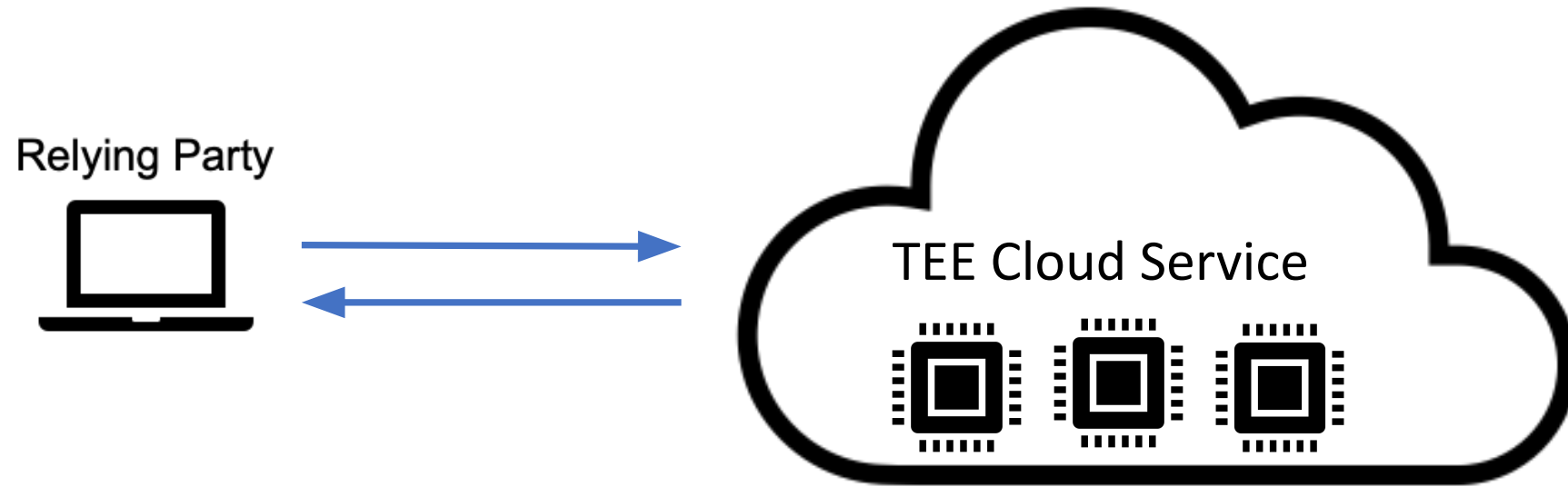


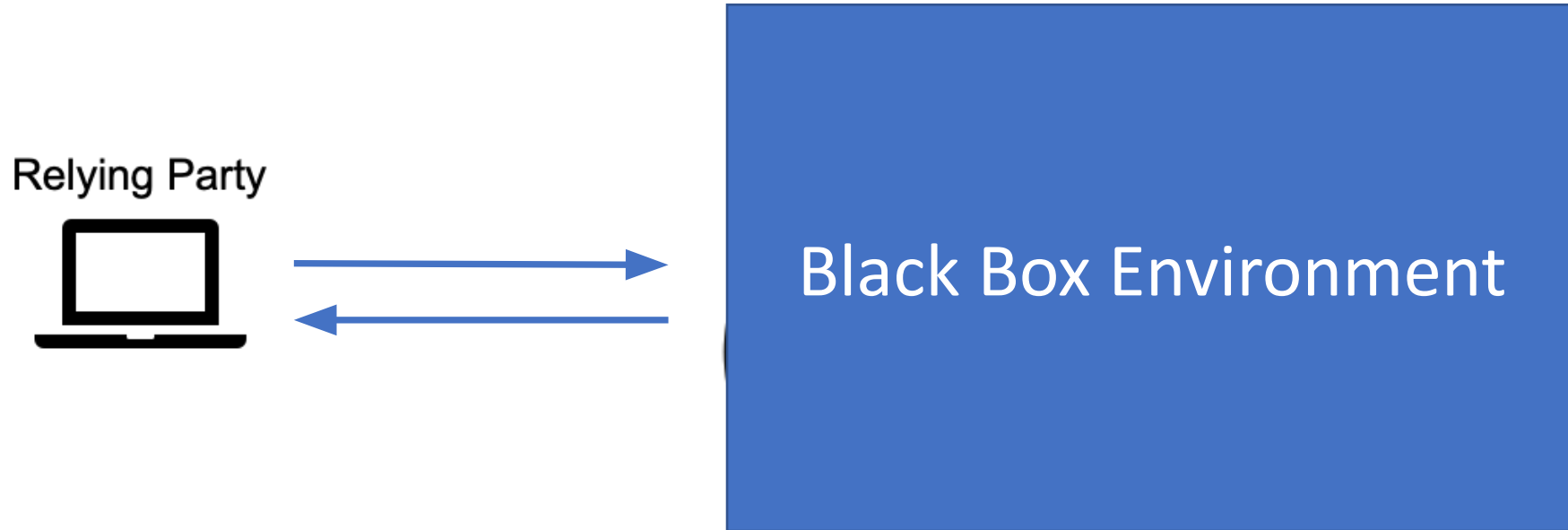
Trustless Attestation Verification in Distributed Confidential Computing

Donghang Lu
Research Scientist @TikTok

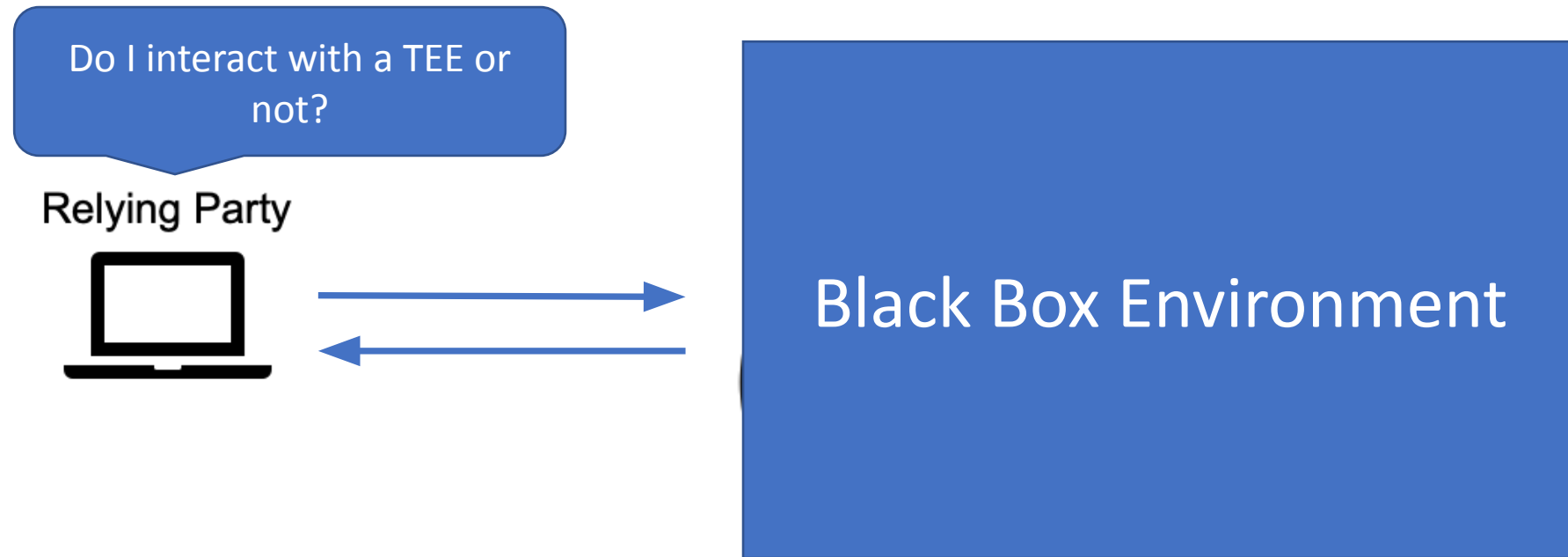
Our Scenario: TEE in a cloud service



Relying Party has no visibility inside the cloud



TEE Cluster needs to prove that the relying party is interacting with a legit and secure TEE



Attestation enables TEEs to self-prove

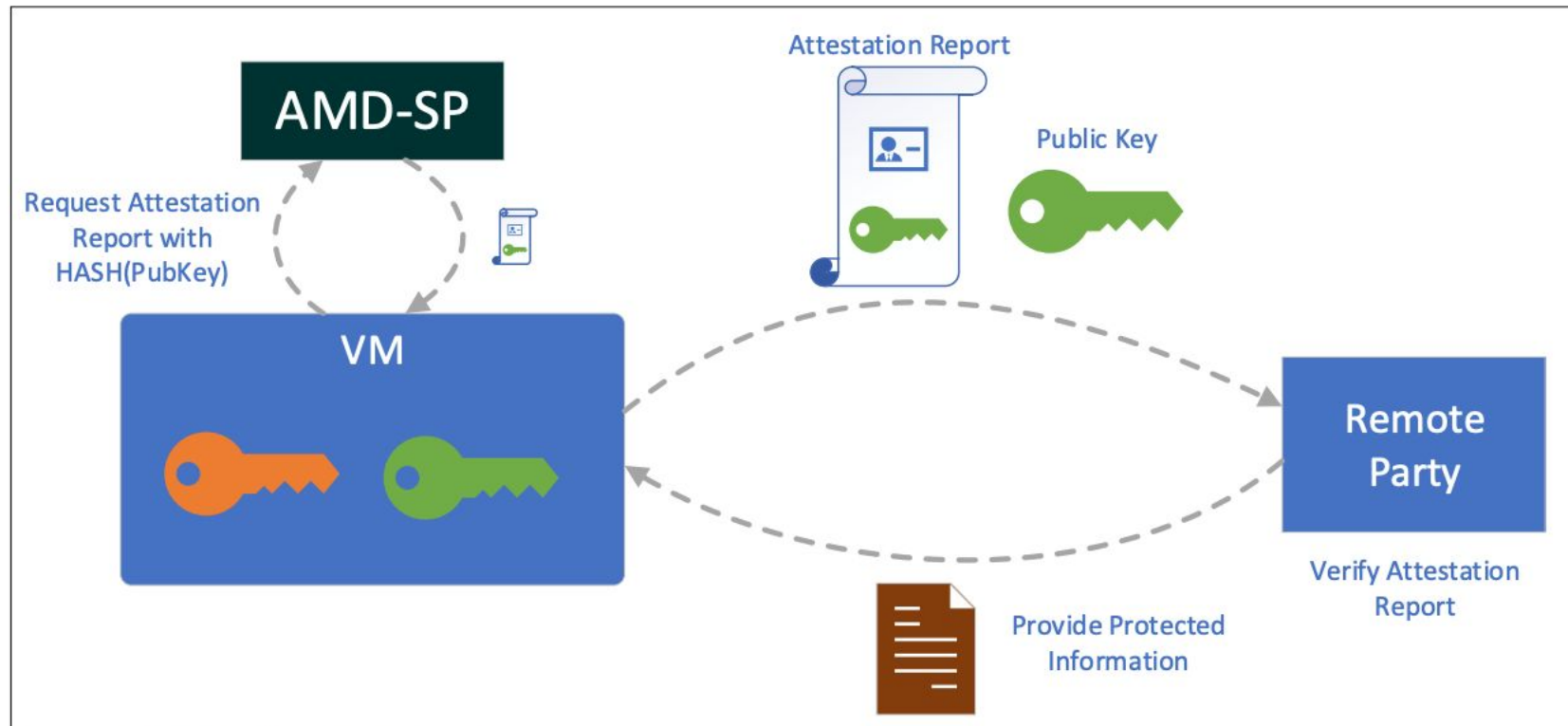
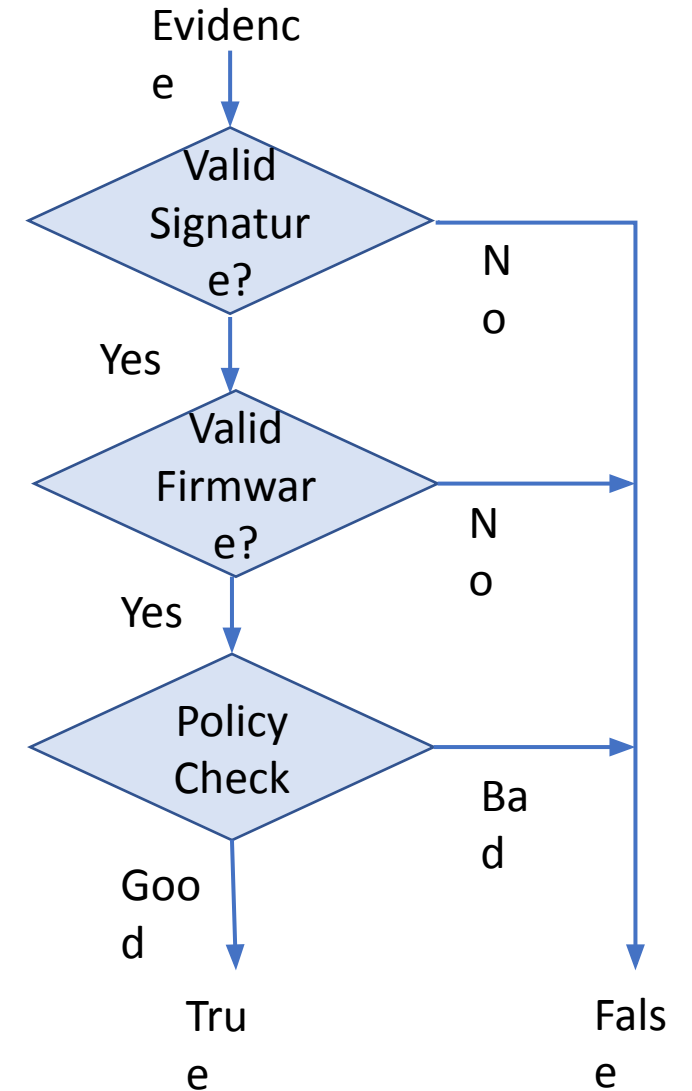


FIGURE 10: SEV-SNP ATTESTATION

Figure Source: AMD SEV-SNP: Strengthening VM Isolation with Integrity Protection and Mo

Attestation enables TEEs to prove their identity

- Attestation Procedure
 - Whether the certificate of TEE is signed by legit hardware vendor?
 - Whether the report is properly signed?
 - The firmware version / Microcode version
 - Guest image/ Software stack
 - ...



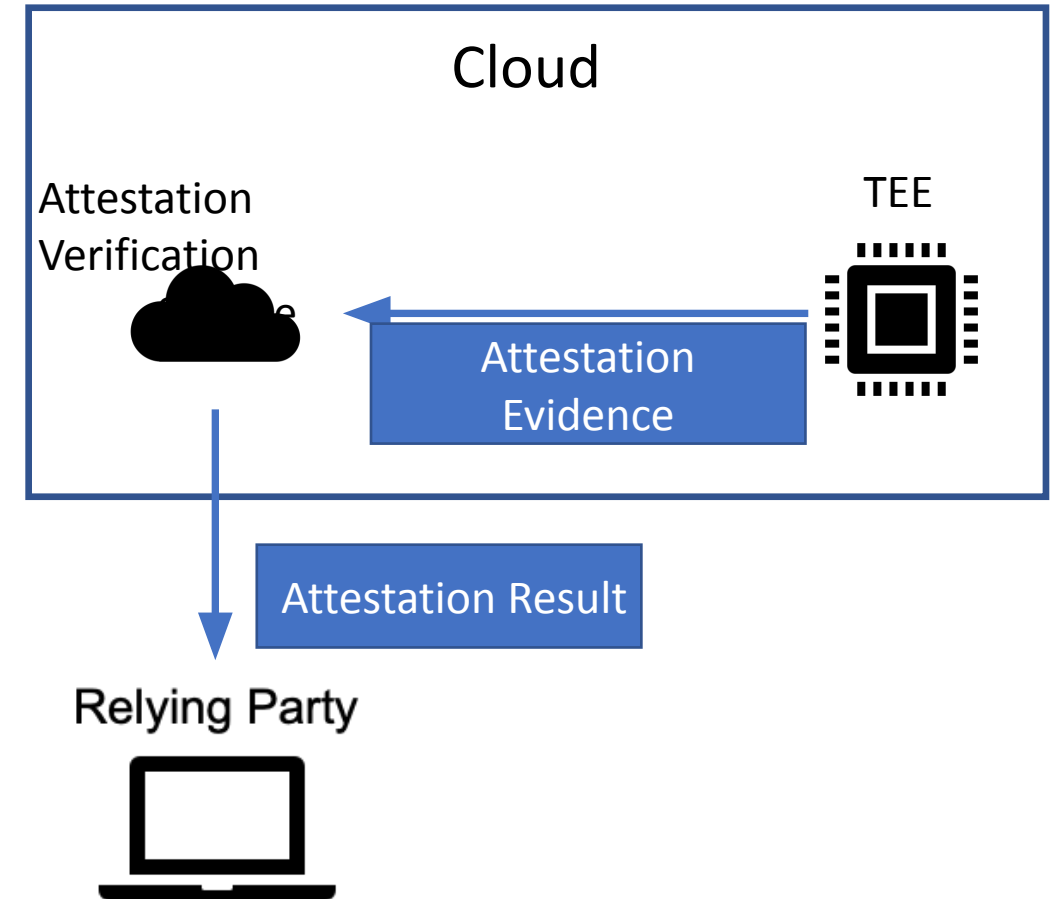
Attestation in Cloud Setting

- Option 1: Relying parties can verify the attestation reports by themselves
 - Relying parties may have to track latest reference values.
 - Relying parties may have to do additional computation.
 - The trust boundary remains the same.



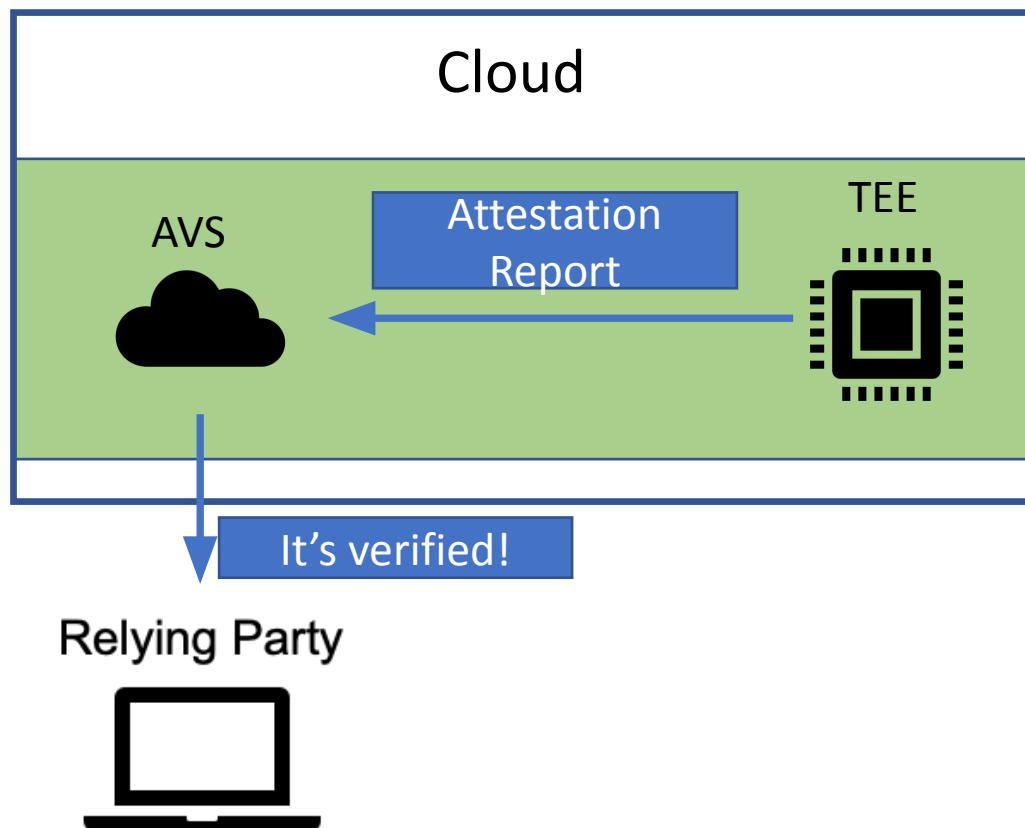
Attestation in Cloud: Attestation Verification Service (AVS)

- The cloud Service provides remote attestation verification service (AVS)
 - Easier for the cloud provider to manage reference values.
 - The remote attestation service can be easier to integrate with other cloud functionalities (authentication, issuing tokens, etc.)
 - However, the attestation verification service has to be trusted!

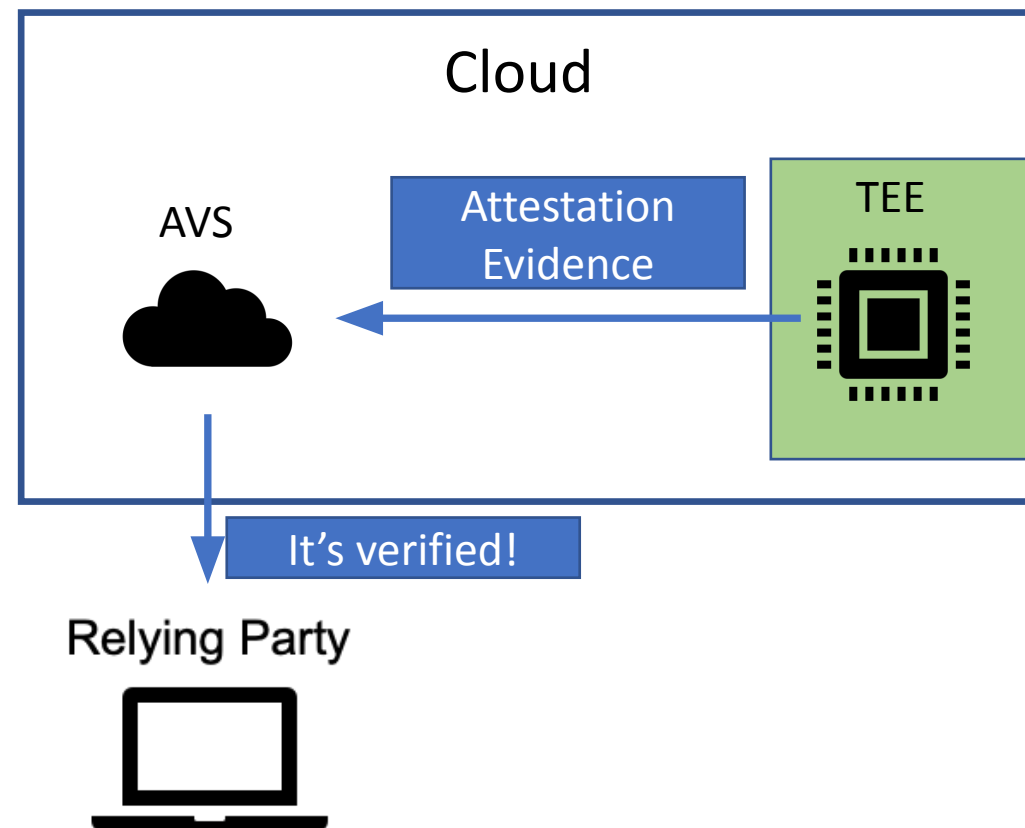


Why current attestation verification services are not perfect: Trust boundary is expanded.

Practice

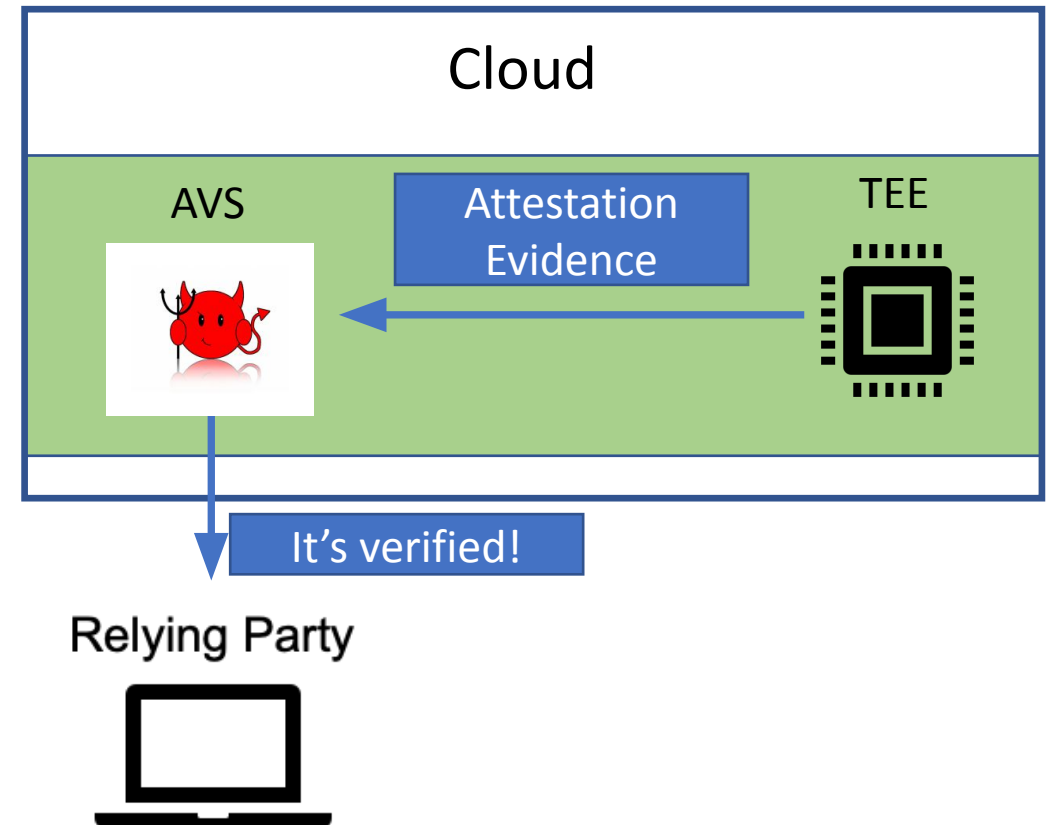


Ideal



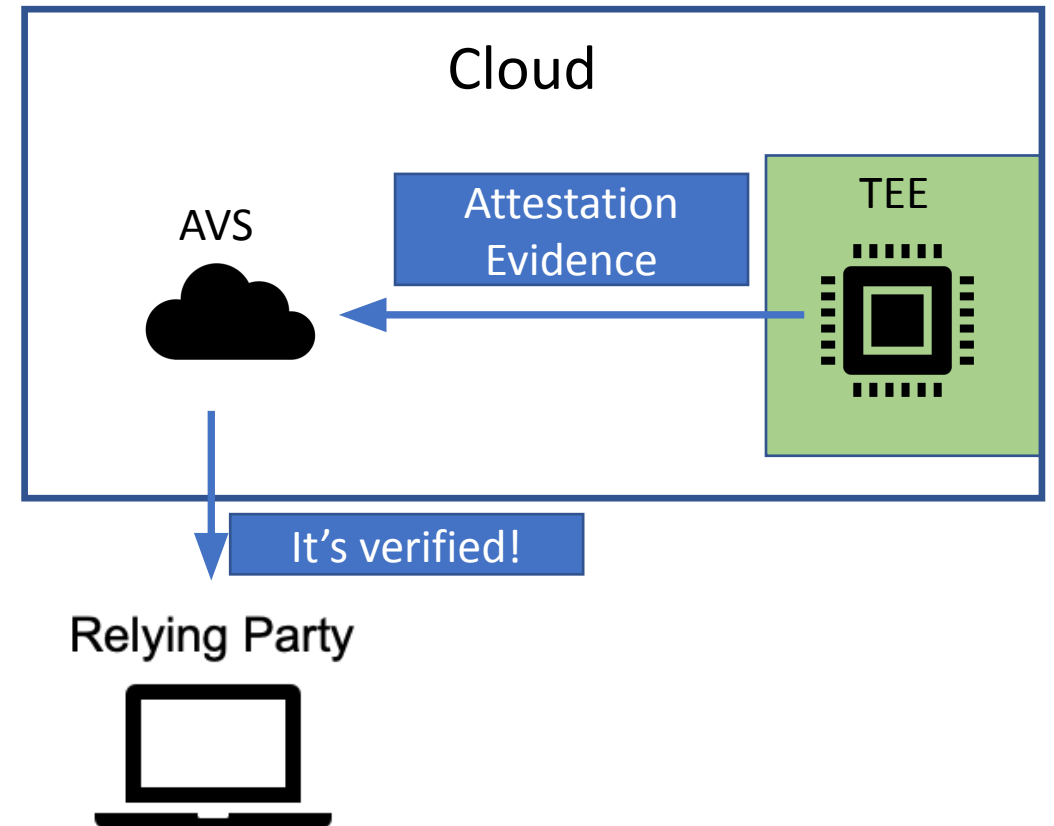
Although the cloud service providers are trustworthy most of the time...

- What if the attestation verification service is malfunctioning or buggy?
- What if the attacker launches an attack on the remote attestation itself instead of TEE?
- Larger trust boundary means larger risks.



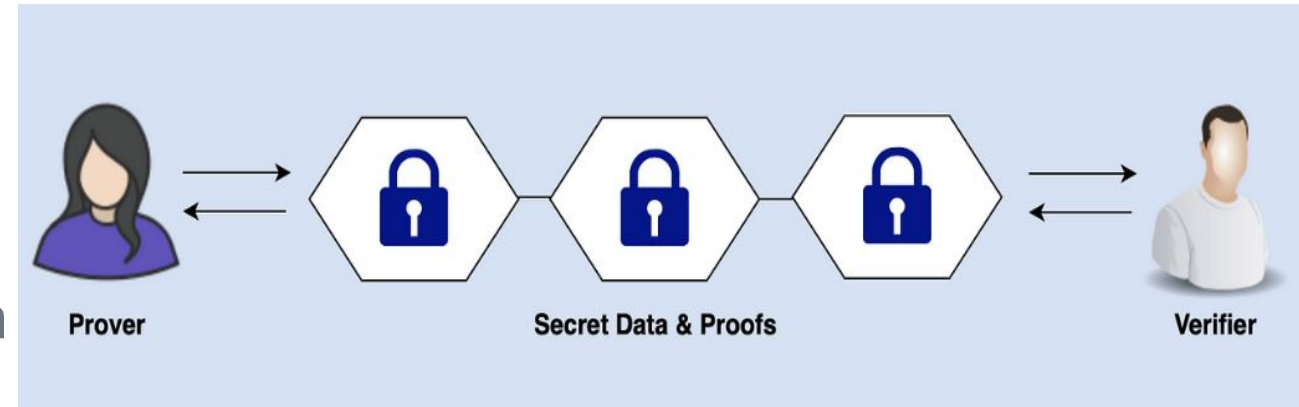
Can we make AVS trustless?

- Attestation verification service can prove to relying parties that it is following the correct attestation logic.
- Cryptographic Proof.
- Instead of trusting attestation service, you can trust math.



Zero Knowledge Proofs (ZKP) provides proof of computation without trusting third parties

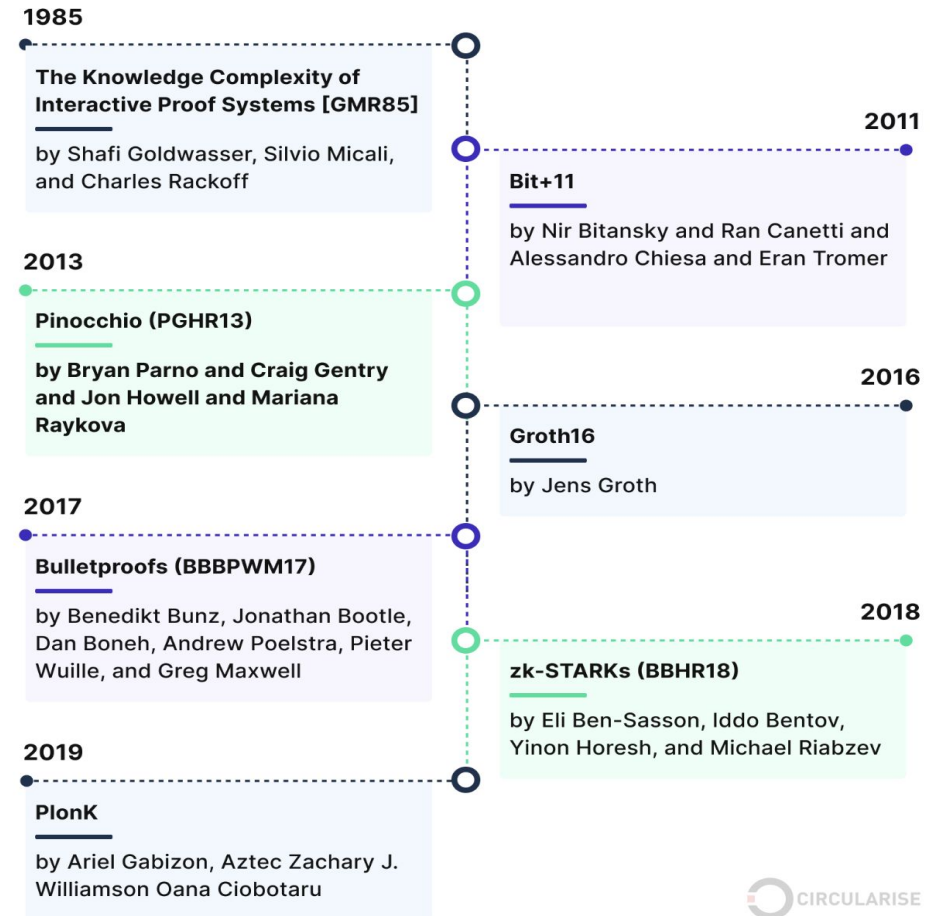
- ZKP allows one party (the prover) to prove to another party (the verifier) that a statement is true without revealing any additional information.
- Examples:
 - prove that you're a citizen of a country, without giving your name or passport number.
 - Prove that your age is over 18, without disclosing your age or your ID card.



ZKP is efficient for large scale computation

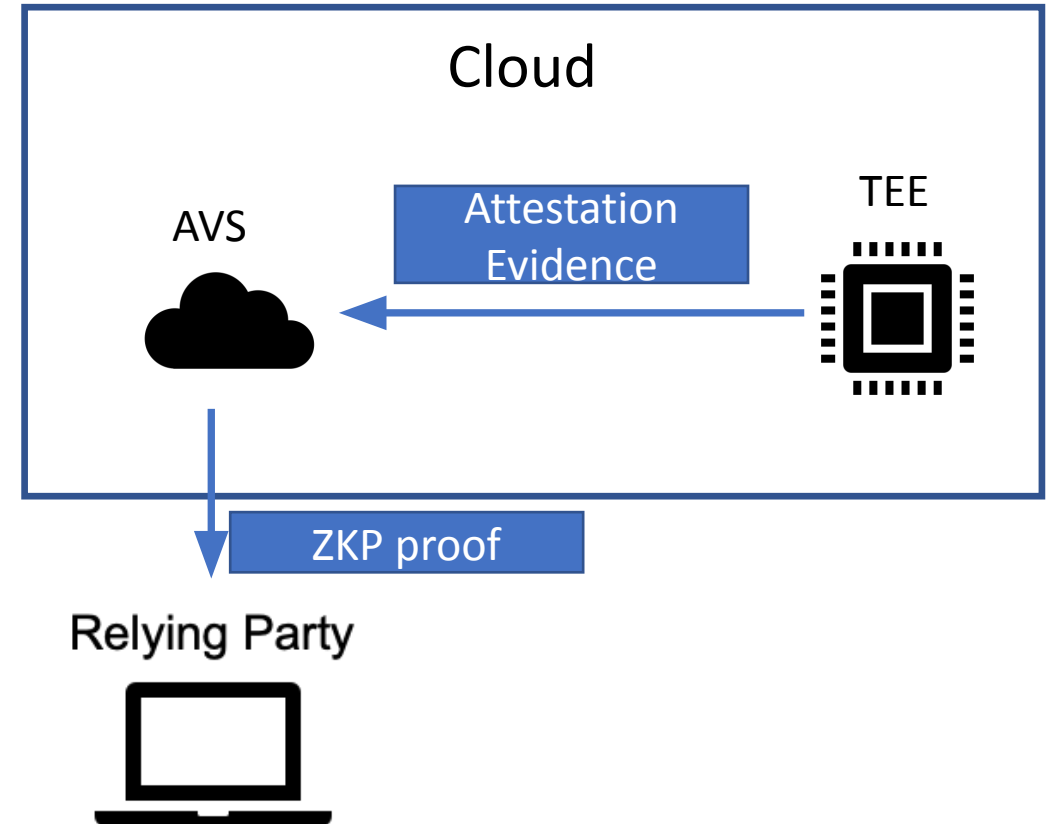
- ZKP becomes efficient enough now to prove large circuits.
- In Blockchain, ZKP is widely used to prove the correctness of computation.
 - E.g. Ethereum uses ZKP to verify off-chain computations.
- ZK-SNARK (Succinct Non-interactive Argument of Knowledge)
 - Fast verification
 - Constant proof size

A timeline of zero-knowledge



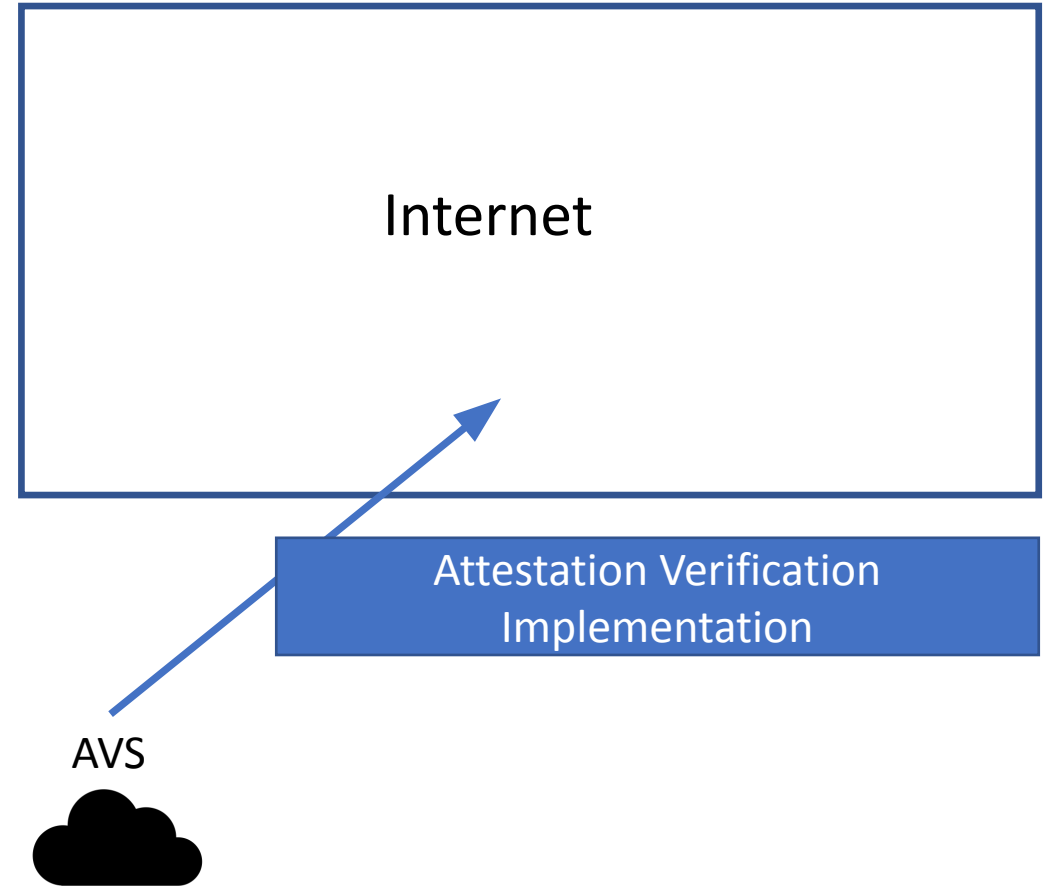
Our solution: embed ZKP into attestation verification

- Ask attestation verification service to generate a ZKP proof, so that anyone can verify that the attestation service honestly executed expected attestation logic.



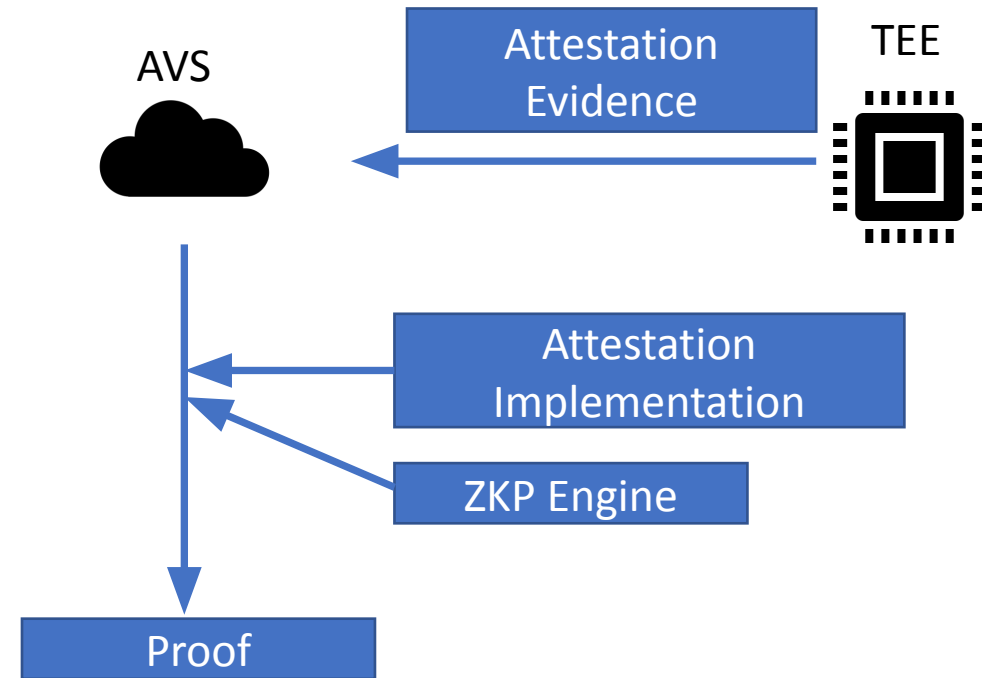
Work Flow: setup

- Attestation verification implementation should be open-sourced and public to everyone.
- Any party can audit, endorse, or challenge it.
- Finally, we have an attestation verification logic that is publicly endorsed.



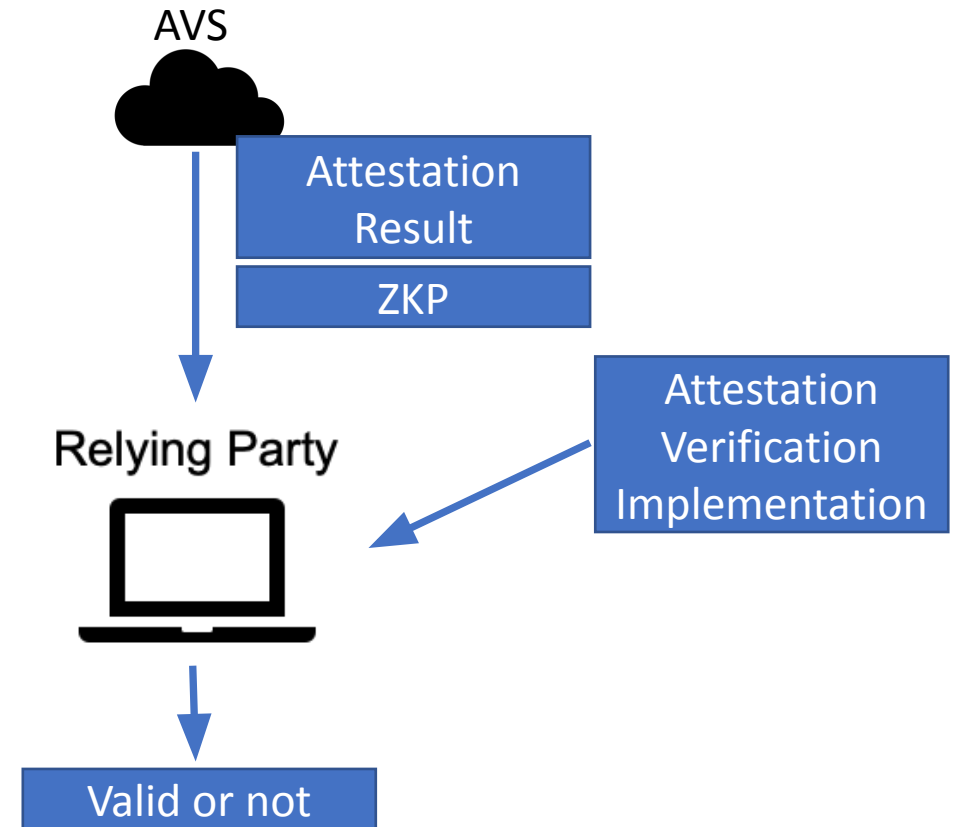
Work Flow: proof of attestation verification

- During attestation verification, AVS proves that the code it runs is consistent with the open-sourced version.



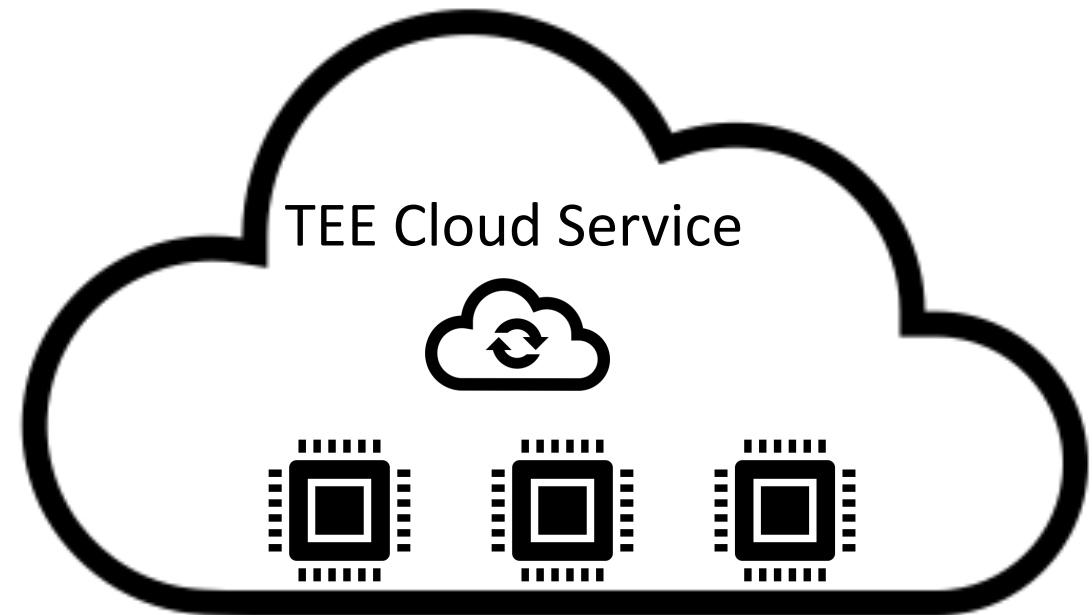
Work Flow: verifying Zero-knowledge Proof

- Relying parties can quickly verify whether the ZKP is valid.



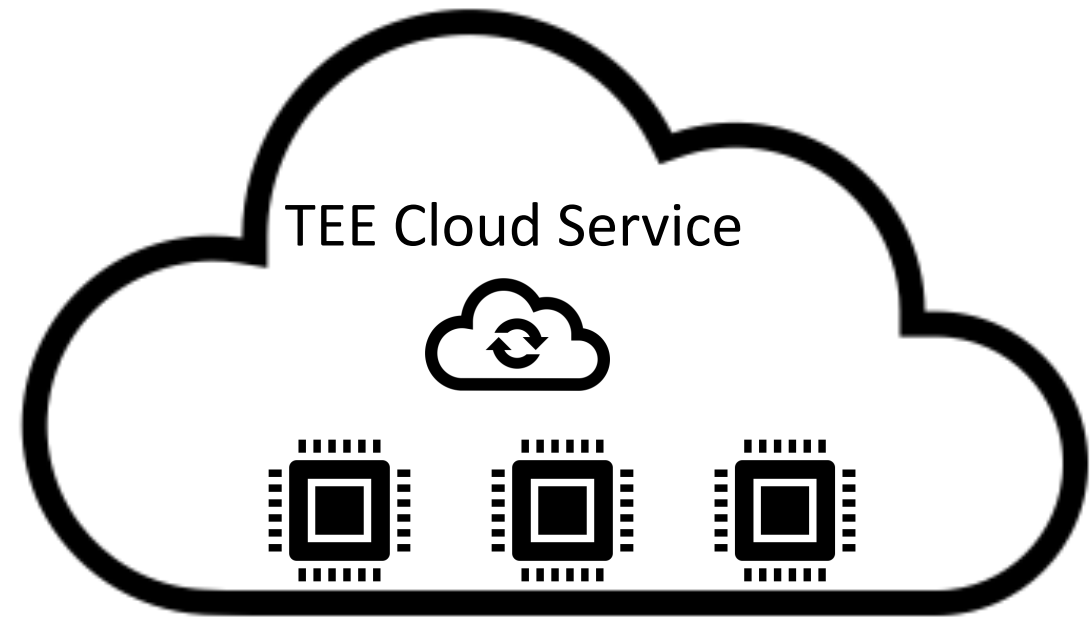
Challenges introduced by distributed confidential computing

- Multiple TEE nodes collaboratively provide services.
- Relying party needs to ensure all of them are legit.
- TEEs require mutual attestation.



Minimizing the efforts of relying party

Relying party still only verifies one proof and ensures that the entire cloud is attested properly.



Proof Aggregation for ZKP: verify everything in one shot

The final proof is valid only if all single proofs and aggregated proofs are valid.

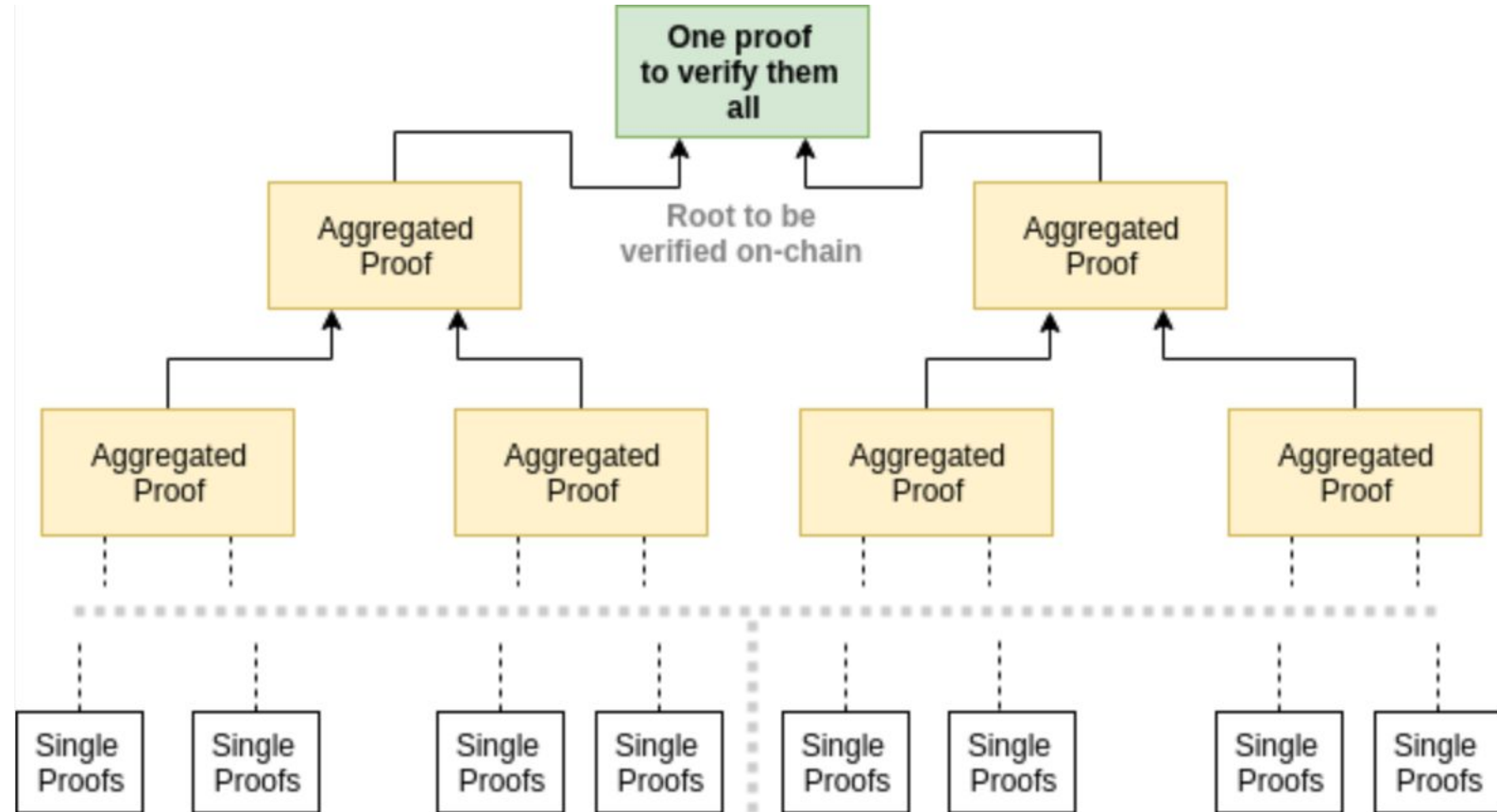
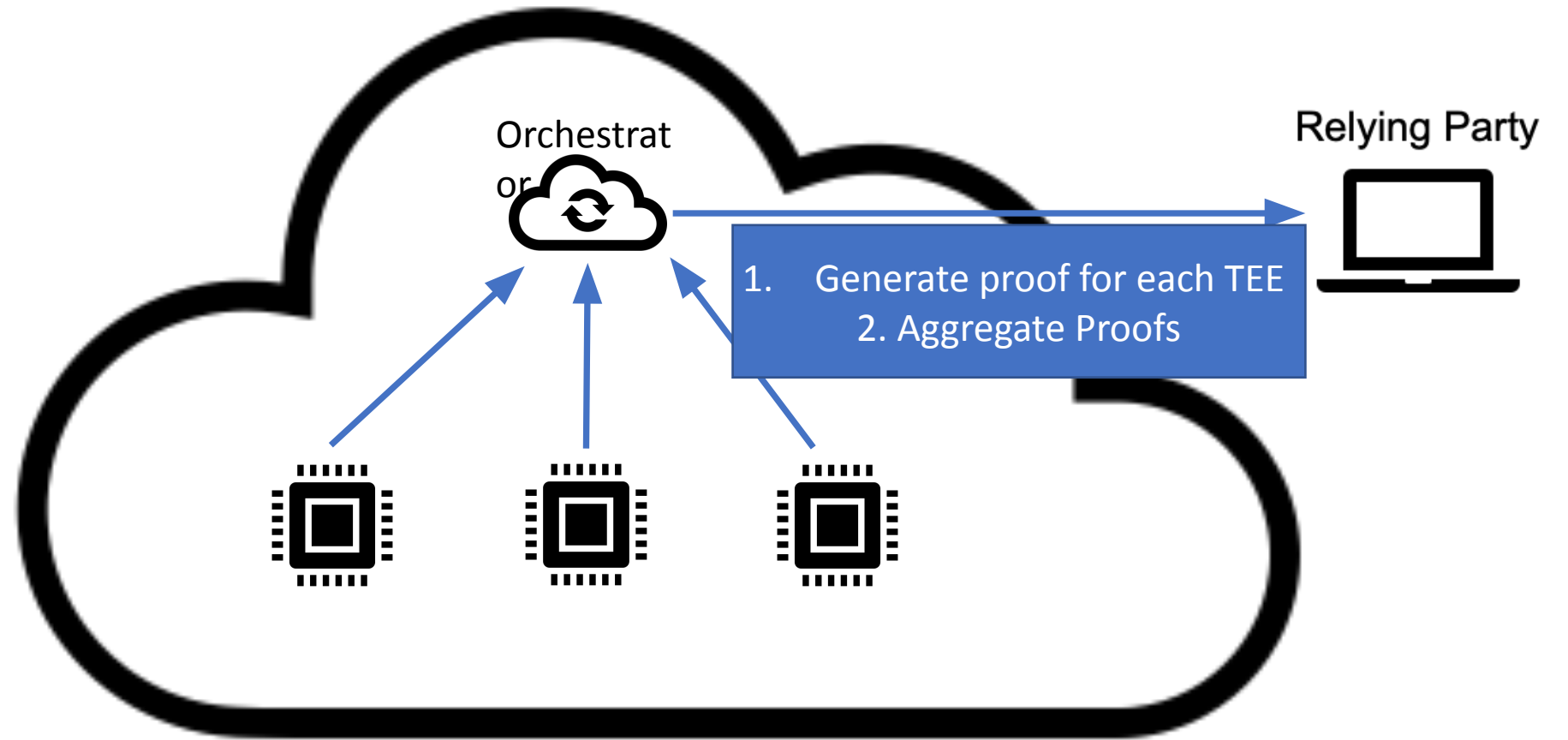


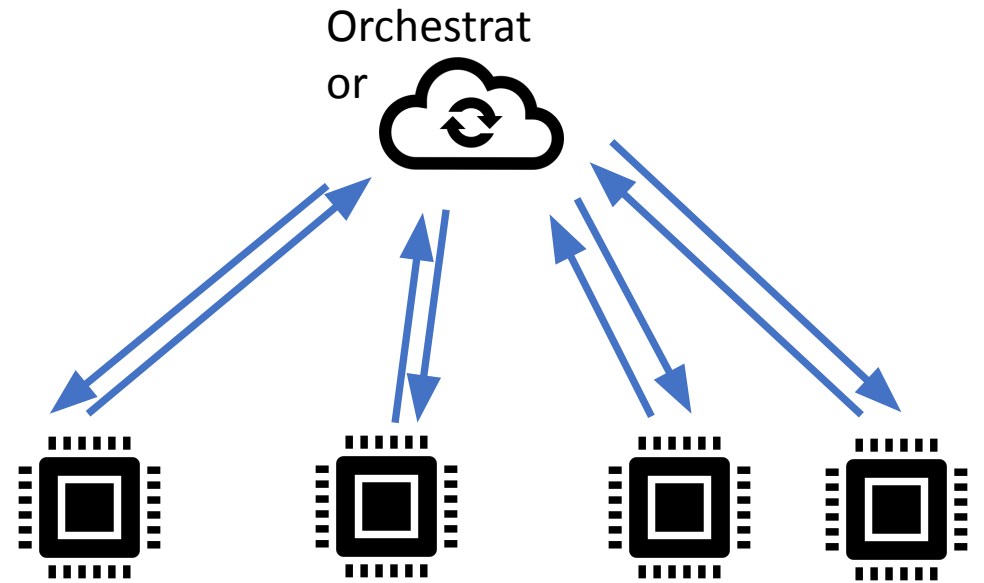
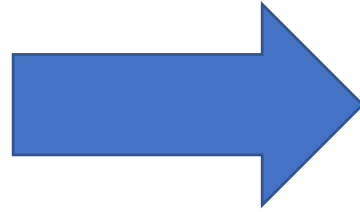
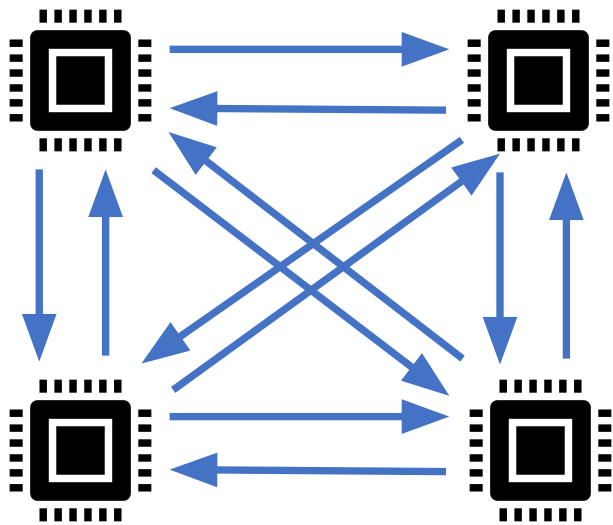
Image Source:

<https://scryptplatform.medium.com/recursive-zero-knowledge-proofs-27f2d934f953>

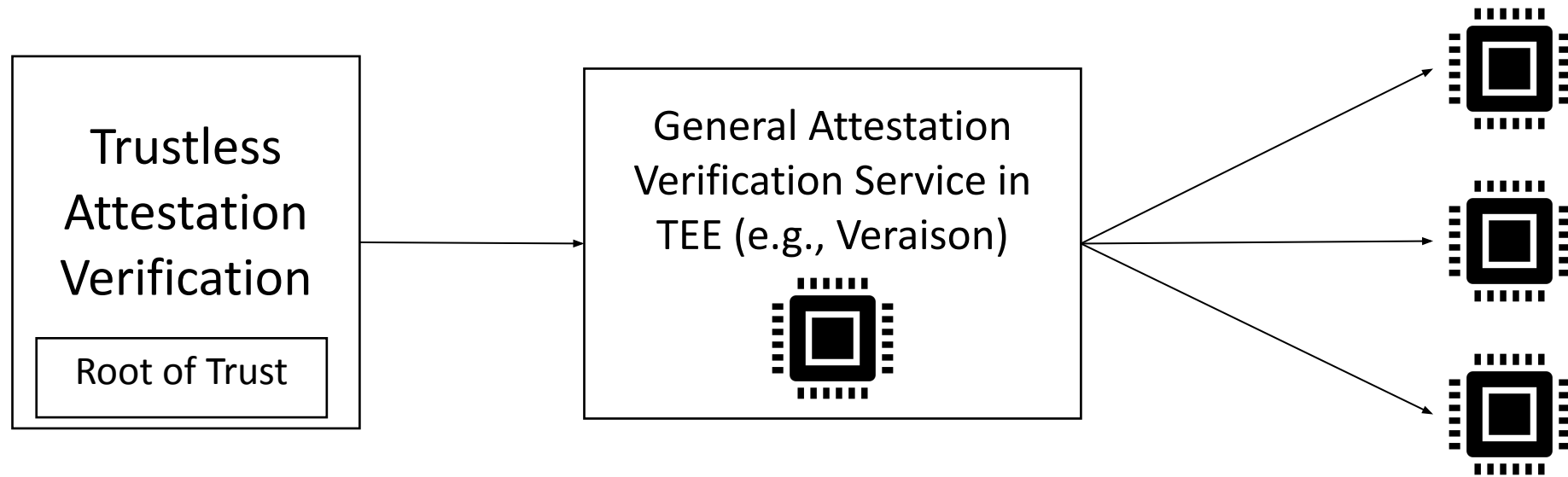
Proof aggregation in distributed TEE system



Mutual attestation becomes easier



Alternative Setup: Using trustless attestation to bootstrap a TEE-based attestation verification service



Open Source Status

- Implementation with various ZKP backends.
 - Circom
 - Circuit-based ZKP backend supporting Groth16.
 - Pros: Transparent trusted setup, clear security model.
 - Cons: Slow.
 - RiscZero
 - VM-based ZKP backend supporting Rust as high level language.
 - Pros: (1) easy to implement and audit. (2) Fast.
 - Cons: More complicated security model.

We have already open-sourced two repositories.

The screenshot shows the GitHub interface for the repository 'trustless-attestation-verification' under the user 'tiktok-privacy-innovation'. The repository is public and has 51 stars, 5 forks, and 0 watches. The main branch is 'main'. A recent pull request (PR) #3 is merged, titled 'Merge pull request #3 from backnotprop/fix/keypair-parameter-order', dated 6 months ago with 4 commits. The file list includes 'samples' (init commit, 9 months ago), 'tool' (Fix parameter order in prepare_keypair function call, 6 months ago), 'LICENSE' (init commit, 9 months ago), and 'README.md' (init commit, 9 months ago). The README is expanded, showing the title 'Trustless Attestation Verification' and a section 'Background' which states: 'Confidential computing is revolutionizing the way sensitive data is processed and shared in the digital era, enabling applications that were once considered impossible. At its core, Trusted Execution Environments (TEEs)'. The right sidebar shows repository details: 'About' (No description, website, or topics provided), 'Readme', 'GPL-3.0 license', 'Code of conduct', 'Contributing', 'Activity', 'Custom properties', '51 stars', '0 watching', '5 forks', 'Report repository', 'Releases' (1 tags, 'Create a new release' link).

trustless-attestation-verification Public

main 1 Branch 1 Tag

Go to file t Add file <> Code

lu562 Merge pull request #3 from backnotprop/fix/keypair-parameter-order 2ad7832 · 6 months ago 4 Commits

samples	init commit	9 months ago
tool	Fix parameter order in prepare_keypair function call	6 months ago
LICENSE	init commit	9 months ago
README.md	init commit	9 months ago

README Code of conduct Contributing GPL-3.0 license

Trustless Attestation Verification

Background

Confidential computing is revolutionizing the way sensitive data is processed and shared in the digital era, enabling applications that were once considered impossible. At its core, Trusted Execution Environments (TEEs)

About No description, website, or topics provided.

- Readme
- GPL-3.0 license
- Code of conduct
- Contributing
- Activity
- Custom properties

51 stars 0 watching 5 forks Report repository

Releases 1 tags Create a new release

<https://github.com/tiktok-privacy-innovation/trustless-attestation-verification>

We have already open-sourced two repositories.

The screenshot shows the GitHub interface for the repository 'trustless-attestation-verification-circom'. The repository is public and has 56 stars and 9 forks. It contains a table of files and folders with their commit history. The 'About' section on the right indicates no description is provided. The 'Releases' section shows 1 tag. The 'Packages' section shows no published packages. The 'Contributors' section lists ArmanKolozyan.

File/Folder	Commit Message	Time Ago
circuits	fix: padding issue in RSA-PSS	2 days ago
docs	fix: rust CLI	7 months ago
images	Initial commit	9 months ago
lib	fix: add submodule	8 months ago
tool	deps: bump sev to v7.1.0	yesterday
.gitmodules	Initial commit	9 months ago
Dockerfile	Initial commit	9 months ago
LICENSE	Initial commit	9 months ago
Makefile	fix: add submodule	8 months ago
README.md	Initial commit	9 months ago
circomkit.json	Initial commit	9 months ago
package.json	Initial commit	9 months ago

Trustless Attestation Verification (Circom)

<https://github.com/tiktok-privacy-innovation/trustless-attestation-verification-circom>

We have already open-sourced two repositories.

Circom Implementation

- Groth16 proof (zk-SNARK) for AMD-SEV-SNP attestation verification.
- User-friendly CLI based on Makefile.

==> circom
CIRCUIT COMPILER

Steps

0. Pull and initiate submodules

This command will pull all the required submodules and set them up.

```
make prepare
```

1. Launch Trusted Setup

Run the following command to initiate the trusted setup:

```
make trusted_setup
```

This generates a .ptau file containing the common reference string required by ZK-SNARK. Please note that this process can be time-consuming (approximately 1.5 hours based on our tests).

2. Compile Circuits








Next, compile the circuits:

```
make compile_circuits
```

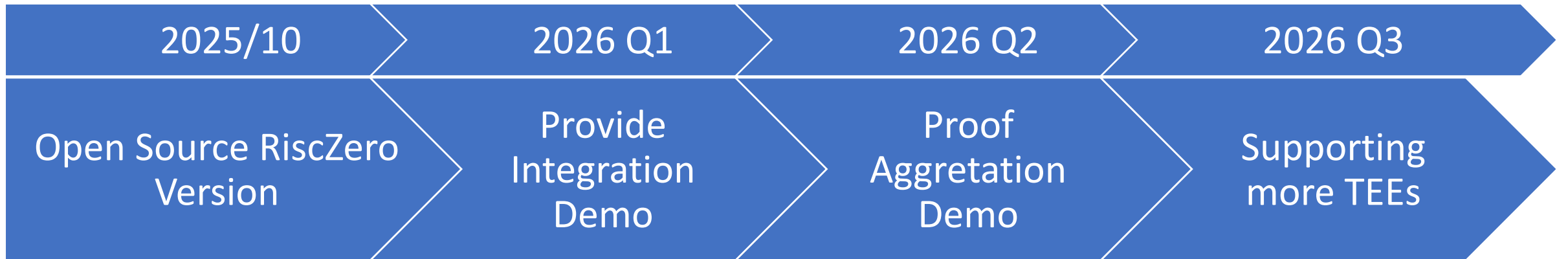
We got support from the community!

- Connected with multiple ZKP companies.
- Independent community contributors have opened 20+ issues and submitted 6 pull requests.
- We are also testing the deployment in the internal use cases.
- It's still early, but the momentum is solid.

✕ Clear current search query, filters, and sorts

<input type="checkbox"/>		0 Open	<input checked="" type="checkbox"/>	6 Closed	Author ▾	Label ▾	Pr
<input type="checkbox"/>		deps: Bump sev crate from v4.0.0 to v7.1.0 ✓					
		#24 by hyperfinitism was merged last week					
<input type="checkbox"/>		Added missing semicolon in mgf1sha384 ✓					
		#22 by hyperfinitism was merged 2 weeks ago					
<input type="checkbox"/>		Fixed logic and documentation issues in mask generation templates ✓					
		#20 by ArmanKolozyan was merged on May 28					
<input type="checkbox"/>		Fixed incorrect type and zero-padding constraint bug in RsaVerifySsaPss ✓					
		#19 by ArmanKolozyan was merged on May 28					
<input type="checkbox"/>		Fixed underconstrained signals and removed reversal duplication in signature validator ✓					
		#18 by ArmanKolozyan was merged on May 28					
<input type="checkbox"/>		Range assumptions for byte and word inputs ✓					
		#17 by ArmanKolozyan was merged on May 27					

Timeline



Thank You!