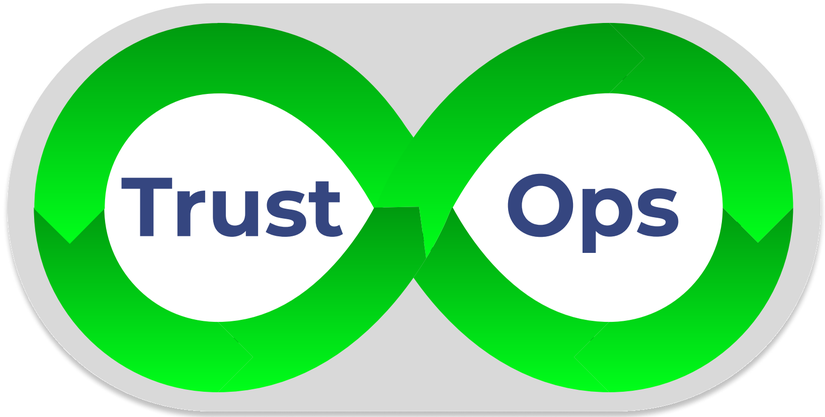


Continuously Building Trustworthy Software

Eduardo Brito, Fernando Castillo, Pille Pullonen-Raudvere, Sebastian Werner

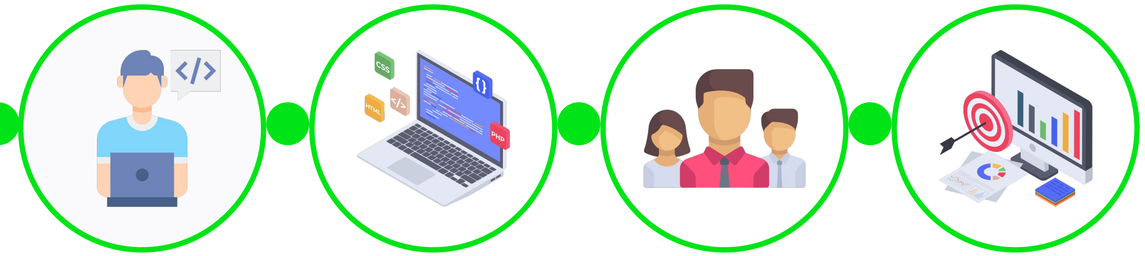
TEADAL

CYBERNETICA

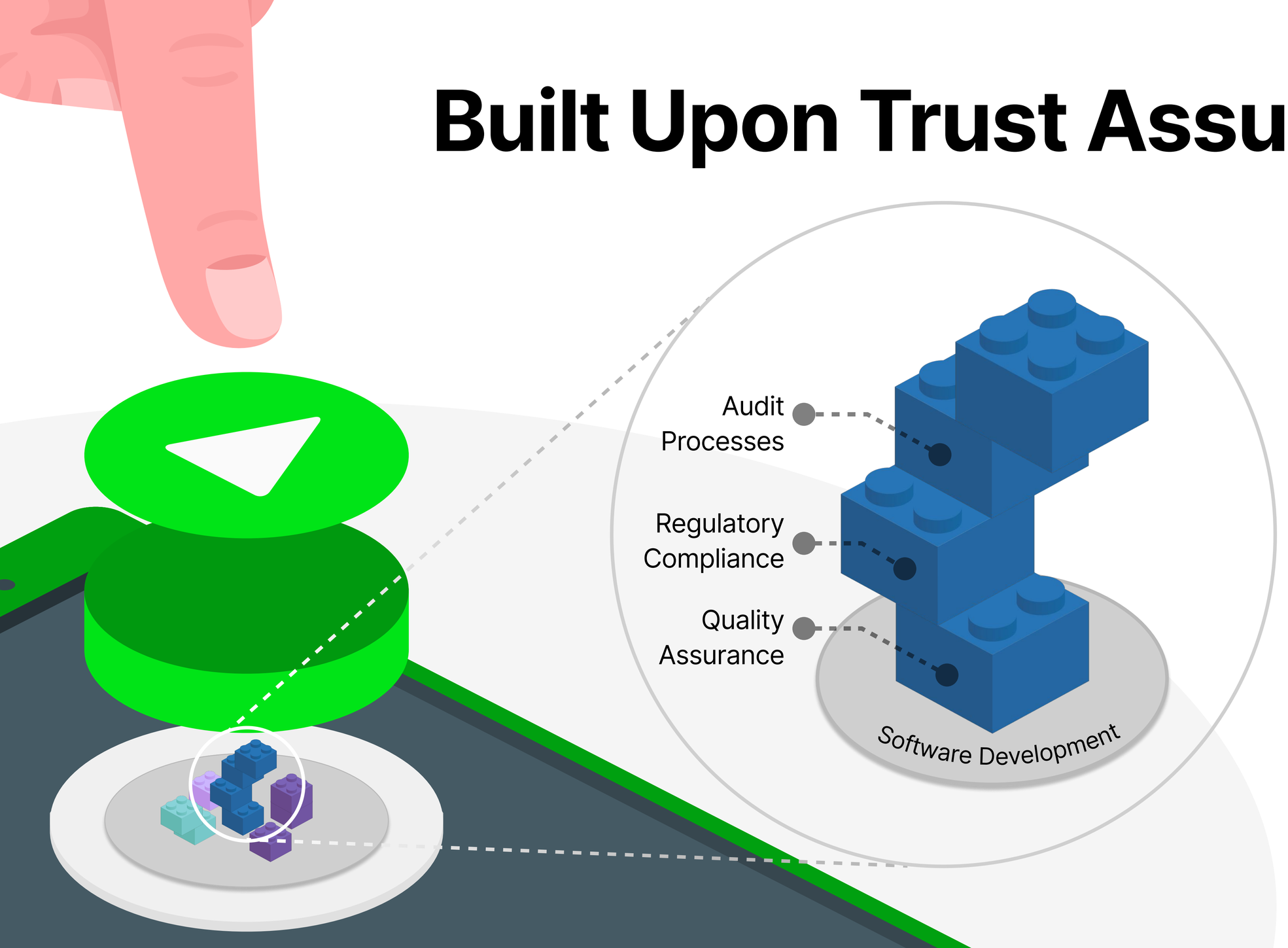
TECHNISCHE UNIVERSITÄT BERLIN

The Ubiquitous Trust Tap

- Current software ecosystems rely on **implicit trust models** that present **increasing risks**.
- **Automated trust** is a necessary paradigm to enforce **verifiable** development and operational **processes**.
- **TrustOps** offers an **evidence-based approach** to building and verifying **software trustworthiness**.
- **Trust-enhancing mechanisms** can be progressively **integrated** into the **software life cycle**.

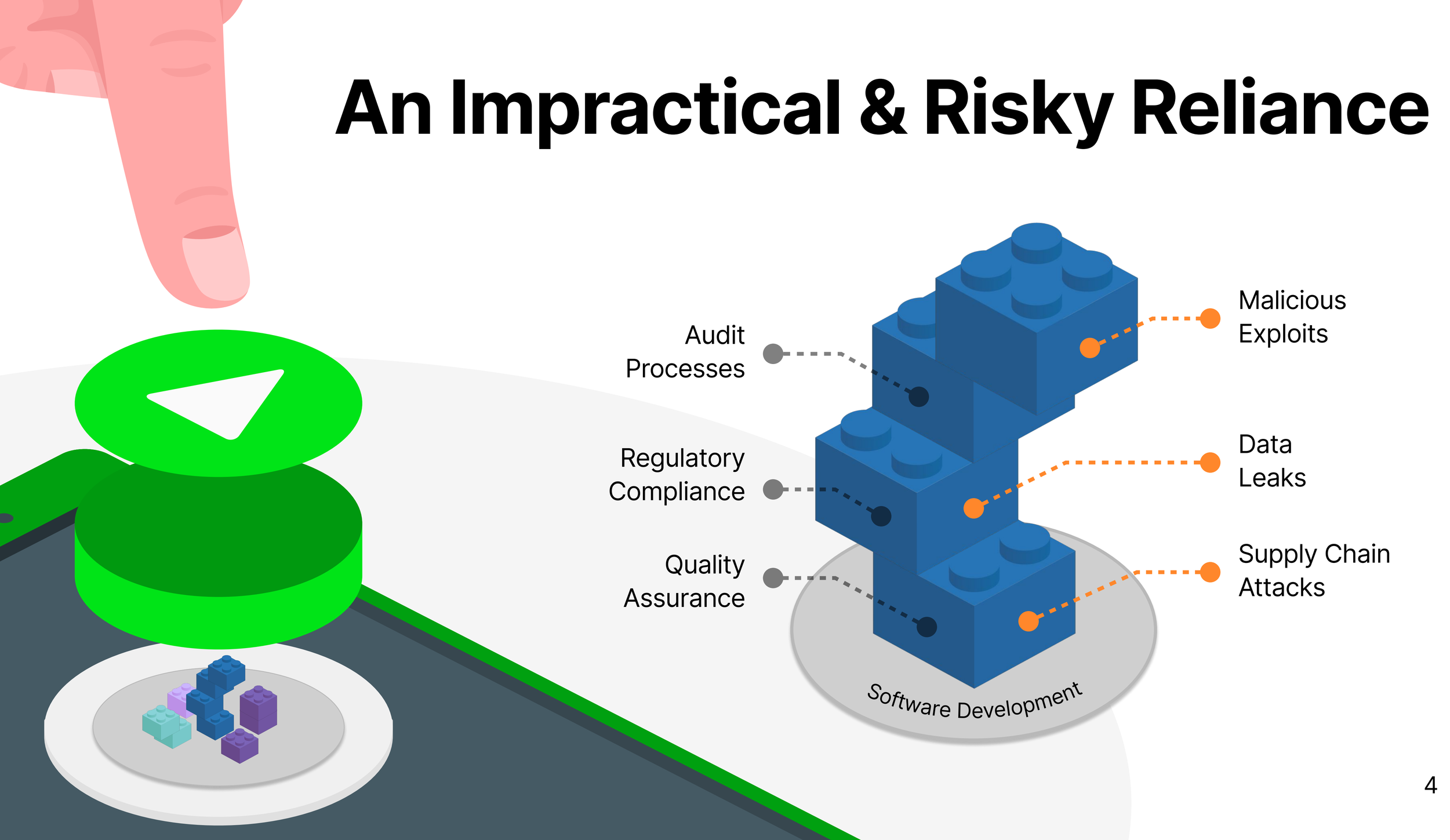


Built Upon Trust Assumptions

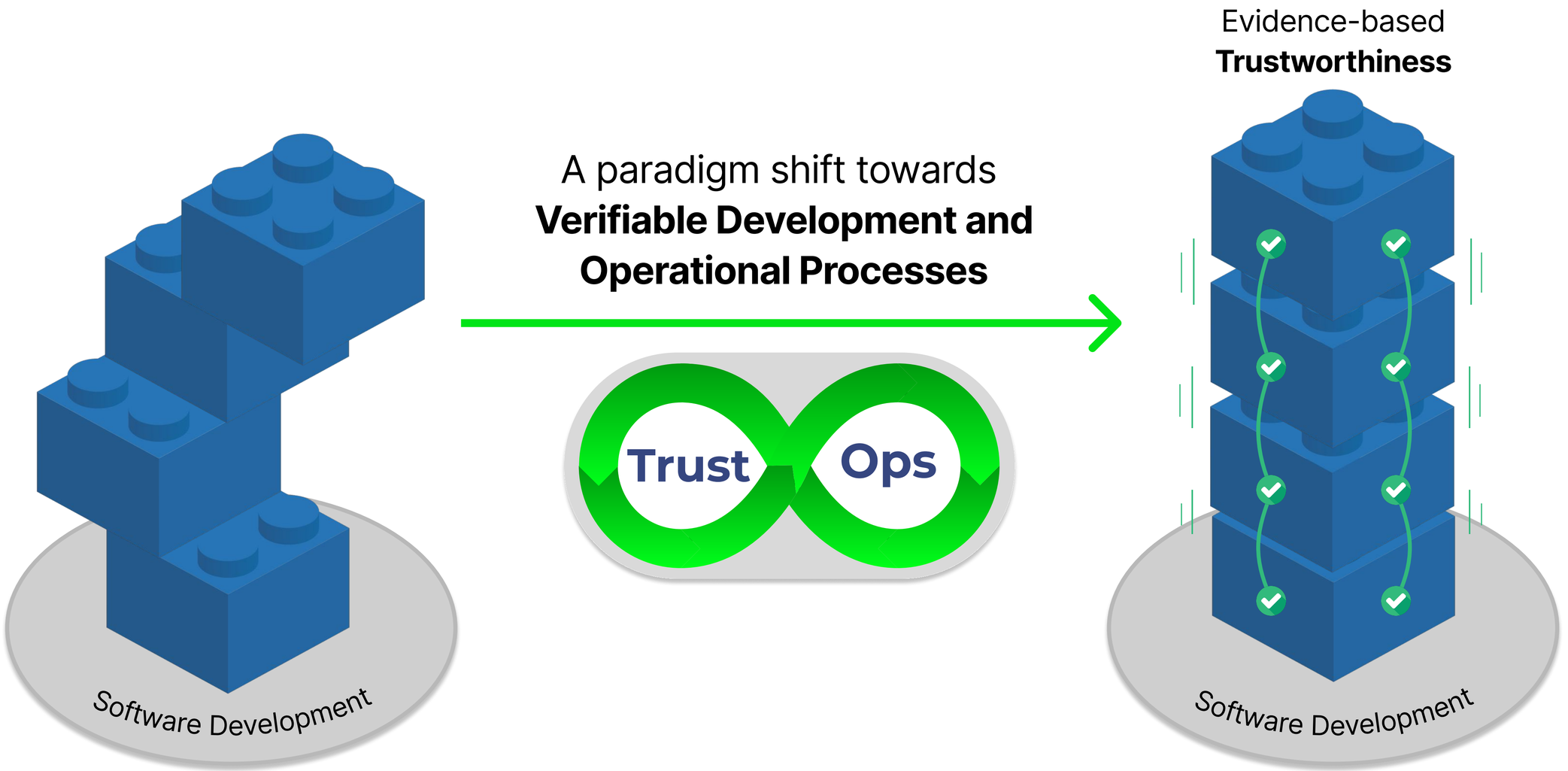


Mainstream methodologies such as **Continuous Compliance, DevOps, DevSecOps**, adherence to **Industry Standards**, and adoption of **Agile** processes, have been contributing to enhance **Perceived Trust.**

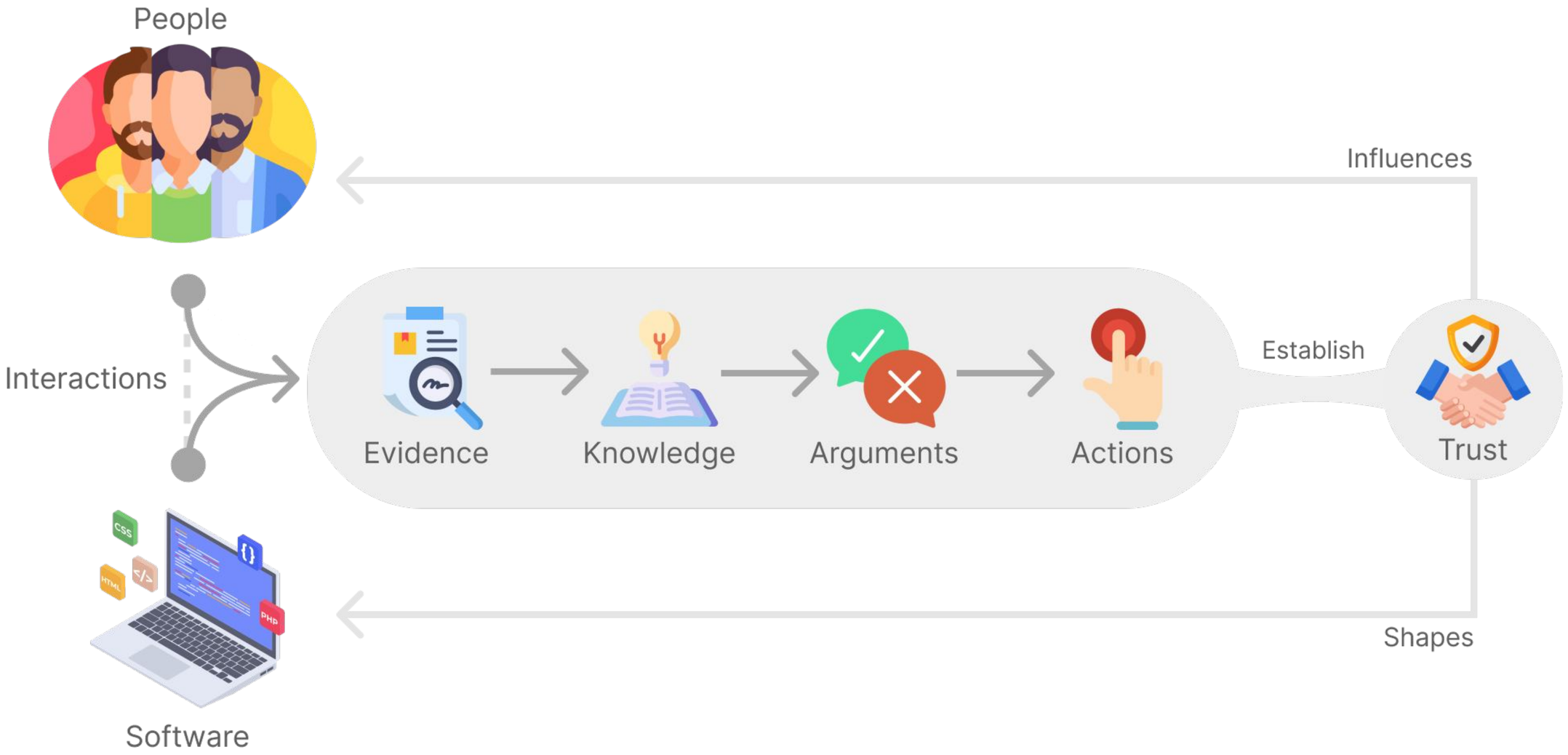
An Impractical & Risky Reliance



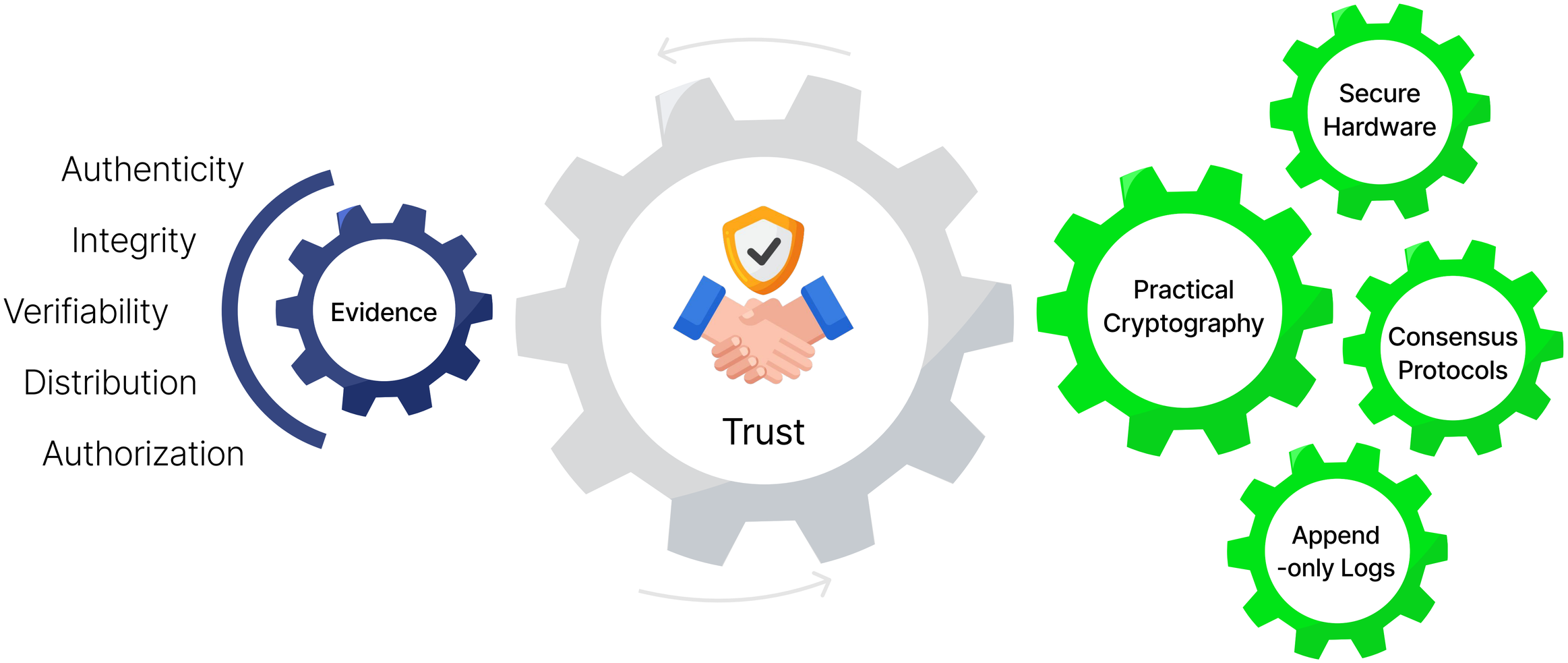
The Need for a New Trust Model



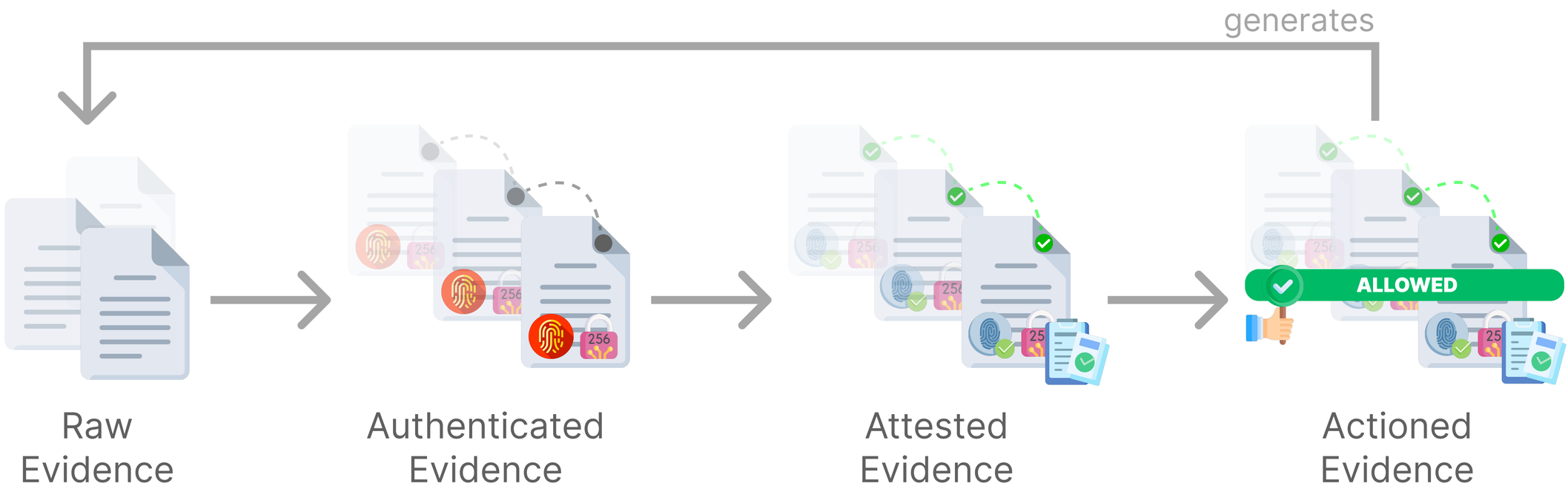
A Socio-Technical Phenomenon



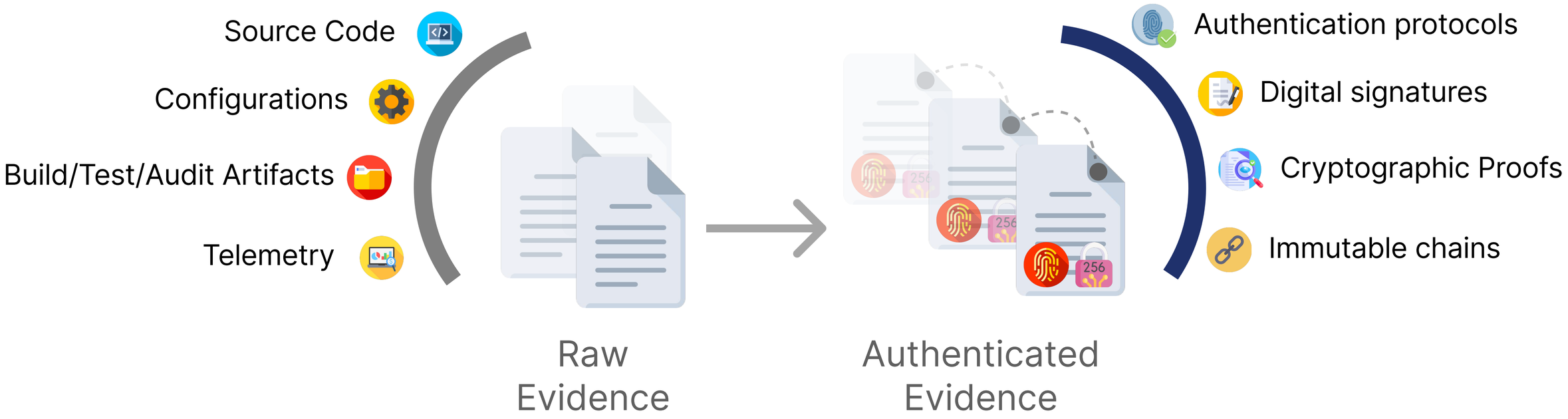
The Automated Trust Paradigm



The Life Cycle of Evidence

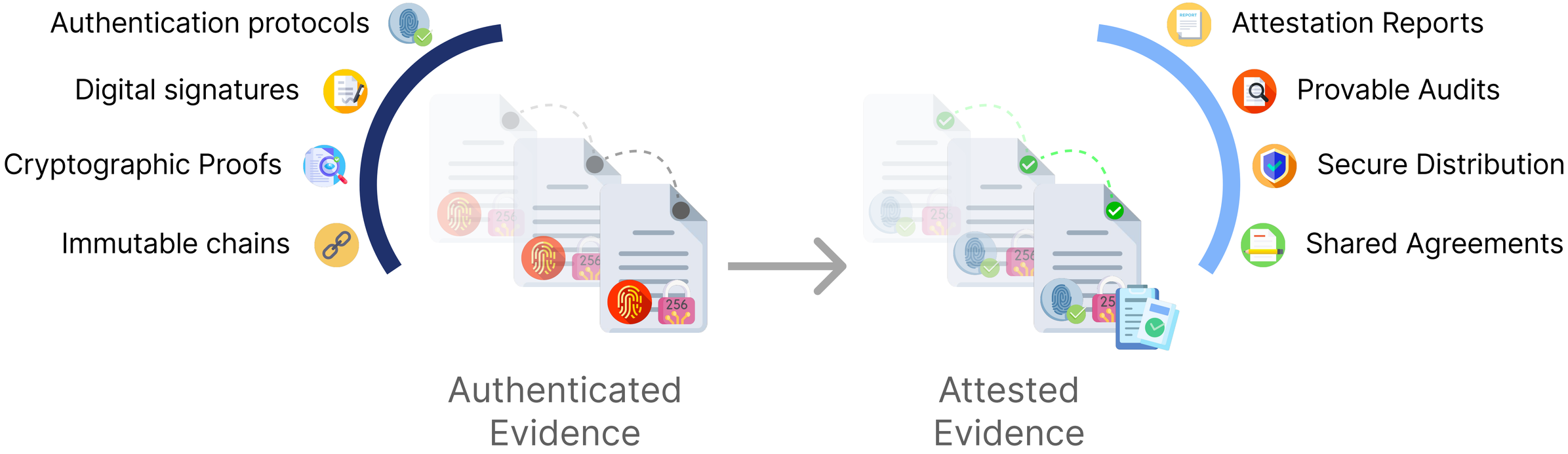


From Raw to Authenticated Evidence



Through an **identity-centric** approach, **enhancing evidence** with contextual metadata, and employing **cryptographic mechanisms** to ensure **authenticity and integrity** of all collected knowledge.

From Authenticated to Attested Evidence

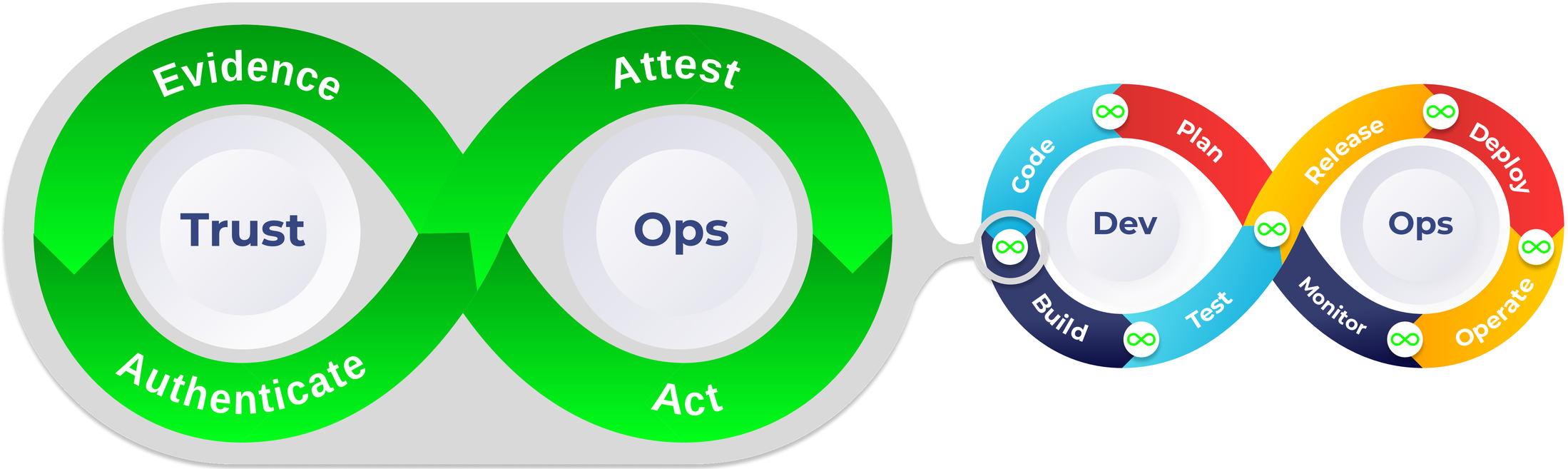


Through **verifiability and consensus** mechanisms, enabling **independent validation** of software properties and actions, leveraging **advanced technologies** like TEEs and ZKPs for **continuous, automated attestation**.

From Attested to Actioned Evidence

Through **automated actions** like authorization policy definition and access control enforcement, supporting **chainable and verifiable accountability, explainability, and observability** in software development and operations.

Augmenting the DevOps Life Cycle



Continuously Building Trustworthy Software

From Planning to Coding

Issues, Tickets, User Stories, Requirements, Risk Assessment, etc...

Augmenting project management and planning tools to support signed artifacts, immutable logs, and automated attestation.

Attested planning evidence and authorized execution of planned changes

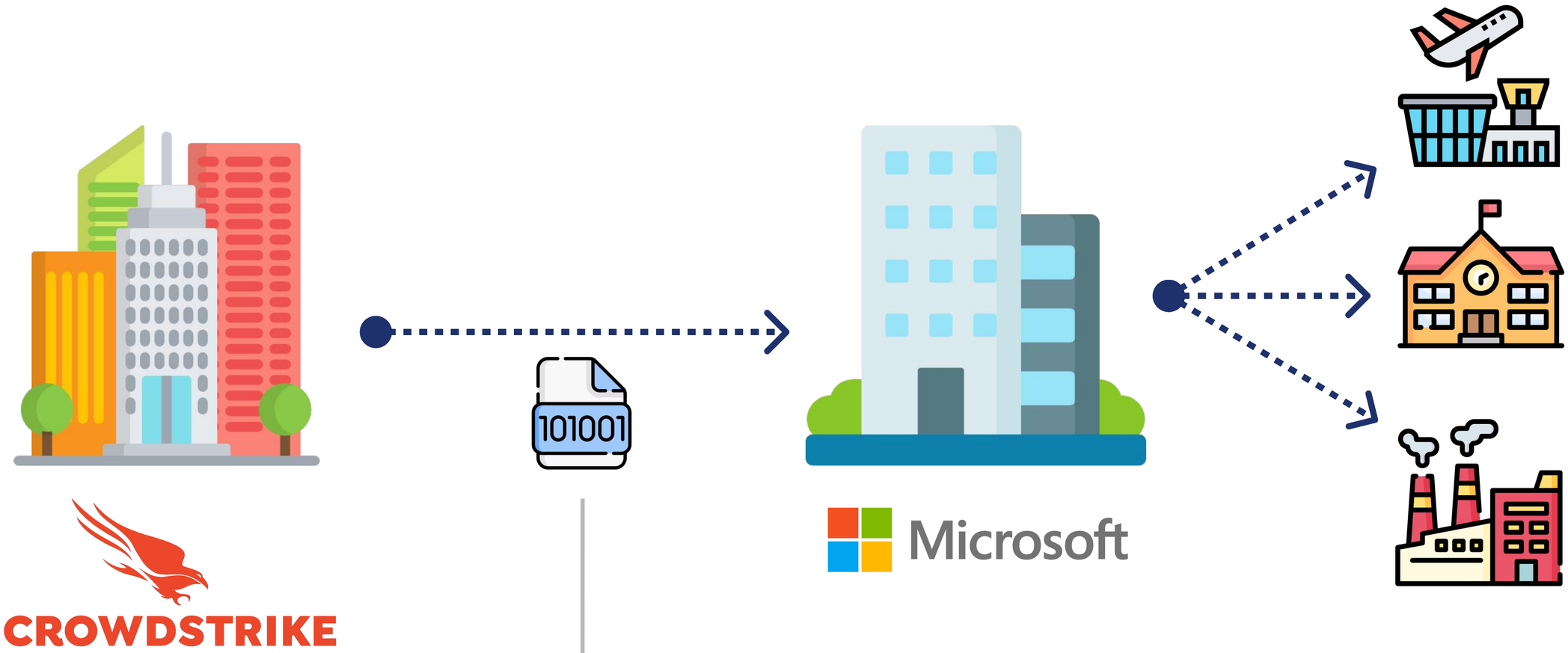
Code, Version Control, Dev Environment, Formatting, AI Assistant Configs, etc..

Augmenting development environments to support evidence provision, authentication, and automated attestation.

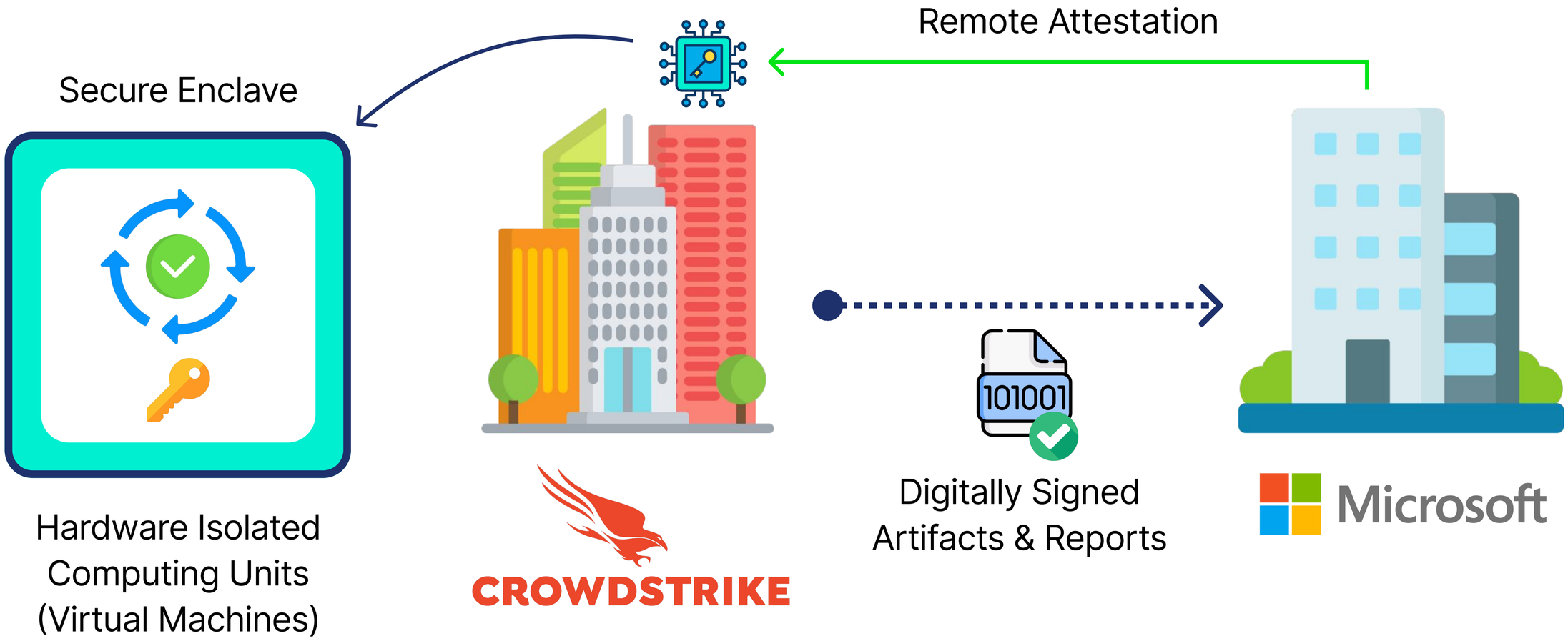
Succinct correctness proofs of development practices

Trustworthy Continuous Integration Pipelines ft. Trusted Execution Environments

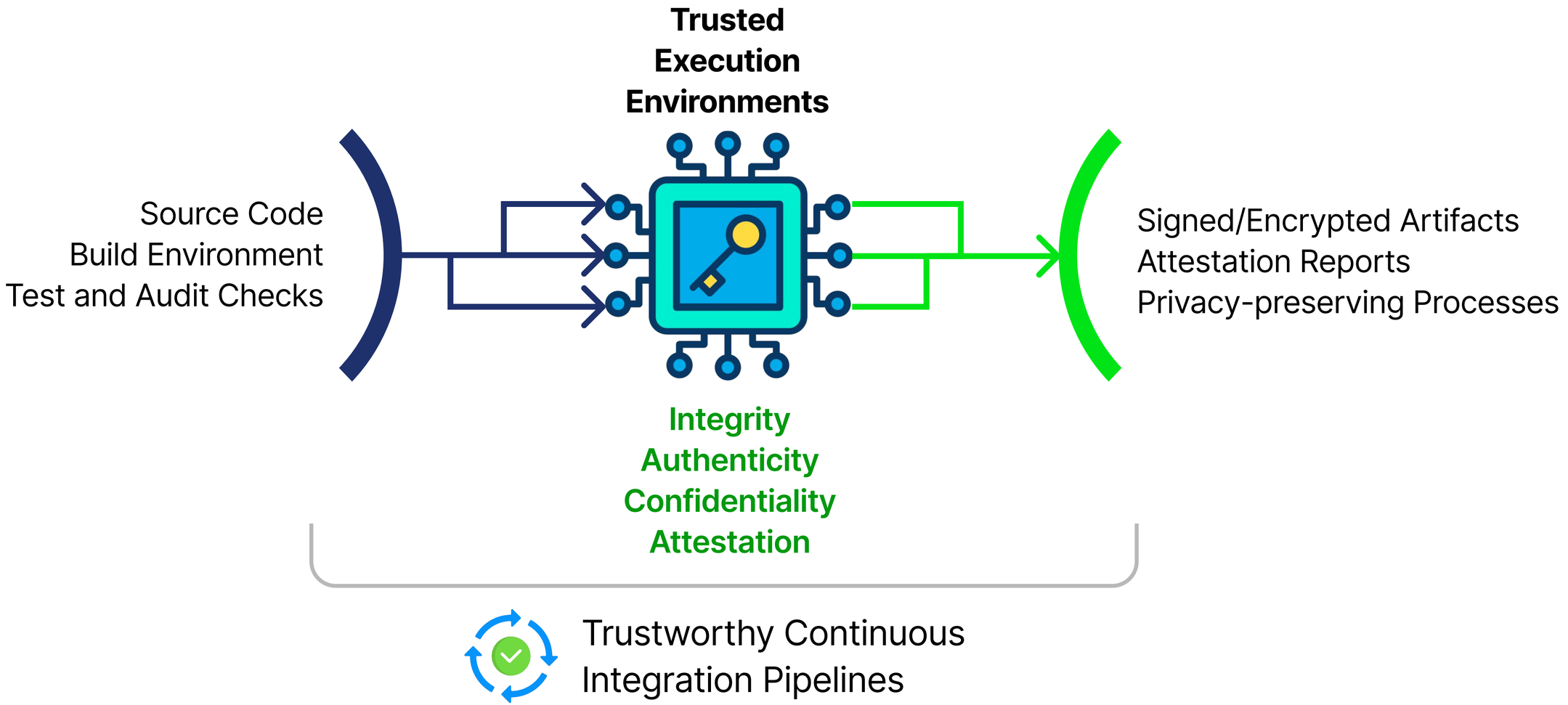
A Case for Supply Chain Security



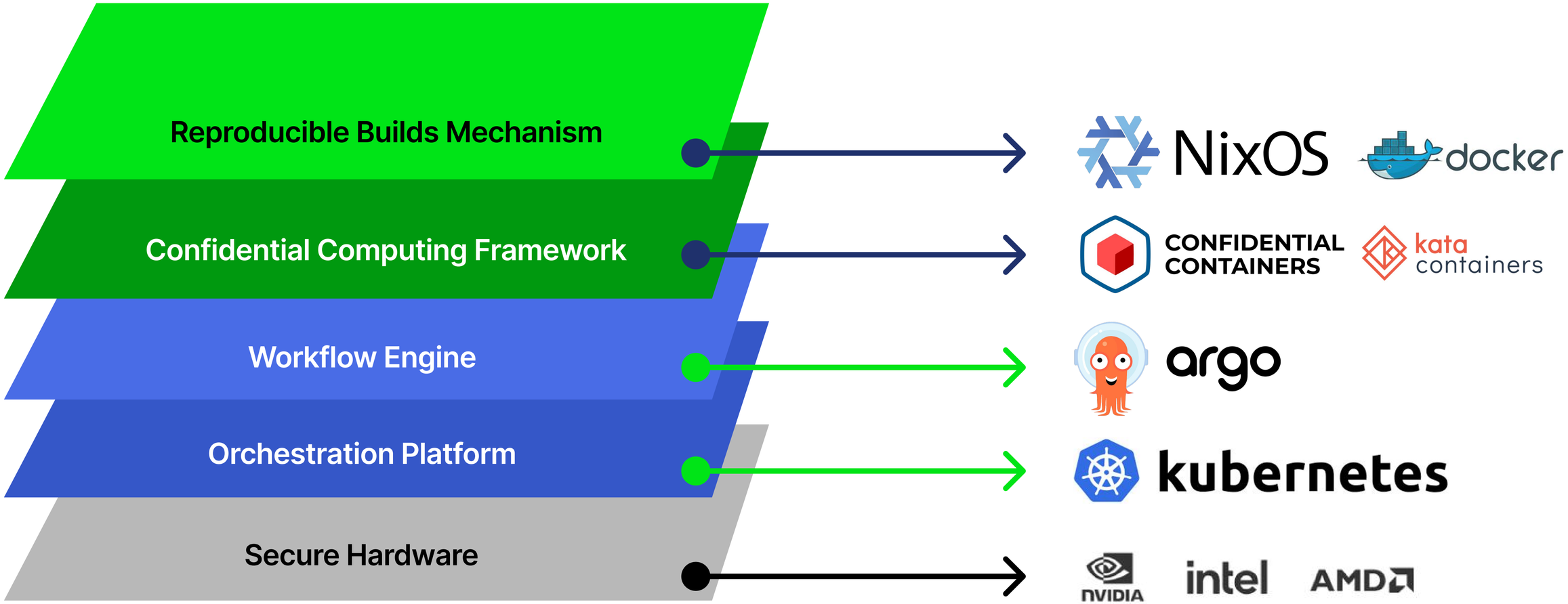
Achieving Inter-Organizational Trust



From Building to Testing

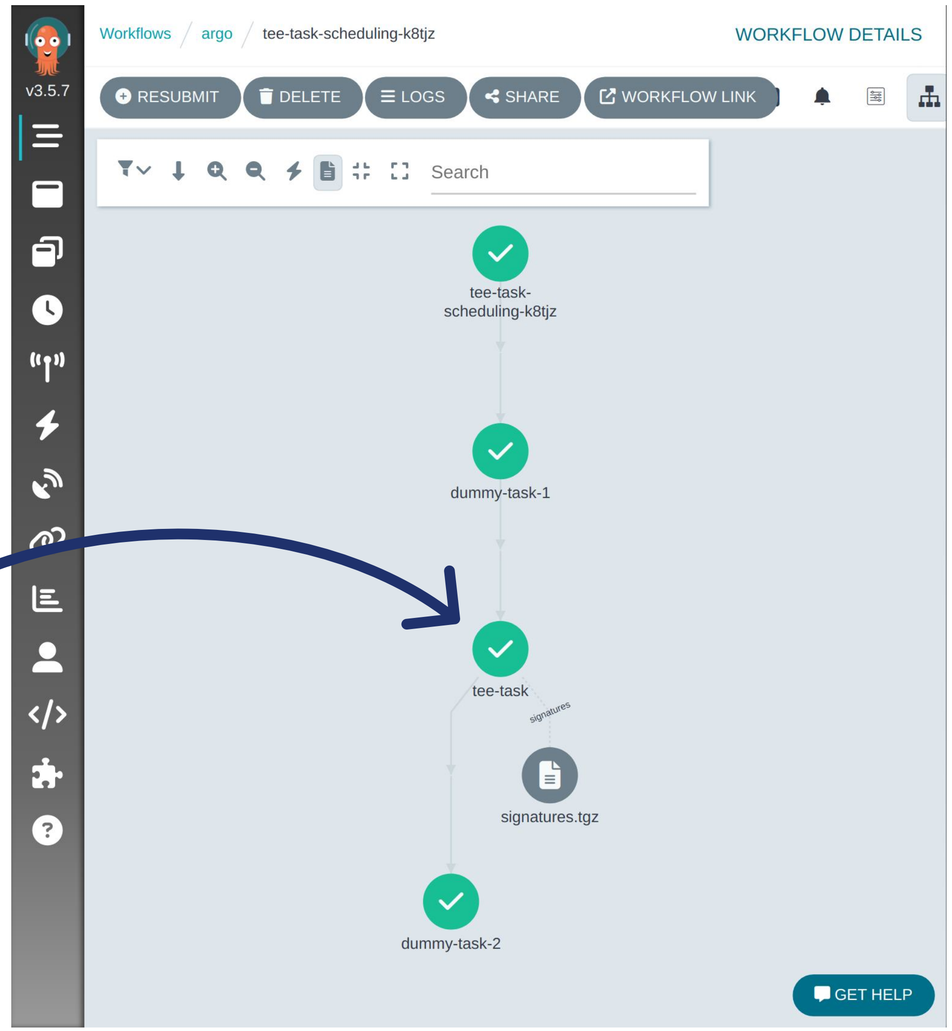


A TEE Orchestration Stack



Confidential Containers and Argo

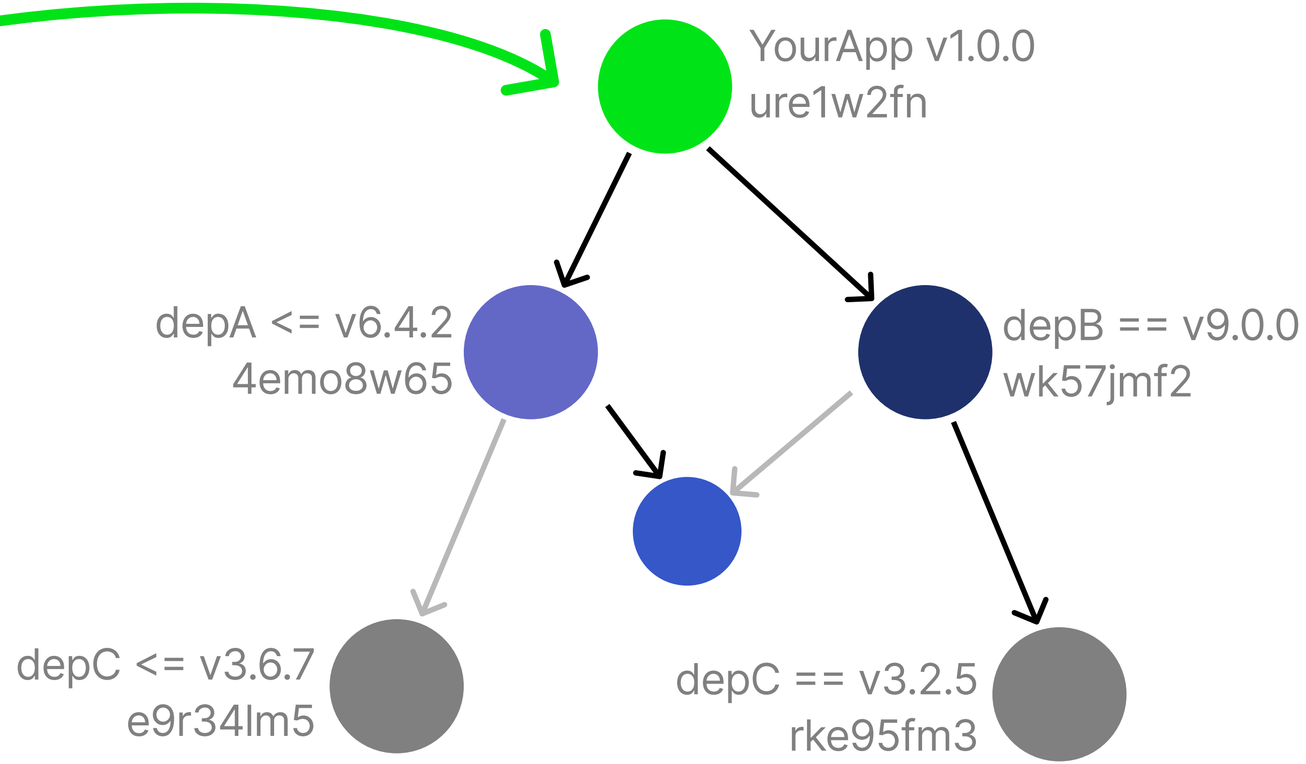
```yaml
apiVersion: argoproj.io/v1
kind: Workflow
...
  templates:
  - name: tee-workflow
    steps:
    - - name: dummy-task-1
        template: dummy-task-1
    - - name: tee-task
        template: tee-task
    - - name: dummy-task-2
        template: dummy-task-2
...
  - name: tee-task
    podSpecPatch: '{"runtimeClassName":"kata-qemu-tdx"}'
    container:
      image: nixos/nix:latest
      command: [sh, -c]
      args: [
        "nix-shell trustops.nix && zip signatures.zip
signature.*"
      ]
      volumeMounts:
      - name: workdir
        mountPath: /tmp/vol
    outputs:
      artifacts:
        - name: signatures
          path: /tmp/vol/{{workflow.name}}/signatures.zip
...
```



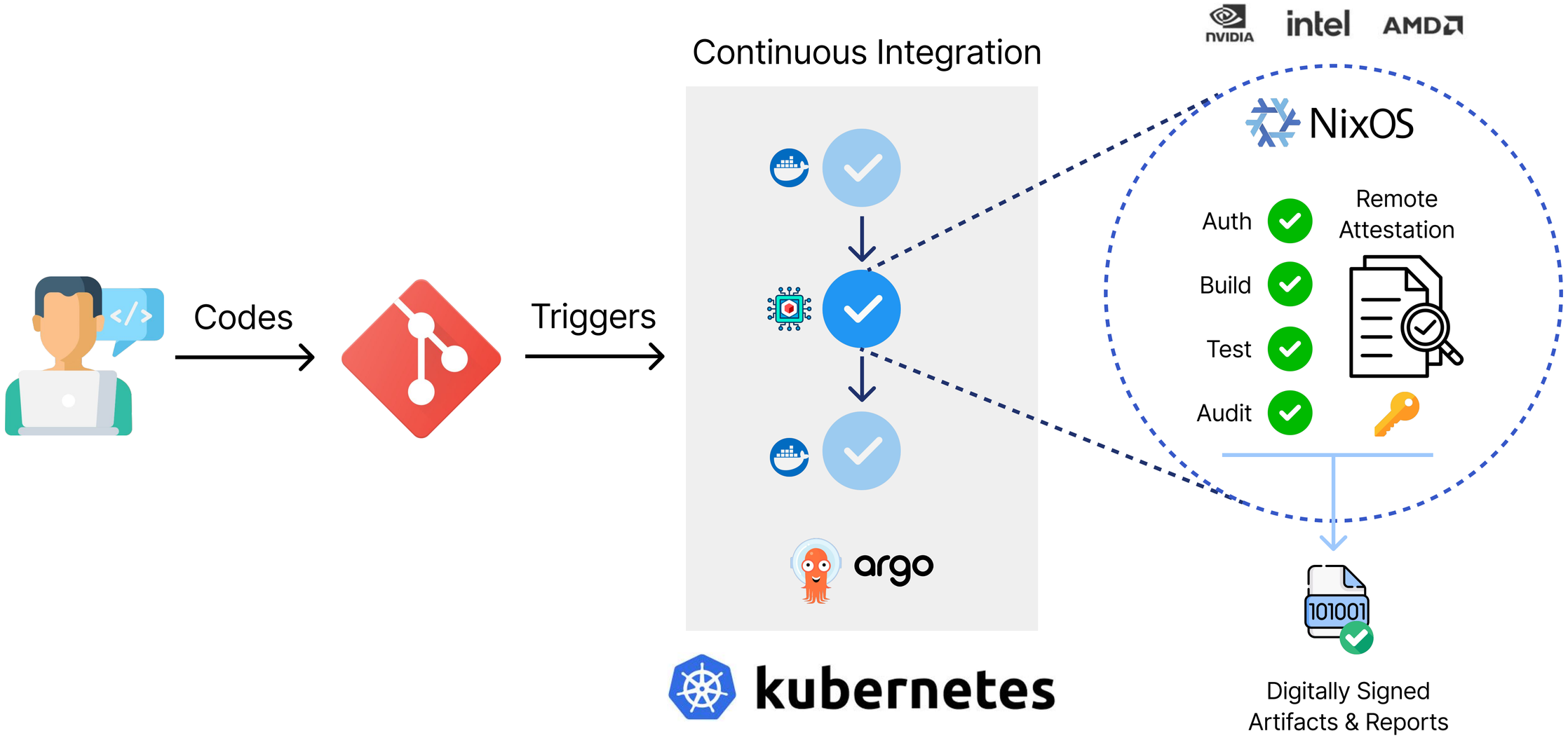
Nix-enabled Reproducible Builds

Declarative Package Management

Reproducible and Isolated Dependency Tree with Integrity Guarantees



Trustworthy CI Pipelines



From Releasing to Deploying

Release Artifacts,
Notes, Test Reports,
Audit Checks, etc...

Augmenting version control platforms and package managers to support signed artifacts, immutable logs, and automated attestation.

Attested release evidence
and authorized publishing of
verifiable artifacts

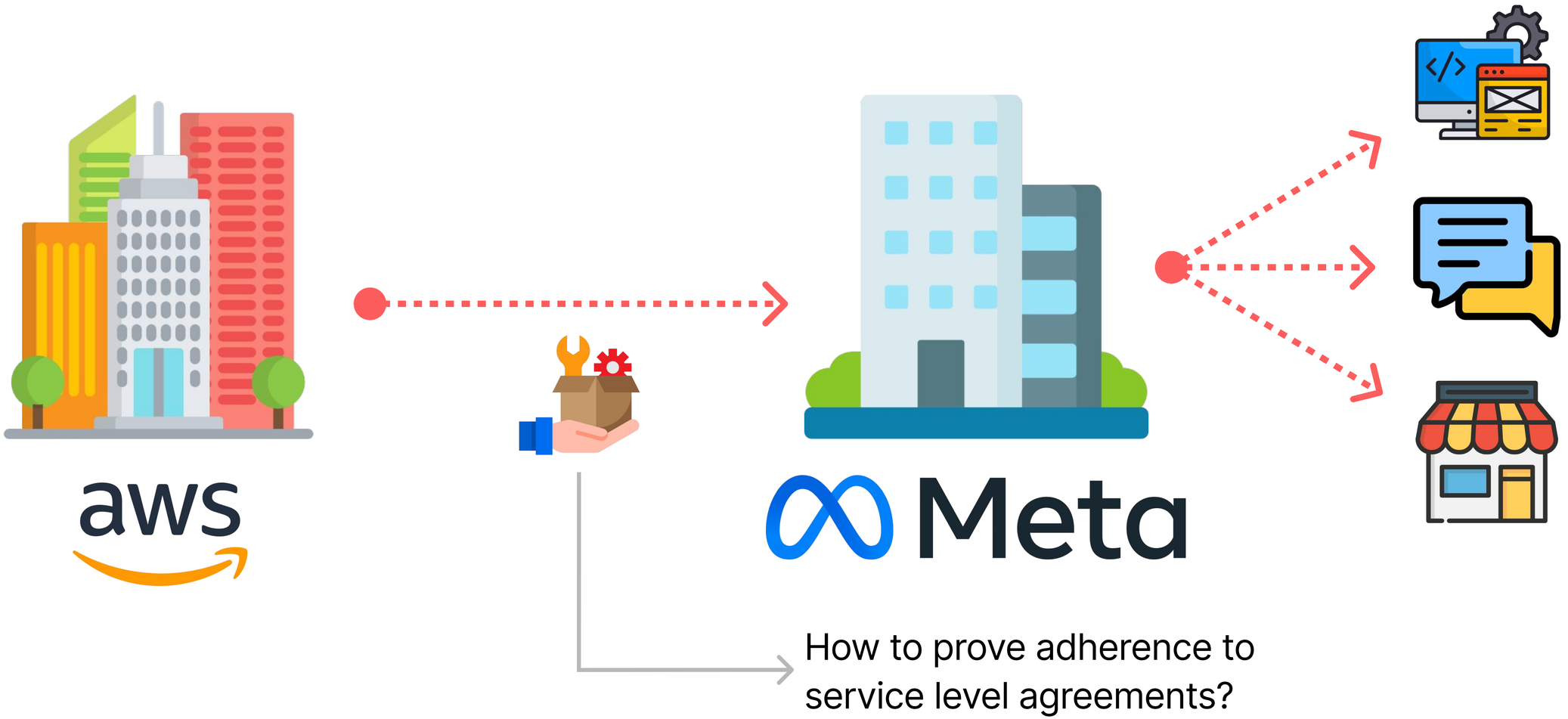
Deployment & Infrastructure
Configs, Rollback Plans,
Operational Permissions, etc...

Augmenting remote execution environments and orchestration tools to support deployment attestation.

Authorized deployment of
attested artifacts in verifiable
production environments

Verifiable SLA Monitoring Pipelines ft. Zero-Knowledge Proofs

Verifying Service Level Agreements



Achieving Inter-Organizational Trust

Prover

aws

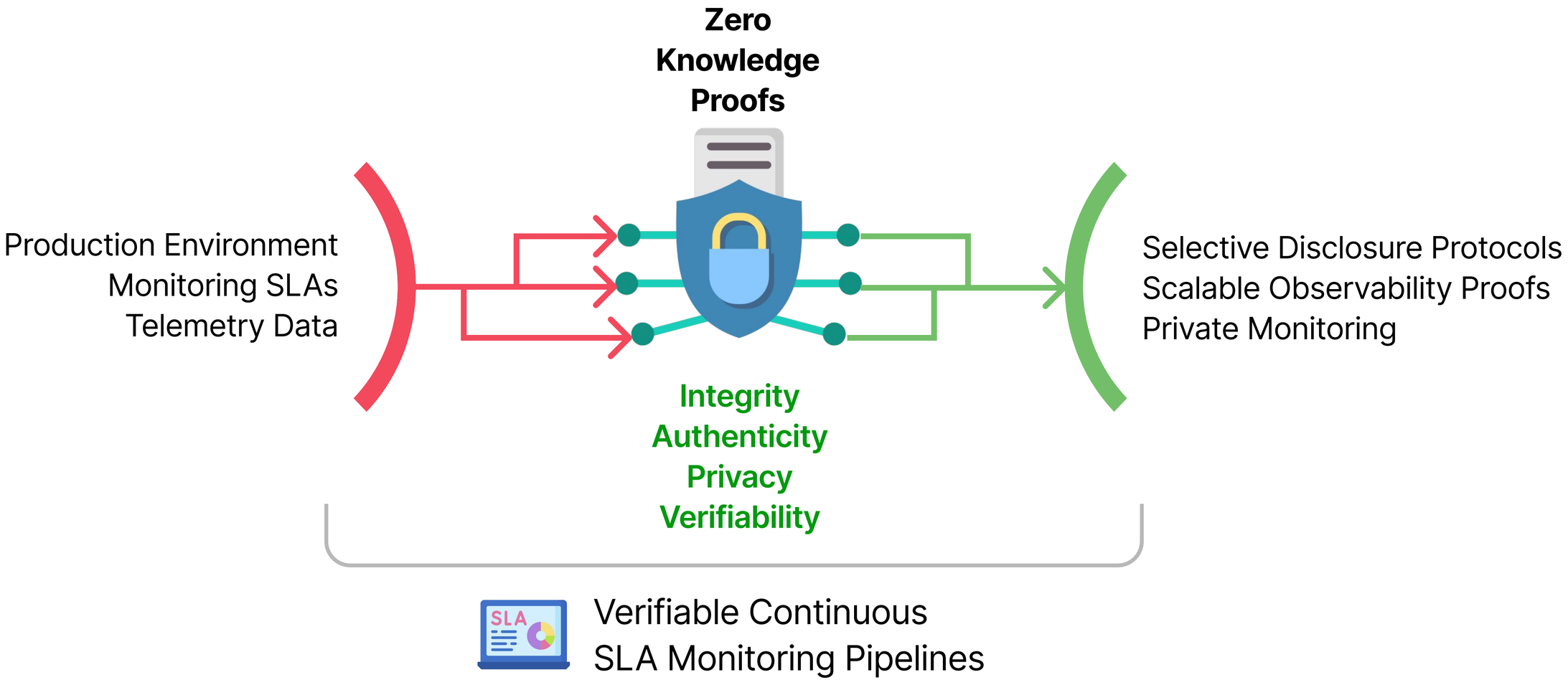
SLA: AWS API Gateway monthly response time should be no more than 100ms for at least 95% of the responses

Complete, Sound, and Privacy Preserving Cryptographic Proofs

Verifier

Meta

From Operating to Monitoring

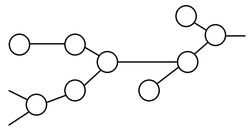


ZK Programmability

ZK

HDL

Languages for describing circuit synthesis



Pros: Clear constraints, Elegant syntax
Cons: Hard to learn, Limited abstraction

Circom

Library

Libraries for describing circuit synthesis

```
circ.add_wire(...)
circ.add_gate(...)
```

Pros: Clear constraints, As expressive as {Rust, C++}
Cons: Need to know {Rust, C++}, Few optimizations

Arkworks (Rust)
Gadgetlib (C++)
Bellman (Rust)
Snarky (OCaml)
PLONKish (Rust)
...

PL + Compiler

Languages, compiled to circuits

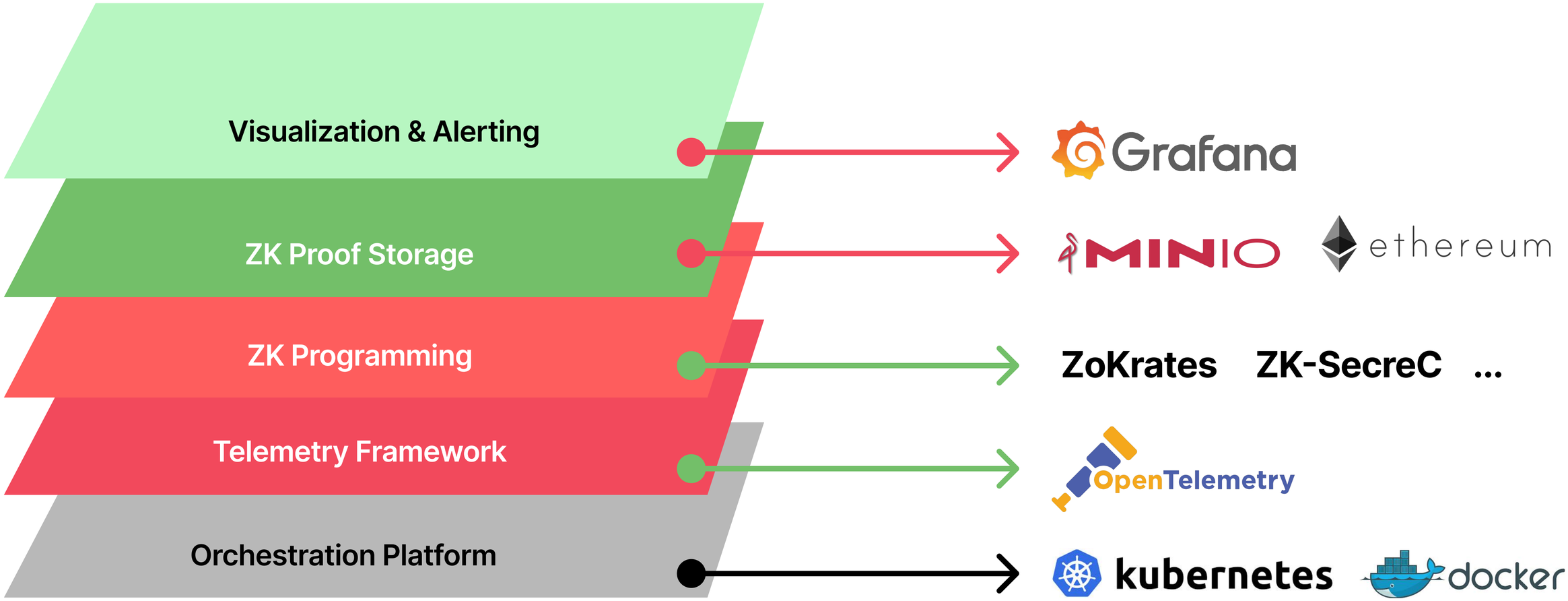
```
func main (...) {
...
assert_eq(...)
}
```

Pros: Easiest to learn, Elegant syntax
Cons: Limited witness computation, Prototype level, Niche use-cases, Custom IRs

ZoKrates
Noir
Leo
Cairo
...
ZK-SecreC

Meta Compilers, Frameworks, Platforms, zkVMs, Hardware Acceleration, ...

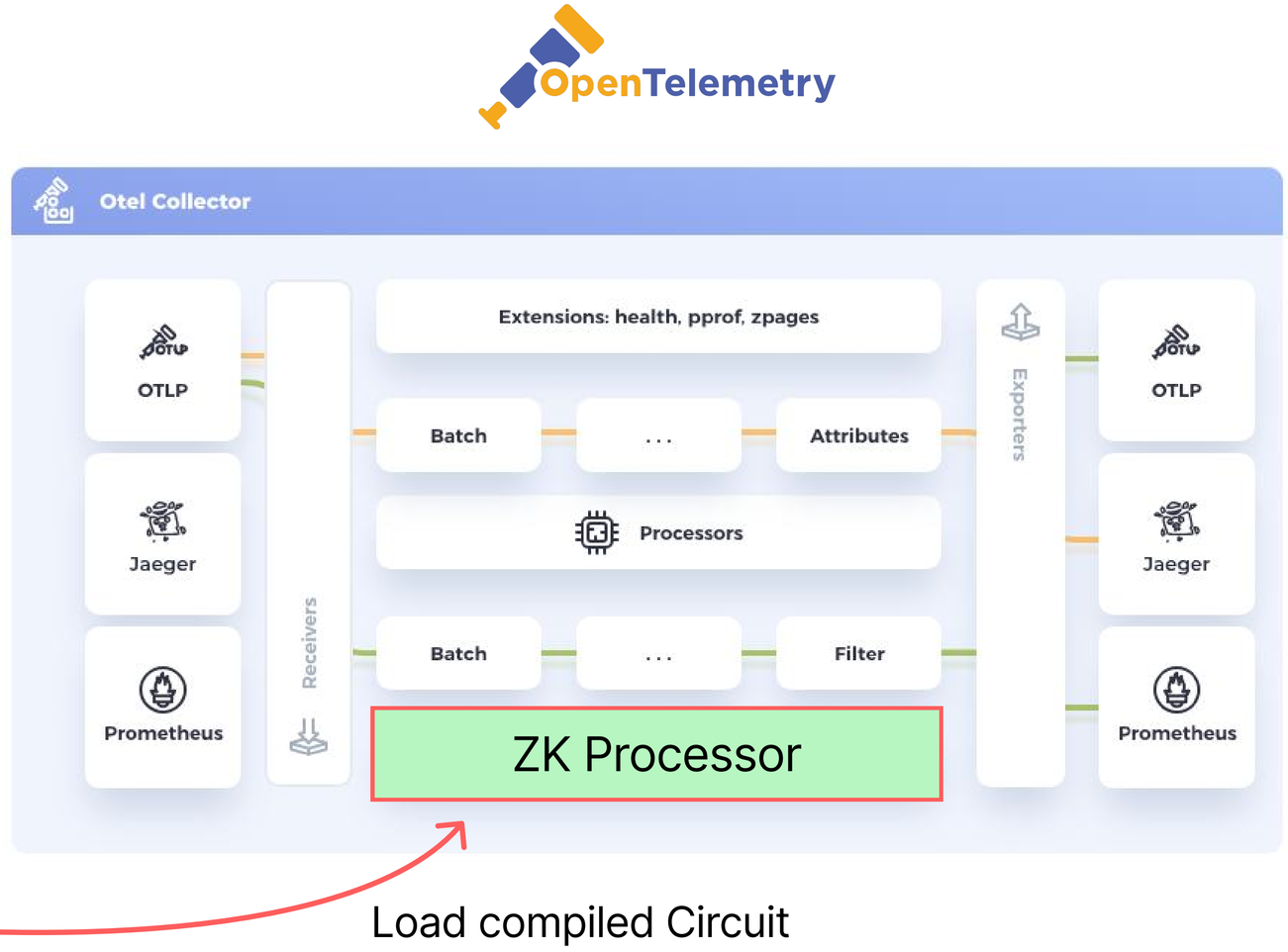
A ZK Monitoring Stack



A ZK OpenTelemetry Processor

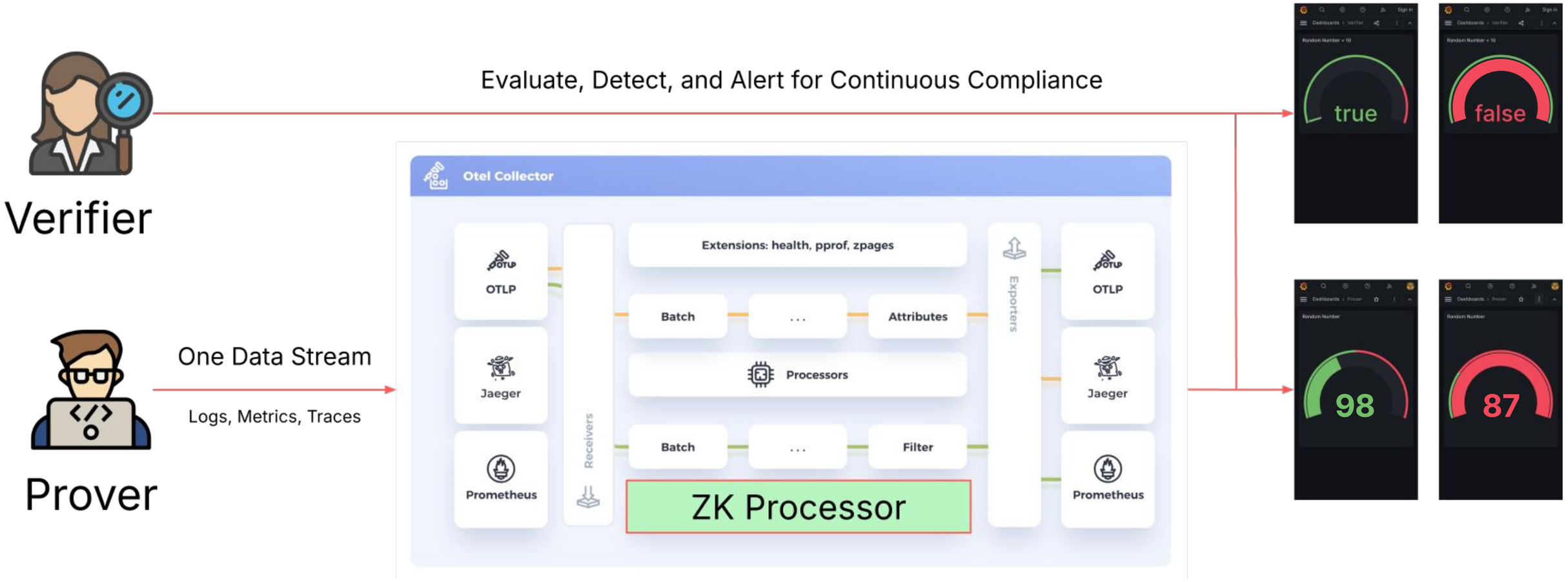
ZK-SecreC

```
fn f[N : Nat](
  BucketCounts : list[uint[N] $post @prover],
  Histogram : list[uint[N] $pre @public],
  ref sizeasserter : SizeAsserter[N, $post, @prover])

where Field[N] {

  let mut s_all = 0;
  let mut s_good = 0;

  for i in 0 .. length(BucketCounts) {
    s_all = s_all + BucketCounts[i];
    if (Histogram[i] <= 100) {
      s_good = s_good + BucketCounts[i]
    }
  };

  assert_ge(100 * s_good, 95 * s_all, ref sizeasserter);
}
```

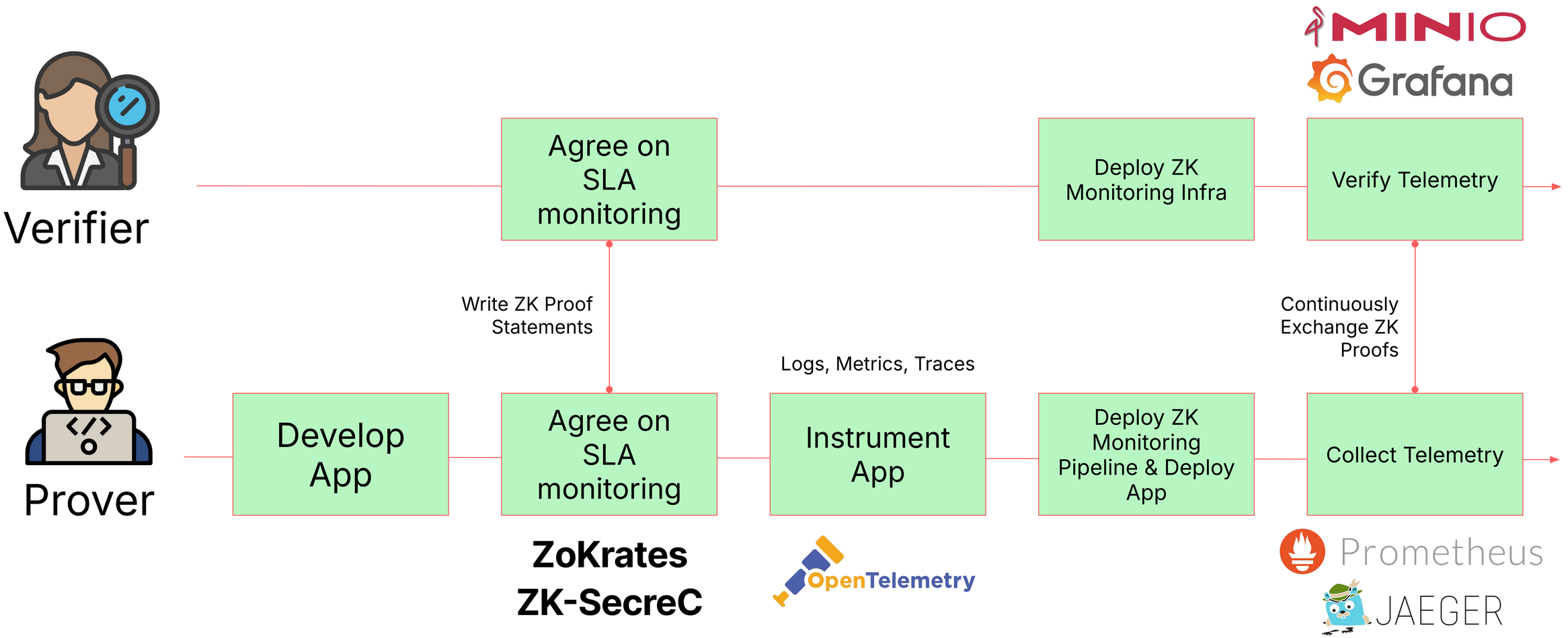


Load compiled Circuit

A ZK OpenTelemetry Processor

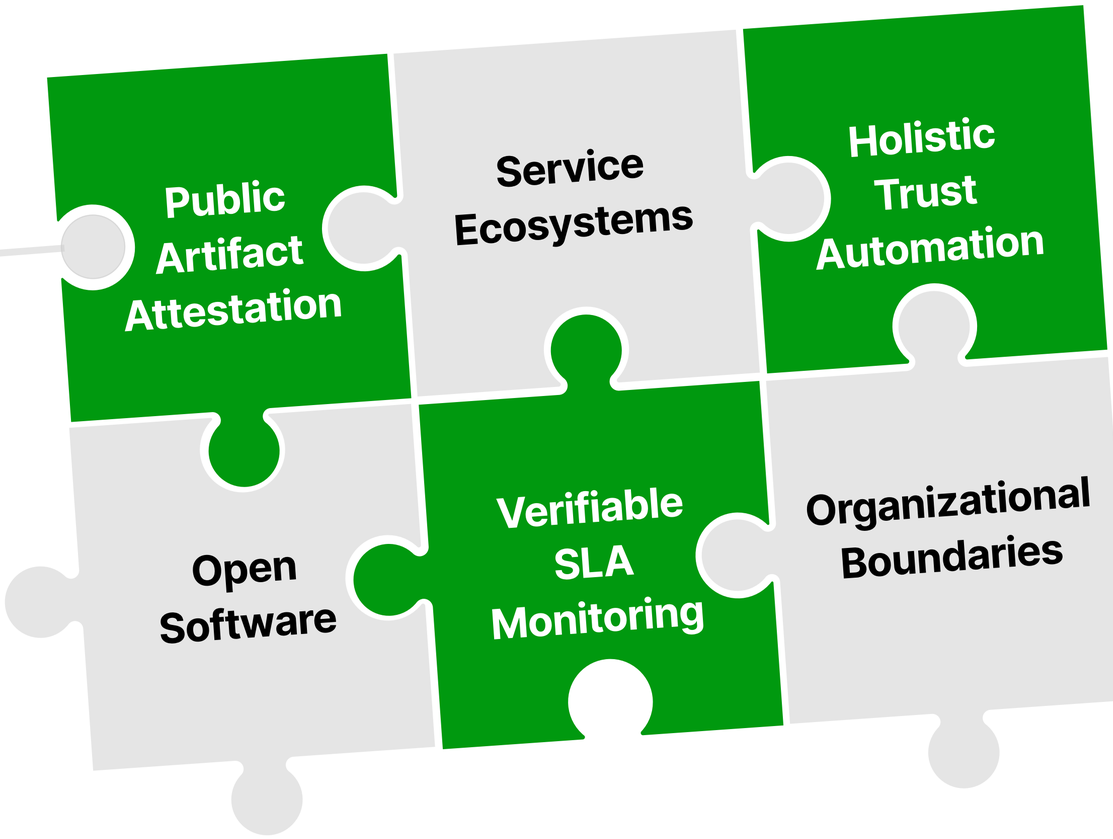


Verifiable SLA Monitoring Pipelines

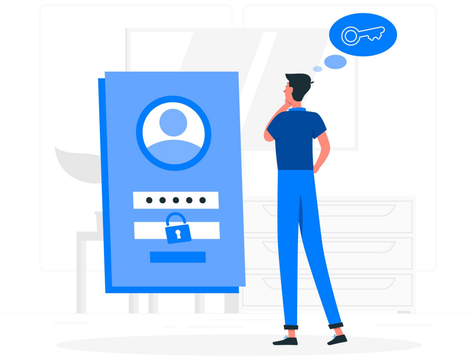


Aligning the Software Supply Chain

From Developers



To Users



Recent News and Advancements

Nix ecosystem supply chain security project

Nixpkgs, 2023

Reinforcing the security framework of Nix open source build and configuration management system across the full software development life cycle.

Reproducible builds as requirement

Reproducible Builds, 2023-2024

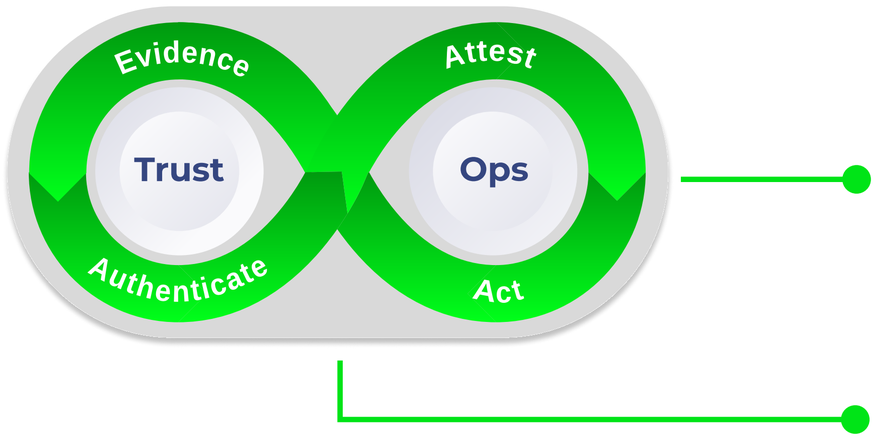
Ensuring the integrity of open source software throughout its lifecycle, from source to binary code, safeguarding open source software supply chains.

Introducing Artifact Attestations

GitHub, 2024

Providing a tamper-proof link between software artifacts and their source code, powered by the open-source Sigstore project, enhancing software supply chain security.

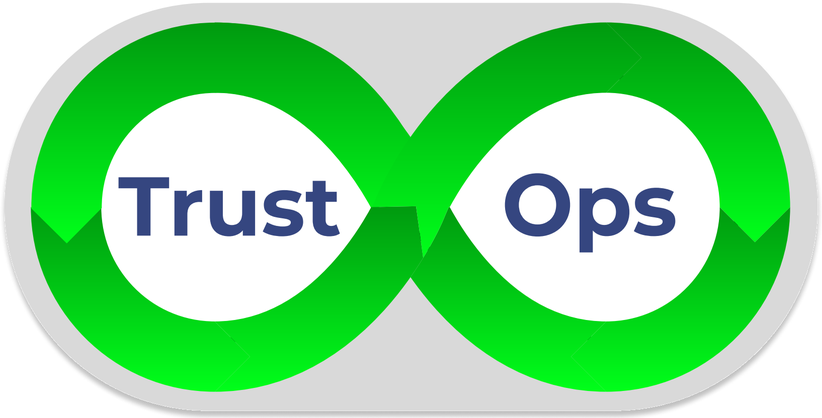
Research and Adoption Challenges



Evidence Management: Handling and storing TrustOps evidence is complex due to privacy concerns, varied evidence types, and potential interoperability issues.

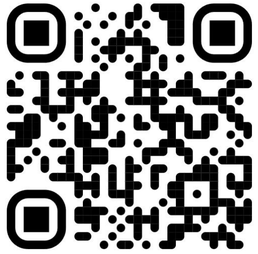
Usability: Improving the usability of technologies like ZKPs and TEEs is essential for TrustOps, alongside thorough threat modeling and user education.

Integration: Successful TrustOps adoption requires coexistence with non-adhering applications and creation of tools and guidelines across the software life cycle.



Eduardo Brito, Fernando Castillo, Pille Pullonen-Raudvere, Sebastian Werner

Continuously Building Trustworthy Software



eduardo.brito@cyber.ee

github.com/trustops

TEADAL

CYBERNETICA

TECHNISCHE UNIVERSITÄT BERLIN

Vectors from Flaticon.com and Freepik.com