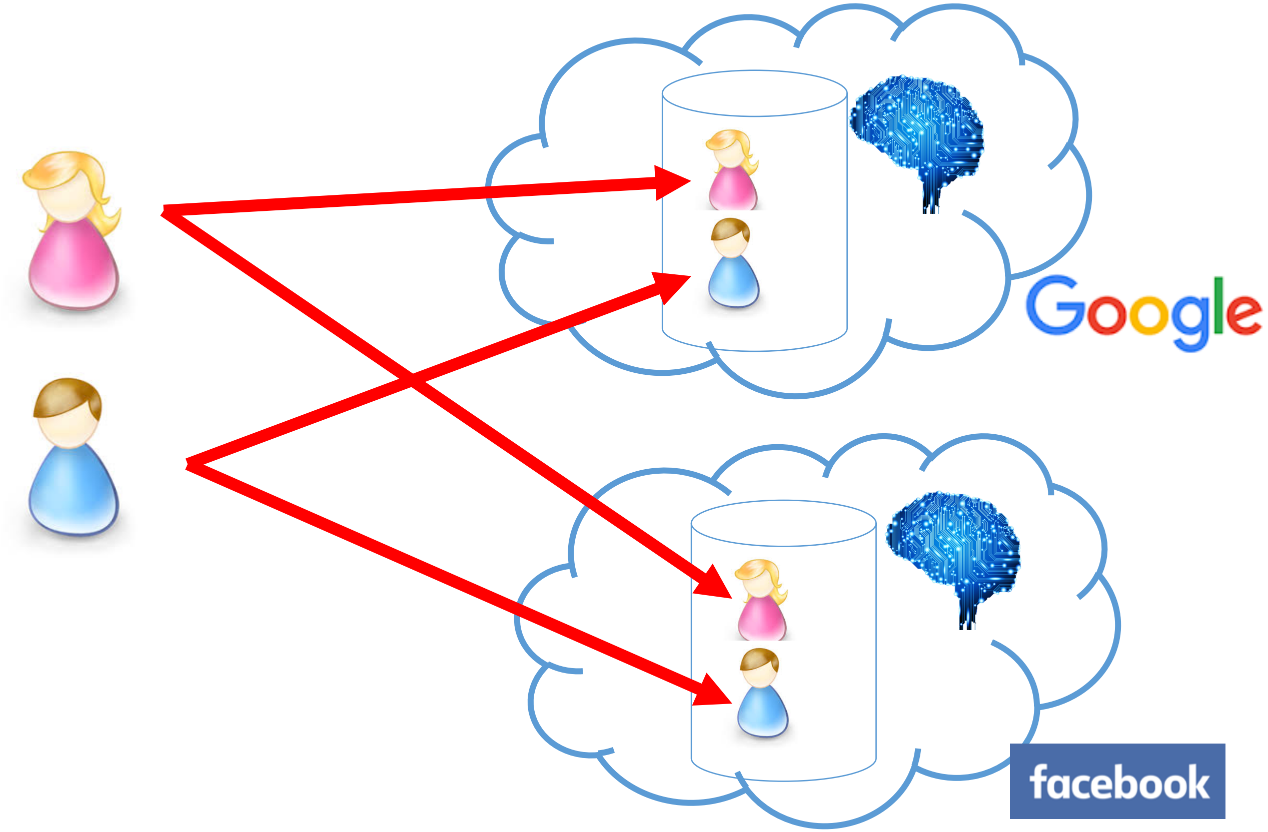
Secure Enclave as a CornerStone Security Primitive

- Strong security capabilities
  - Authenticate “itself (device)”
  - Authenticate “software”
  - Guarantee the integrity and privacy of “execution”
- Platform for building new security applications
  - Couldn’t be built otherwise for the same practical performance
  - Many examples
    - Haven [OSDI’14], VC3 [S&P’15], M2R[USENIX Security‘15], Ryoan [OSDI’16], Opaque [NSDI’17]
      - Single node or distributed computation using trusted hardware

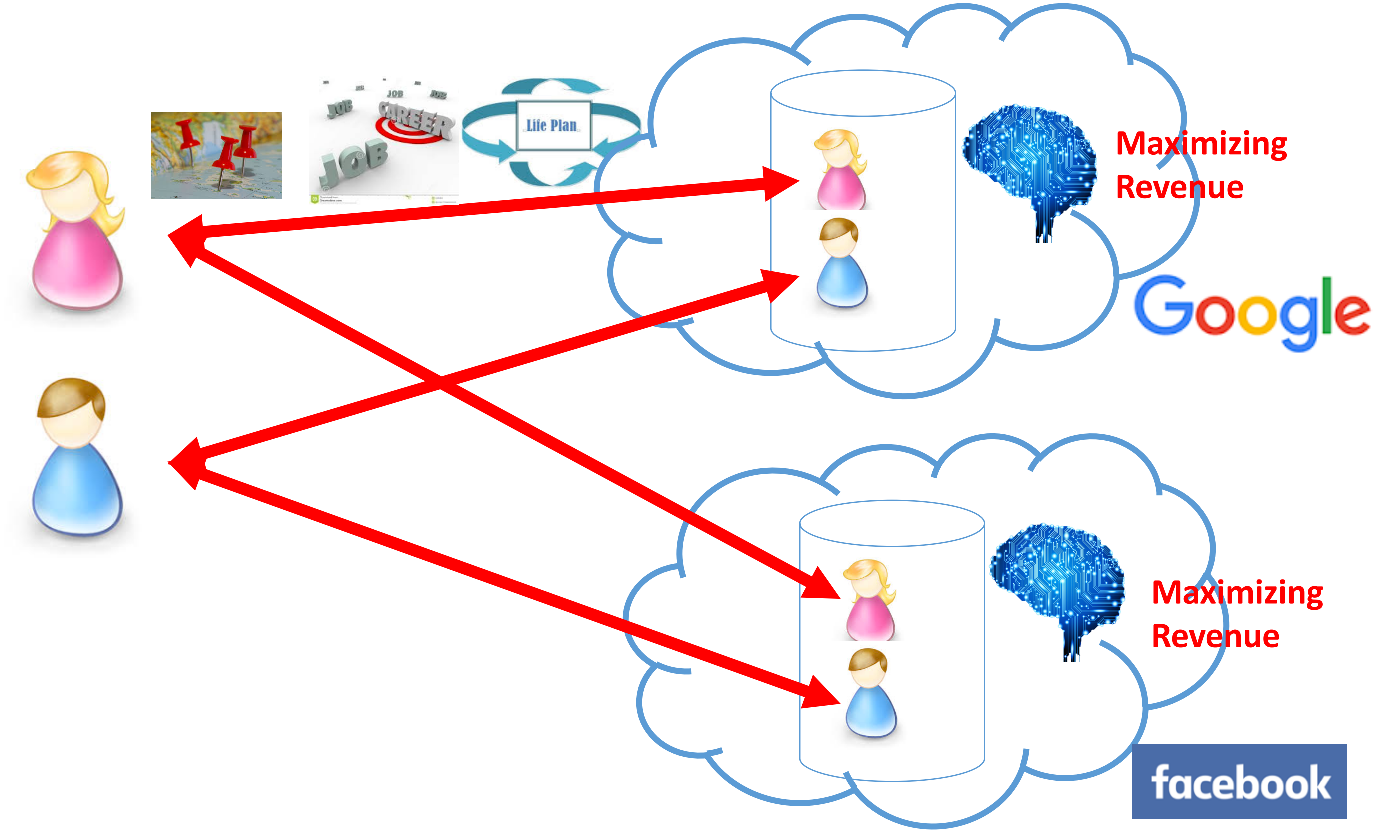
Whoever Controls & Leads in AI Will Rule the World

--Nation State Leaders

The Status Quo Today

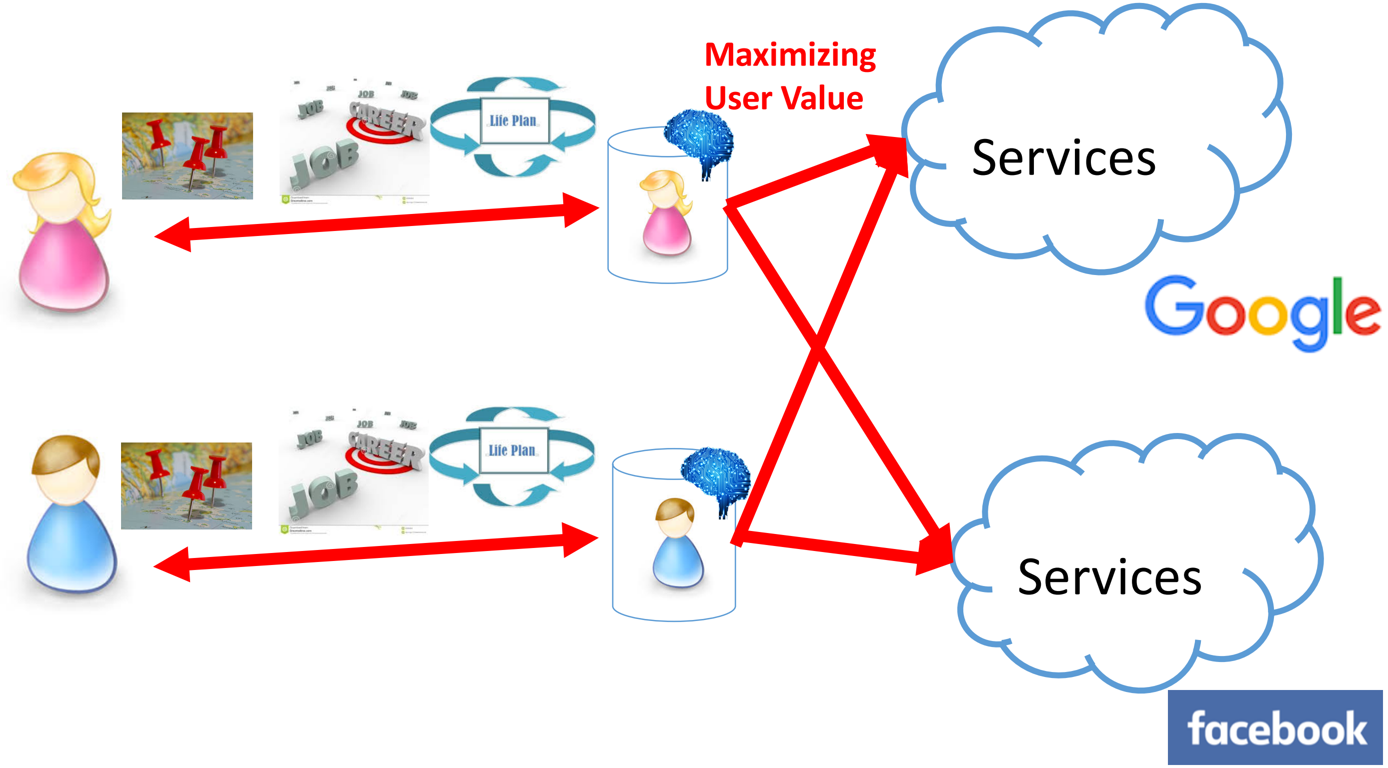


Who Will Be Running Our Lives?



Is there a different future?

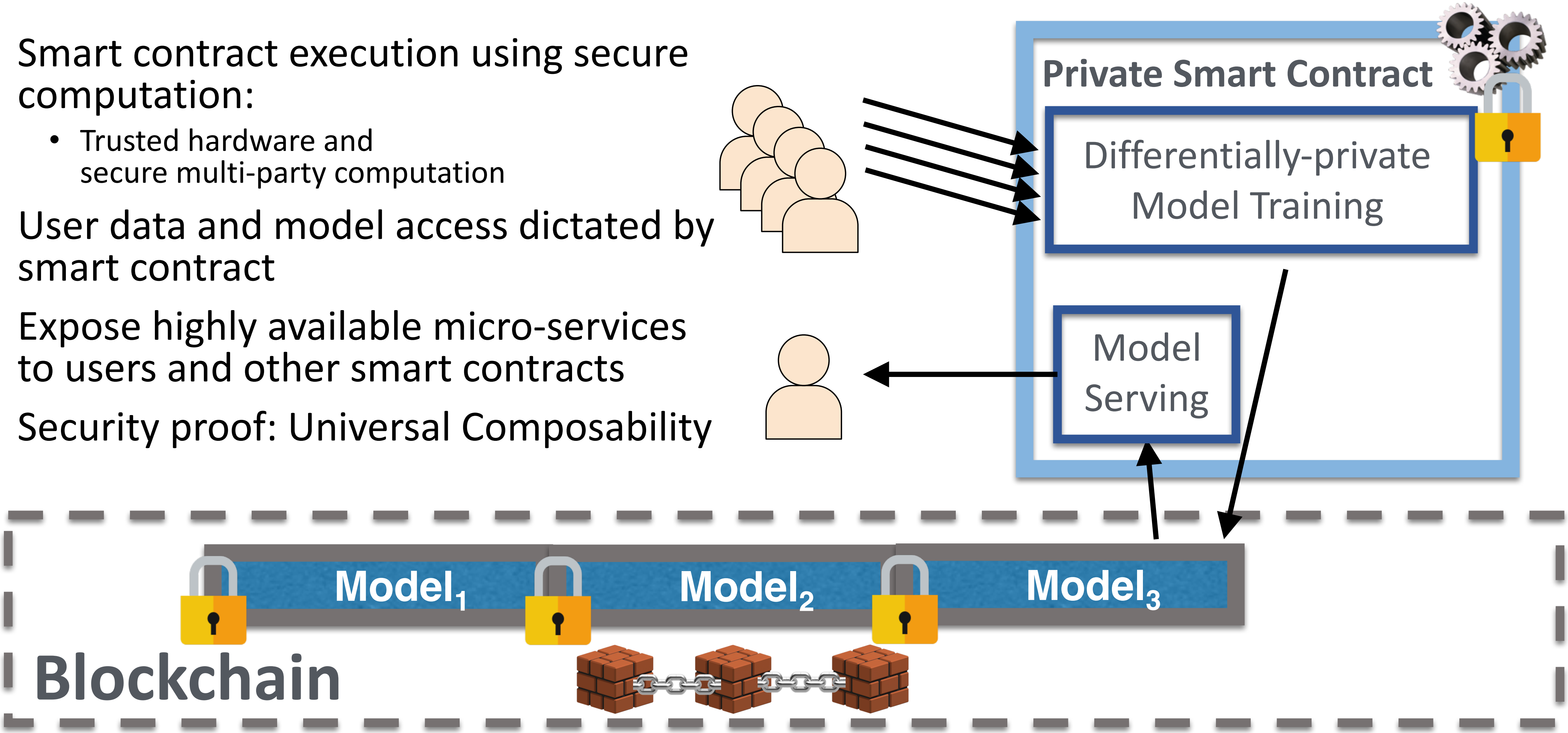
Intelligent Agent/Virtual Assistant under User Control



AIEden: Blockchain for Democratizing AI

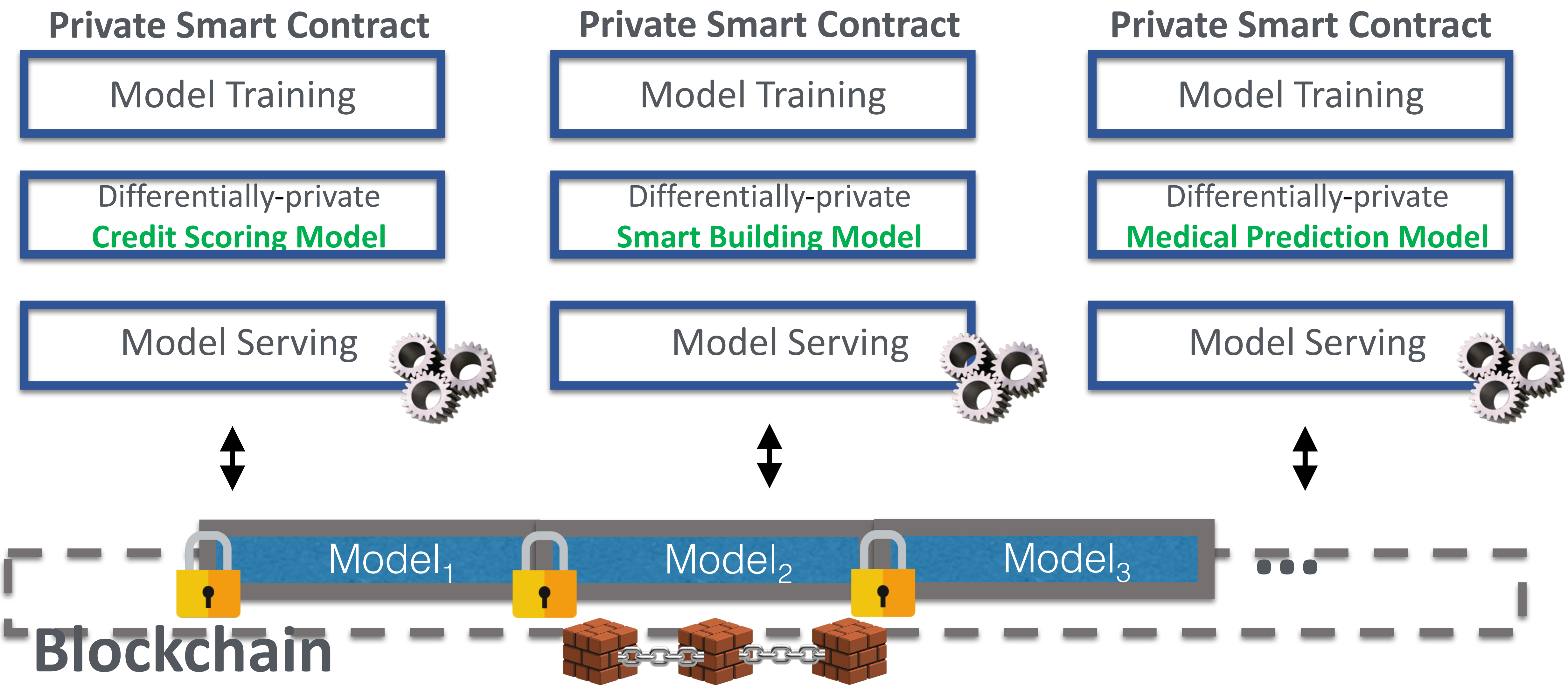
Ekiden: Confidentiality-preserving Smart Contract

- Smart contract execution using secure computation:
  - Trusted hardware and secure multi-party computation
- User data and model access dictated by smart contract
- Expose highly available micro-services to users and other smart contracts
- Security proof: Universal Composability



Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contract Execution, Cheng et al., https://arxiv.org/abs/1804.05141

Towards Blockchain of Intelligent Smart Contracts



Growth in Secure Hardware Deployment

SGX: Software Guard Extensions

- All Intel Core Processors since August 2015 (6th-Generation and later)
- SGX v2 expected to release in relatively near future

AMD Secure Processor

- Built into AMD Accelerated Processing Units (APUs)

SEV: Secure Encrypted Virtualization

- Introduced in EPYC server processor line (2017)
- Provides confidentiality but not integrity

Trusted execution environment

- Hardware-based isolation and integrity for Tegra chipsets
- TLK (Trusted Little Kernel): open-source stack for TEE

GlobalPlatform Trusted Execution Environment

- Already embedded in more than 17 Billion devices

Challenges in Secure Hardware

- How secure can it be? Under what threat models?

- What would you entrust with secure hardware?
  - Your bitcoin keys
  - Financial data
  - Health data

Grand Challenge

- Can we create trustworthy secure enclave as a cornerstone security primitive?
- Widely deployed, enable secure systems on top
- A new secure computation era

Path to Trustworthy Secure Enclave

- Open source design
- Formal verification
- Secure supply-chain management

Importance of Open Source Secure Enclave Design

- None of the commercial TEE designs is opened to public
- Security guarantee relies on trusting a hardware vendor’s design
- No industry agreement on right solution for everything
- Open source provides transparency & enables high assurance
- Open source builds a community

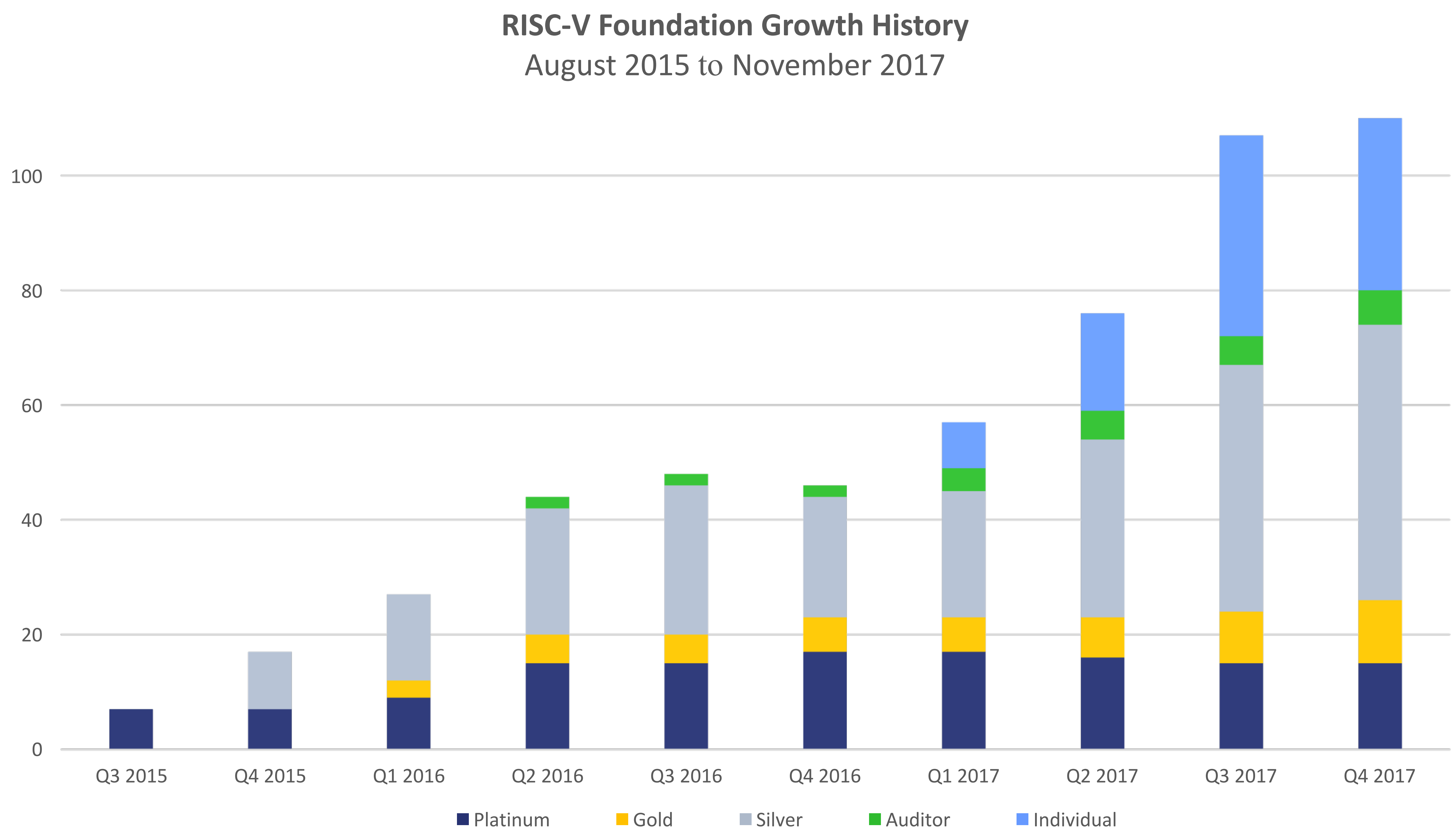
RISC-V Open-Source Hardware Ecosystem

- RISC-V: A high-quality, license-free, royalty-free RISC ISA specification originally from UC Berkeley
- Large companies started to adopt RISC-V for deeply embedded controllers in their SoCs (e.g., NVIDIA, Western Digital)
- India government, US DARPA, and Israel have adopted RISC-V
- Many startups choosing RISC-V for new products
- Becoming standard ISA for academic research

RISC-V ISA (**www.riscv.org**)

- A completely free and open ISA
  - Upstreamed GCC, Linux, glibc, LLVM, …
  - RV32, RV64, and RV128 variants for 32b, 64b, and 128b address spaces defined
- Base ISA only <50 integer instructions, but supports compiler, linker, OS, etc.
- Extensions provide full general-purpose ISA, including IEEE-754/2008 floating-point
- Better/comparable ISA-level metrics to other ISAs but simpler
- Designed for extension, customization
- Seventeen 64-bit silicon prototype implementations completed at Berkeley so far (45nm, 28nm, 16nm)

RISC-V Foundation: 100+ Members


RISC-V Foundation Growth History
August 2015 to November 2017
100
80
60
40
20
0
Q3 2015
Q4 2015
Q1 2016
Q2 2016
Q3 2016
Q4 2016
Q1 2017
Q2 2017
Q3 2017
Q4 2017
Platinum
Gold
Silver
Auditor
Individual

Sanctum

- Secure processor design on RISC-V
- Fully-isolated per-enclave page table
- Defending against cache-based side-channel attacks

Sanctum: Minimal Hardware Extensions for Strong Software Isolation

Victor Costan, Ilia Lebedev, and Srinivas Devadas
victor@costan.us, ilebedev@mit.edu, devadas@mit.edu
MIT CSAIL

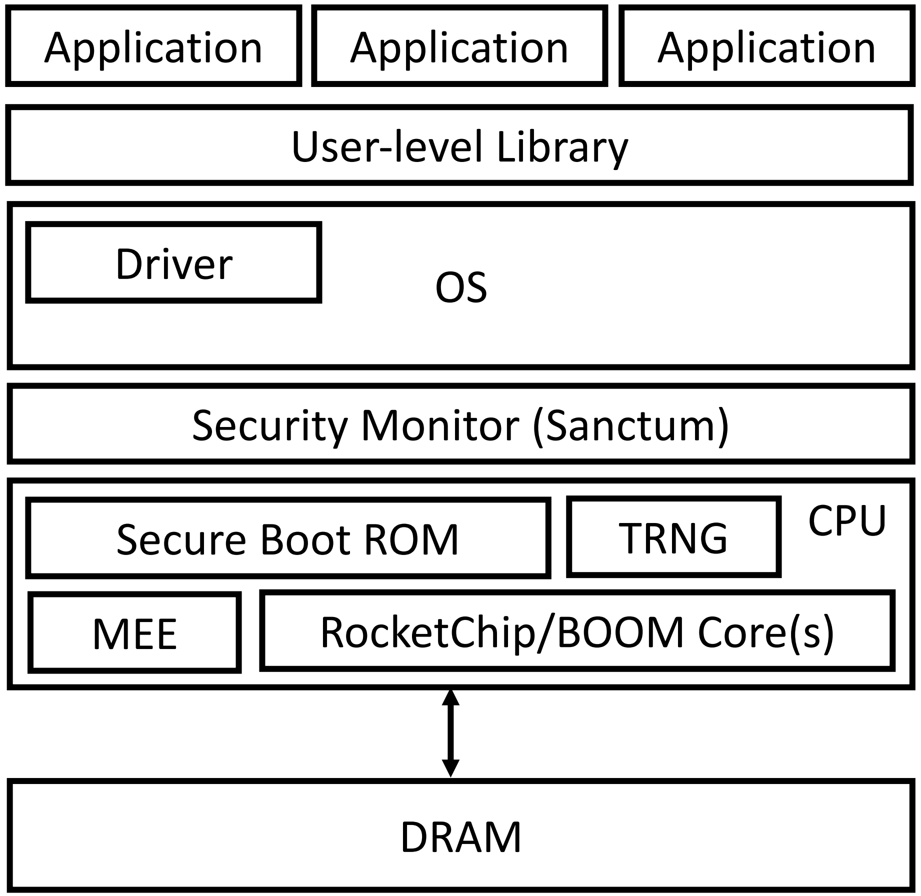
Abstract

Intel's Software Guard Extensions (SGX) have captured the attention of security practitioners by promising

the public's confidence in software systems has decreased considerably. At the same time, key initiatives such as cloud computing and the IoT (Internet of Things) require

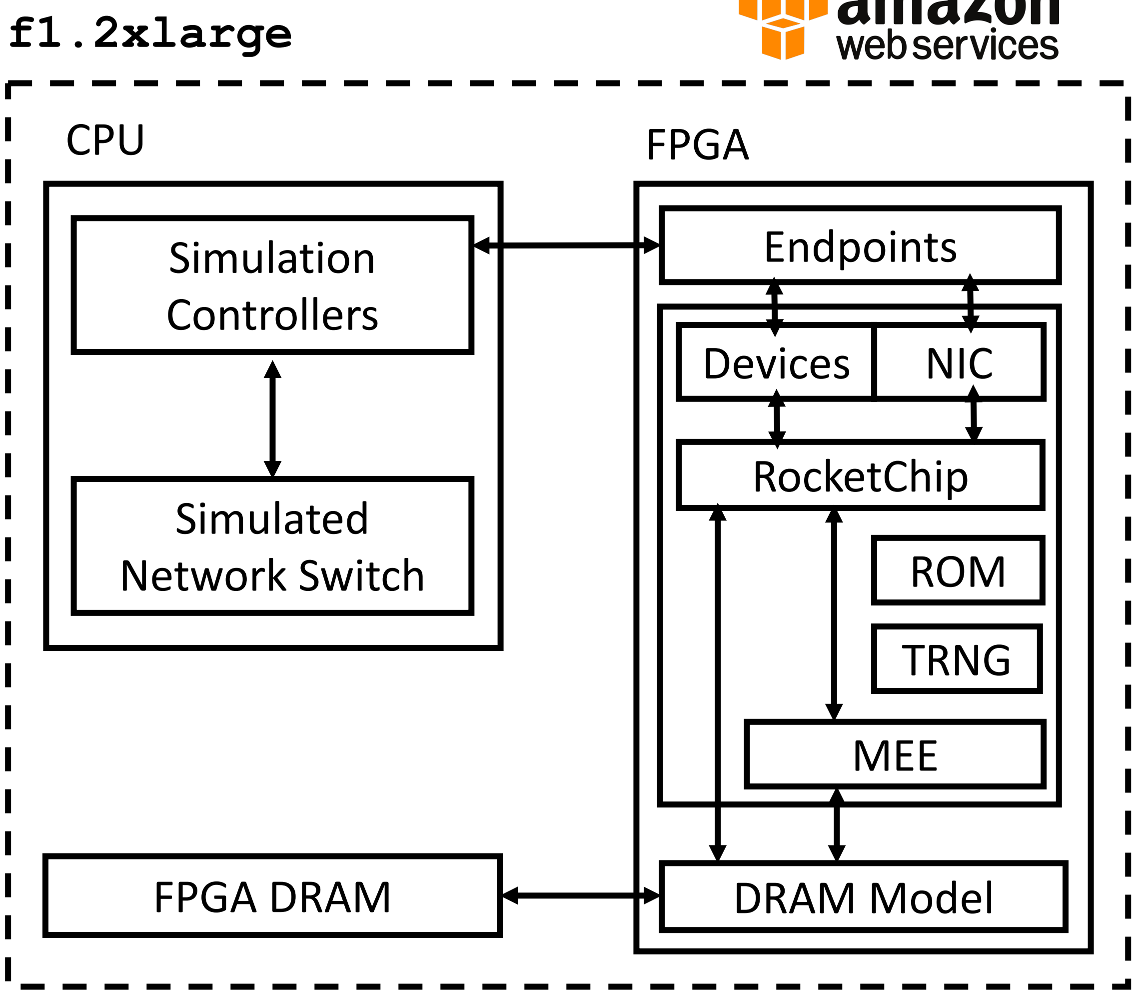
Project: Open Source Secure Enclave on RISC-V

- Full-stack open-source hardware enclave implementation for RISC-V processors
- CPU
  - RocketChip/BOOM: Berkeley-built Open-Source Cores
  - TRNG
  - Memory Encryption Engine (MEE)
- Hardware Enclave
  - MIT Sanctum: Software-based Hardware Enclave
- OS, Library, and Applications



FireSim: Simulation Framework

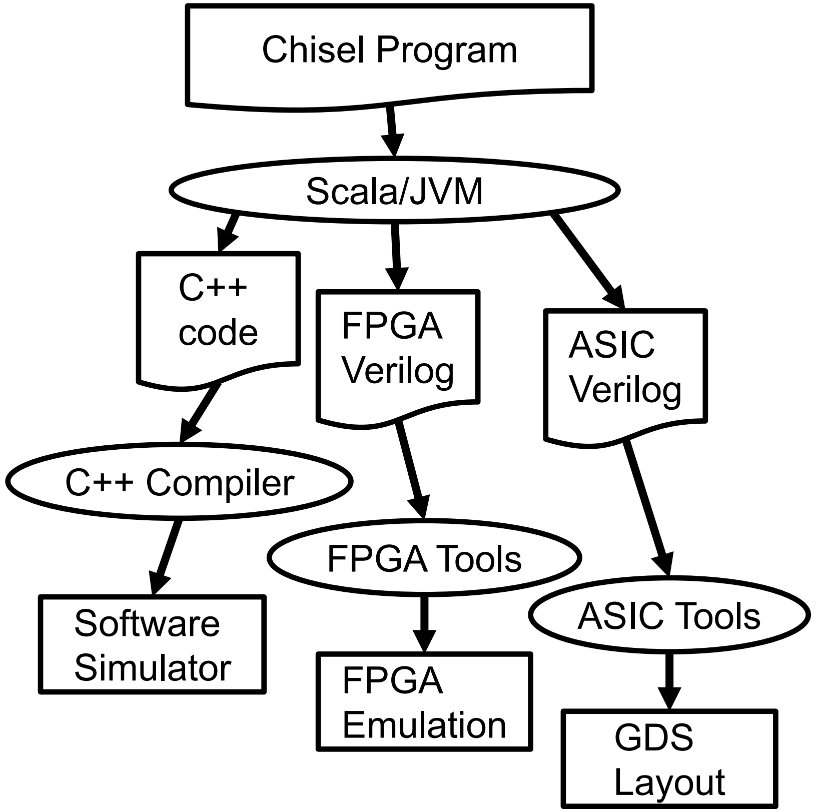
- FPGA-accelerated, Cycle-accurate simulator for arbitrary RTL in the public cloud (to appear in ISCA ’18)
- Uses a commodity host platform with FPGAs (EC2 F1)
- Lets users work with:
  - RTL (Chisel/Verilog) for customizing server blades, building accelerators, etc.
  - Software models (C++) for switches
- Runs real software stacks at reasonable speed (Linux + apps)



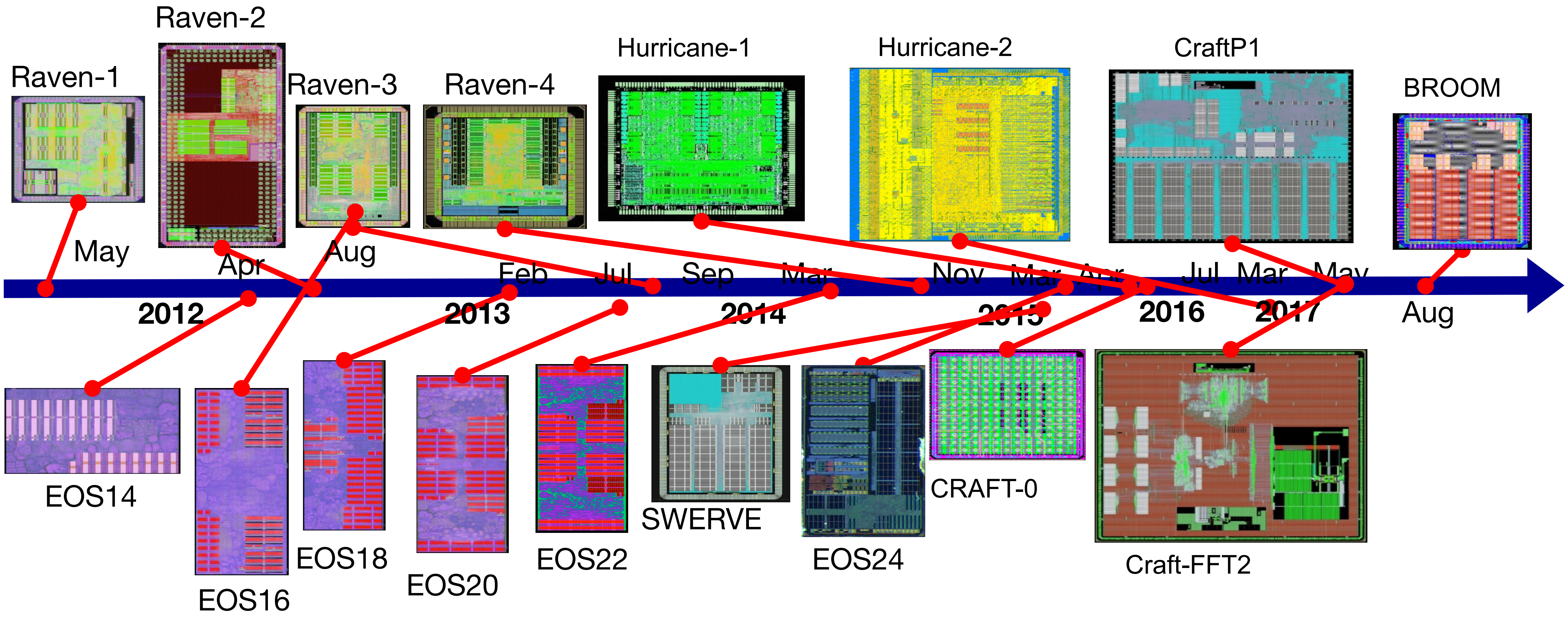
Full-stack Simulation of RISC-V TEE with FireSim

Chisel Construction Language

- Embed hardware-description language in Scala, using Scala’s extension facilities: Hardware module is just data structure in Scala
- Different output routines generate different types of output (C, FPGA-Verilog, ASIC-Verilog) from same hardware representation
- Full power of Scala for writing hardware generators
  - Object-Oriented: Factory objects, traits, overloading etc
  - Functional: Higher-order funcs, anonymous funcs, currying
  - Compiles to JVM: Good performance, Java interoperability



RISC-V Chips Designed at Berkeley



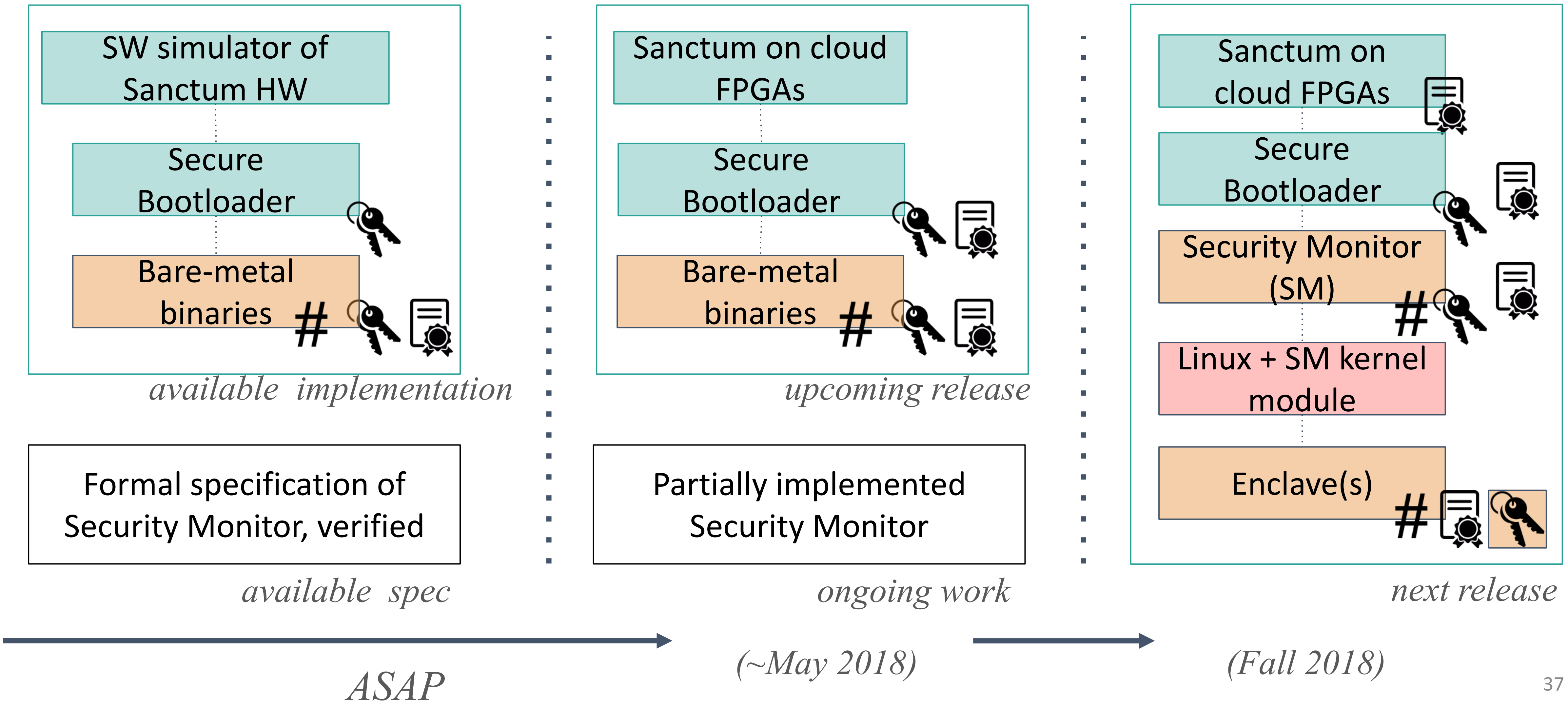
Goals

- Making hardware enclave usable to everyone
- Improving design using power of community
- Finding and patching security holes
- Exploring performance-security trade-offs for various threat models
- Finding solutions to address remaining problems
  - e.g., multi-node enclaves, side-channels, performance, and physical attacks

Timeline

Legend: *Can attest* · *Can sign* · *Is authenticated* #

trusted | authenticated | untrusted



Team





Ilia Lebedev
MIT
ilebedev@mit.edu

Srini Devadas
MIT
devadas@mit.edu

Dayeol Lee
UC Berkeley
dayeol@berkeley.edu

Krste Asanović
UC Berkeley
krste@berkeley.edu

Dawn Song
UC Berkeley
dawnsong@berkeley.edu

Massachusetts Institute of Technology

Berkeley
UNIVERSITY OF CALIFORNIA

Grand Challenge: Building Trustworthy Secure Hardware

- Open source design
- Formal verification
  - Subramanyan et al., A Formal Foundation for Secure Remote Execution of Enclaves [CCS 2017, Best Paper Award]
- Secure supply-chain management

Grand Challenge: Building Trustworthy Secure Hardware

More resources needed for research & development.

It requires community effort.

Let's tackle the big challenges together!