black hat USA 2024

AUGUST 7-8, 2024

BRIEFINGS

Crashing the Party: Vulnerabilities in RPKI Validation

Donika Mirdita, Niklas Vogel, Haya Schulmann, Michael Waidner

Outline

❖ Resource Public Key Infrastructure (RPKI)

- ✓ A niche new protocol
- ✓ & why it matters

❖ Systemic Analysis of RPKI Software

- ✓ Introducing a bespoke fuzzing mechanism
- ✓ & how it works

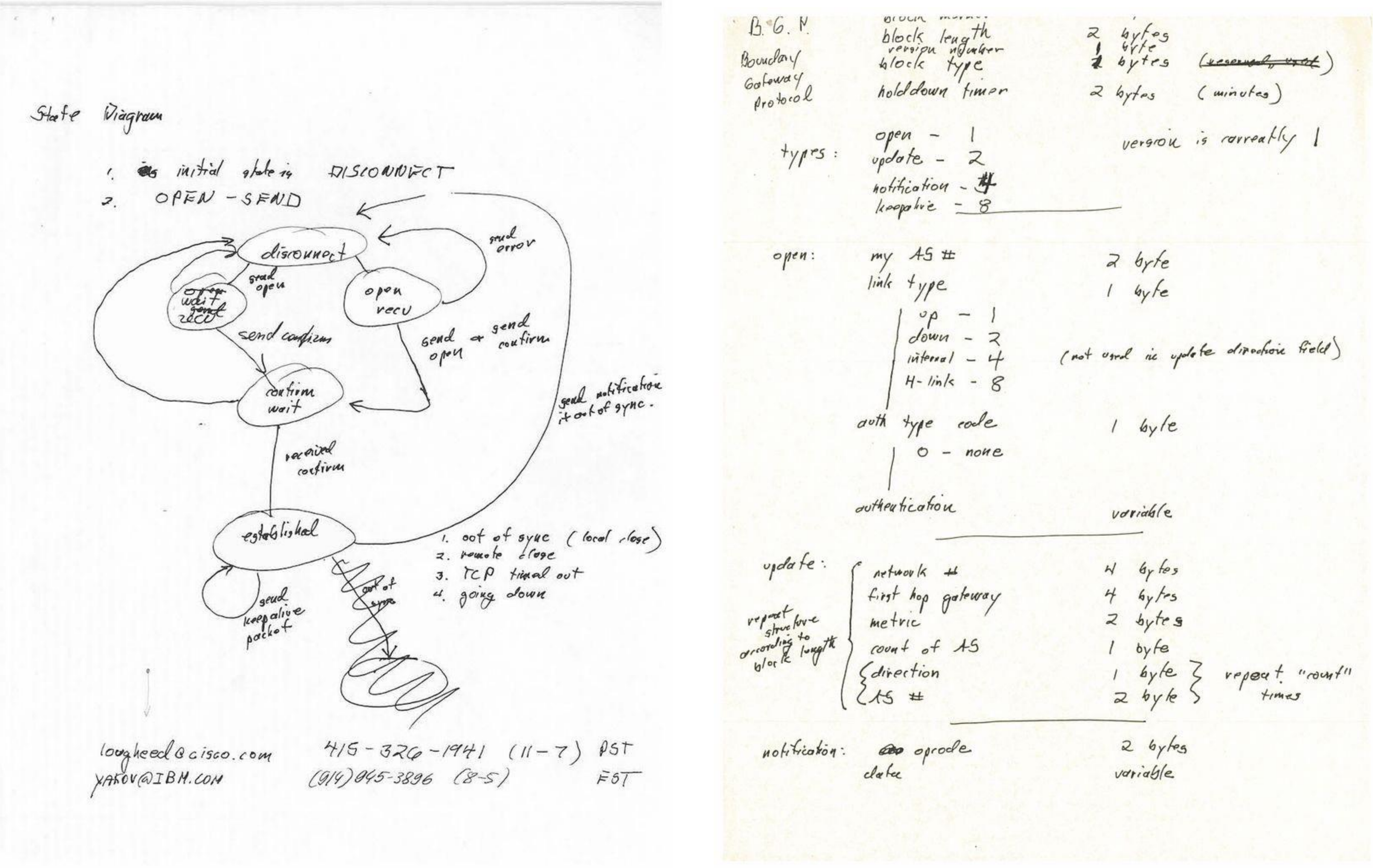
❖ Analysis Results

- ✓ What they mean
- ✓ & consequences

❖ Disclosure Process

BGP as Achille's Heel

BGP as Achille's Heel



Notes from the IETF Cafeteria, 1989

BGP as Achille's Heel

Cloudflare blames recent outage on BGP hijacking incident

By **Bill Toulas** July 5, 2024 02:41 PM 1

Russian telco hijacks internet traffic for Google, AWS, Cloudflare, and others

Rostelecom involved in BGP hijacking incident this week impacting more than 200 CDNs and cloud providers.

Written by **Catalin Cimpanu,** Contributor
April 5, 2020 at 2:53 p.m. PT

ROUTING SECURITY INCIDENTS

For 12 Hours, Was Part of Apple Engineering's Network Hijacked by Russia's Rostelecom?

By Aftab Siddiqui • 27 Jul 2022

OUTAGE ANALYSES

Twitter Outage Analysis: March 28, 2022

By Chris Villemez | April 15, 2022 | 14 min read

The RPKI Protocol

[RFC Home] [TEXT|PDF|HTML] [Tracker] [IPR] [Errata] [Info page]

INFORMATIONAL
Errata Exist

Internet Engineering Task Force (IETF)
Request for Comments: 6480
Category: Informational
ISSN: 2070-1721
February 2012

M. Lepinski
S. Kent
BBN Technologies

An Infrastructure to Support Secure Internet Routing

Abstract

This document describes an architecture for an infrastructure to support improved security of Internet routing. The foundation of this architecture is a Resource Public Key Infrastructure (RPKI) that represents the allocation hierarchy of IP address space and Autonomous System (AS) numbers; and a distributed repository system for storing and disseminating the data objects that comprise the RPKI, as well as other signed objects necessary for improved routing security. As an initial application of this architecture, the document describes how a legitimate holder of IP address space can explicitly and verifiably authorize one or more ASes to originate routes to that address space. Such verifiable authorizations could be used, for example, to more securely construct BGP route filters.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6480.

The RPKI Protocol

BleepingComputer

Comcast now blocks BGP hijacking attacks and route leaks with RPKI

Comcast, one of America's largest broadband providers, has now deployed RPKI on its network to defend against BGP route hijacks and leaks.

20 May 2021

How AWS is helping to secure internet routing

by Fredrik Korsbäck | on 13 JAN 2021 | in Announcements, Best Practices, Networking & Content Delivery,

Some of the larger service provider networks have implemented RPKI Origin Validation in the last year. This can be seen in the preceding chart (figure 5) by looking at the reduction of BGP prefixes with an Invalid RPKI state accepted by their networks. Telia Carrier deployed in February, and many other large operators followed suit afterwards. The number of

BleepingComputer

All Dutch govt networks to use RPKI to prevent BGP hijacking

The Dutch government will adopt the RPKI (Resource Public Key Infrastructure) standard on all its systems before the end of 2024 to upgrade...

9 Apr 2023

Verisign's Path to RPKI

By Mike Hollyman • 7 Jun 2023

case study RPKI

Capacity Media

Telia Carrier set to install RPKI to global backbone

Telia Carrier has announced that it will be implementing resource public key infrastructure (RPKI) technology to its global network.

17 Sept 2019

The RPKI Protocol

How AWS
by Fredrik Korsbä

BleepingComputer

Harry Coker: Federal Agencies Advance Resource Public Key Infrastructure Adoption

by Jane Edwards · May 28, 2024 · 1 min read

attacks and route leaks

lers, has now deployed RPKI on its
ıks.

BGP HIJACKING —

FCC pushes ISPs to fix security flaws in Internet routing

Chair: Addressing BGP flaws will "help make our Internet routing more secure."

JON BRODKIN - 6/6/2024, 11:40 PM

BleepingComputer

All Dutch govt net
hijacking

The Dutch government wi
standard on all its system

9 Apr 2023

ix,
ıting

/ent crooks, spies hijacking victims'

'ue 31 Mar 2020 // 12:00 UTC

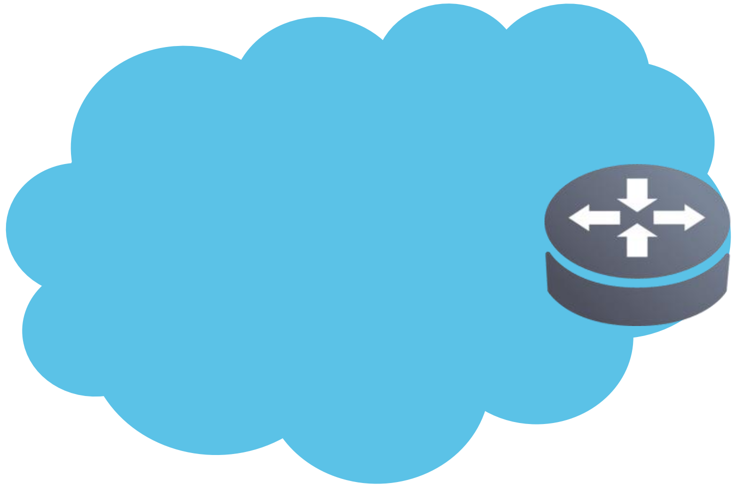
Capacity Media

Telia Carrier se

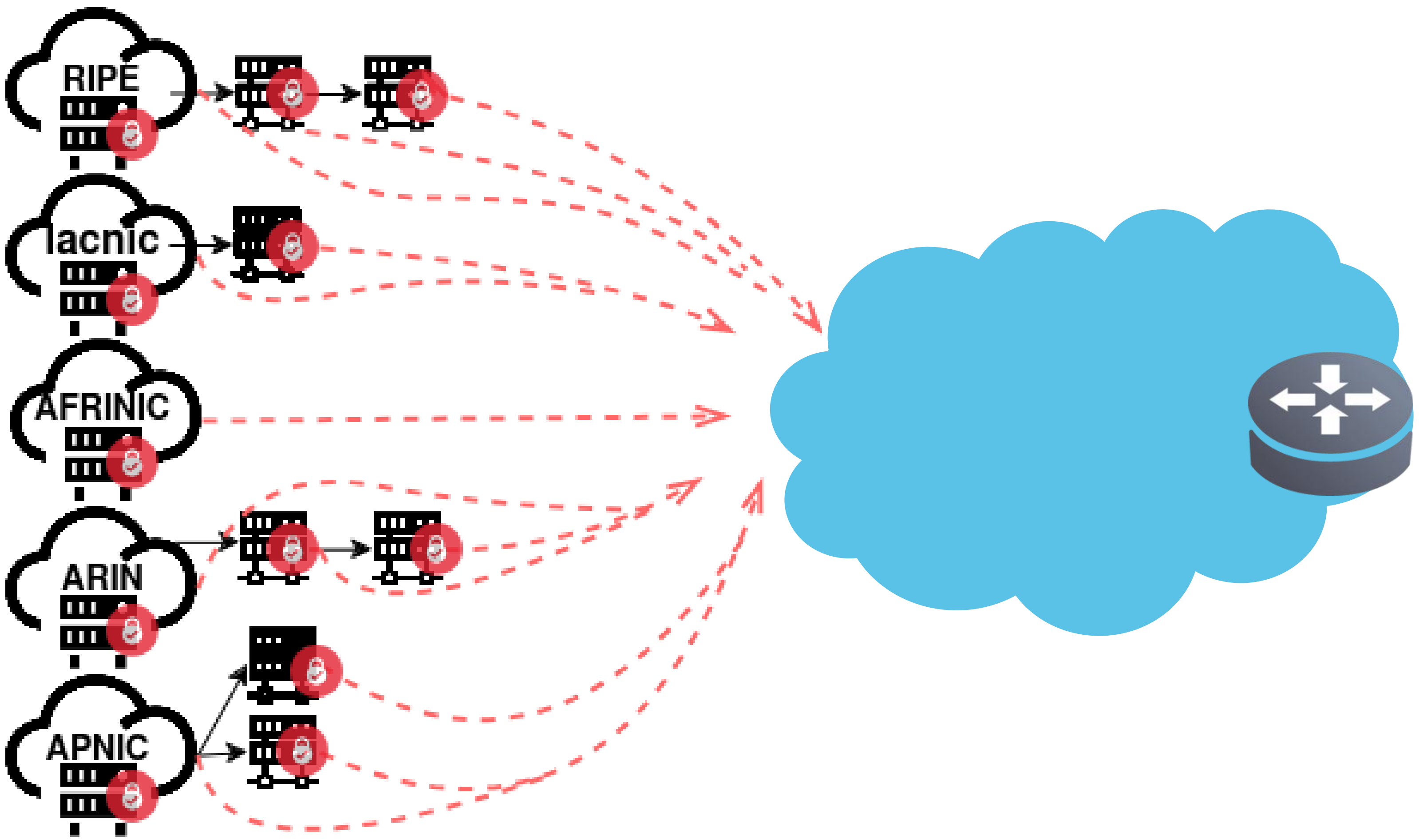
Telia Carrier has ann
infrastructure (RPKI)

17 Sept 2019

BGP Security with RPKI

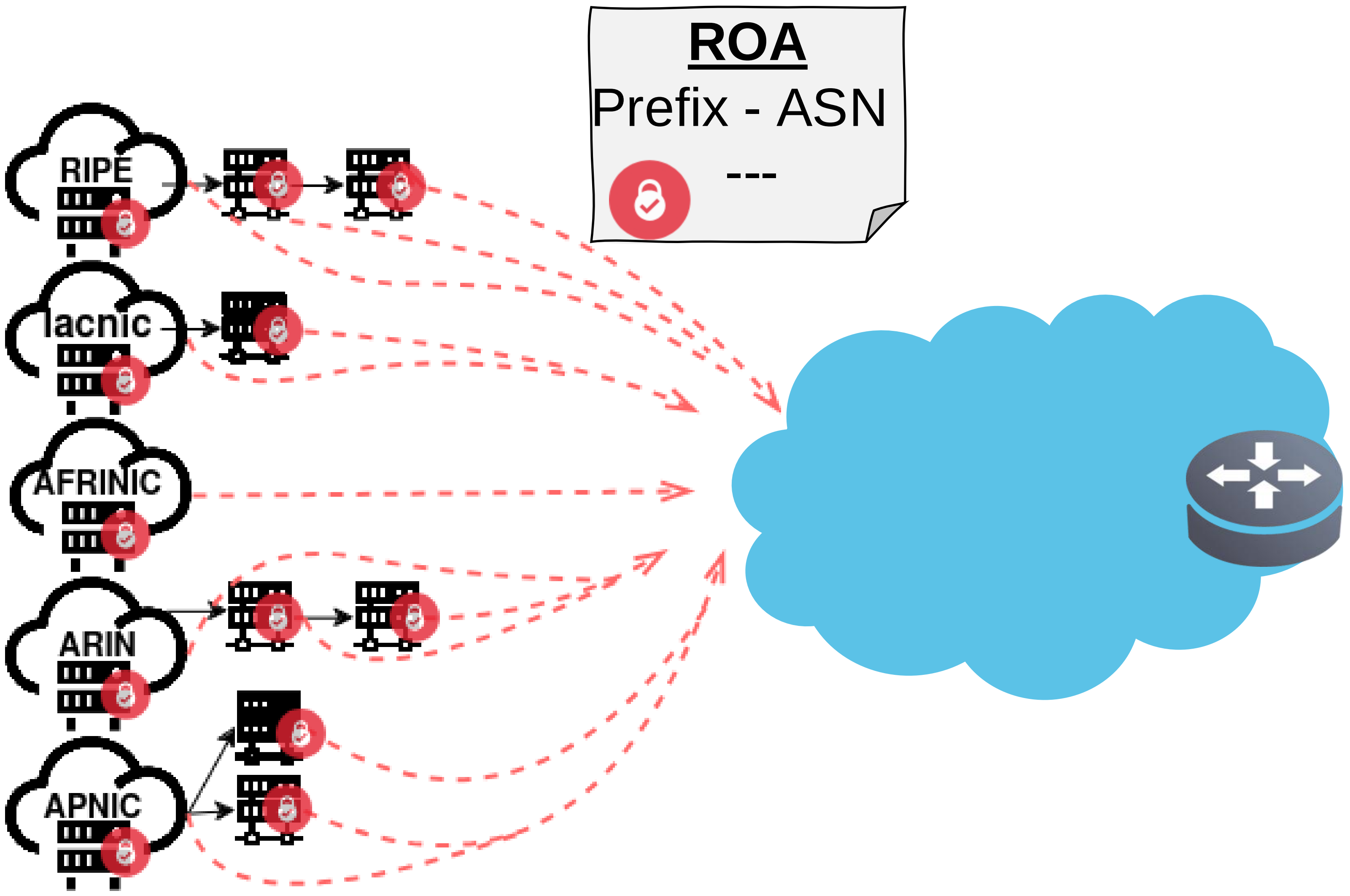


BGP Security with RPKI



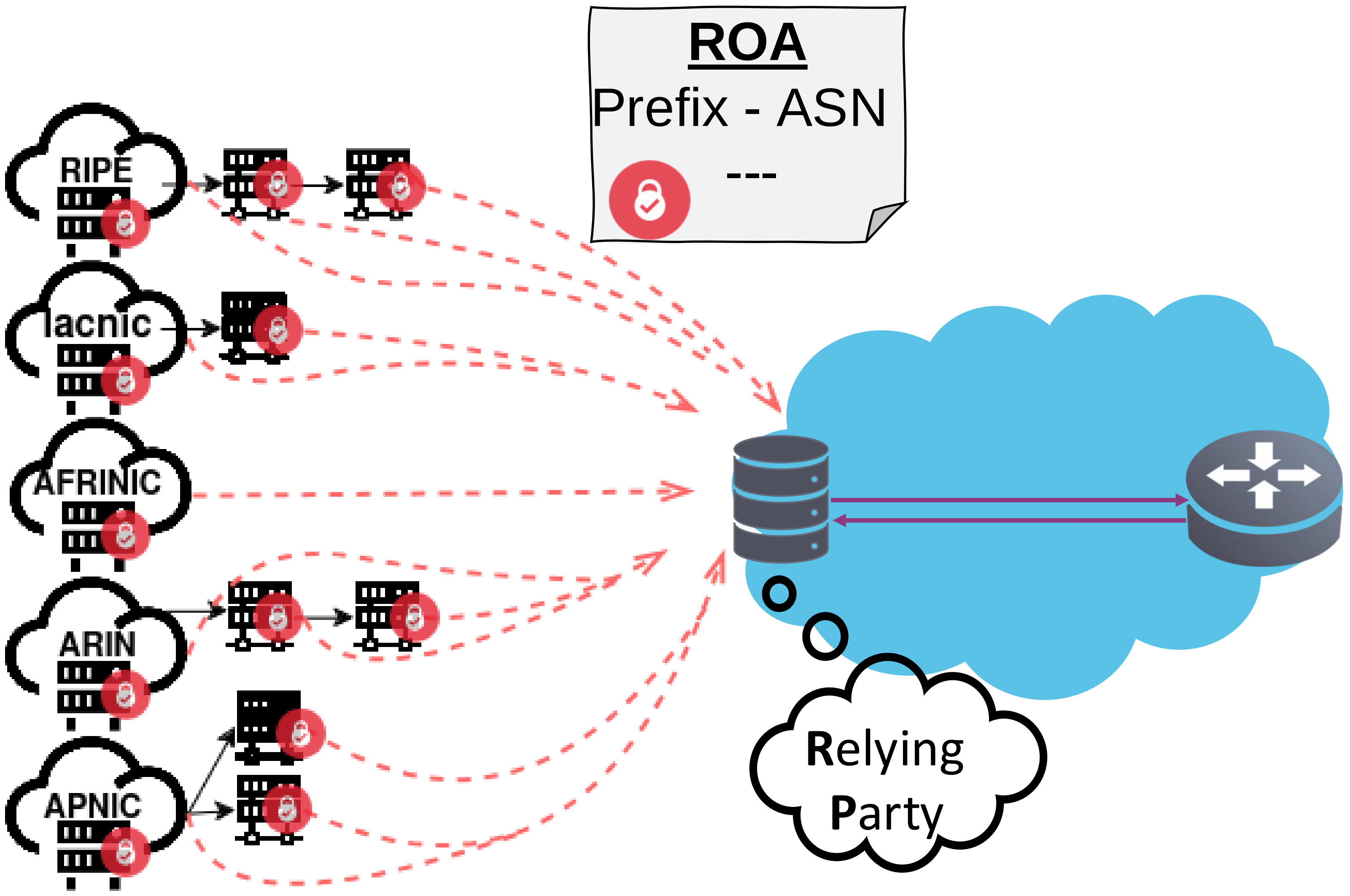
RPKI Repositories

BGP Security with RPKI



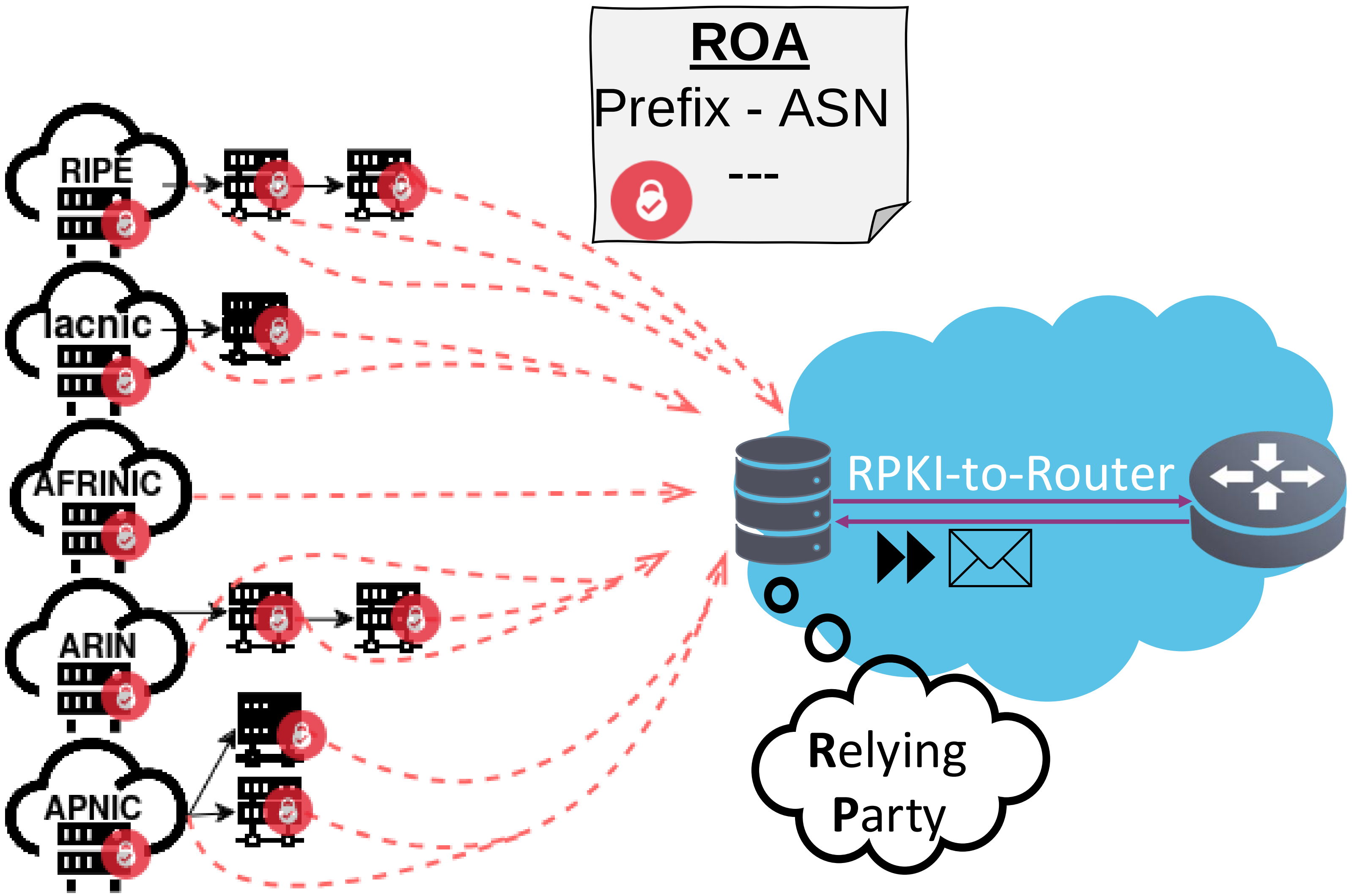
RPKI Repositories

BGP Security with RPKI



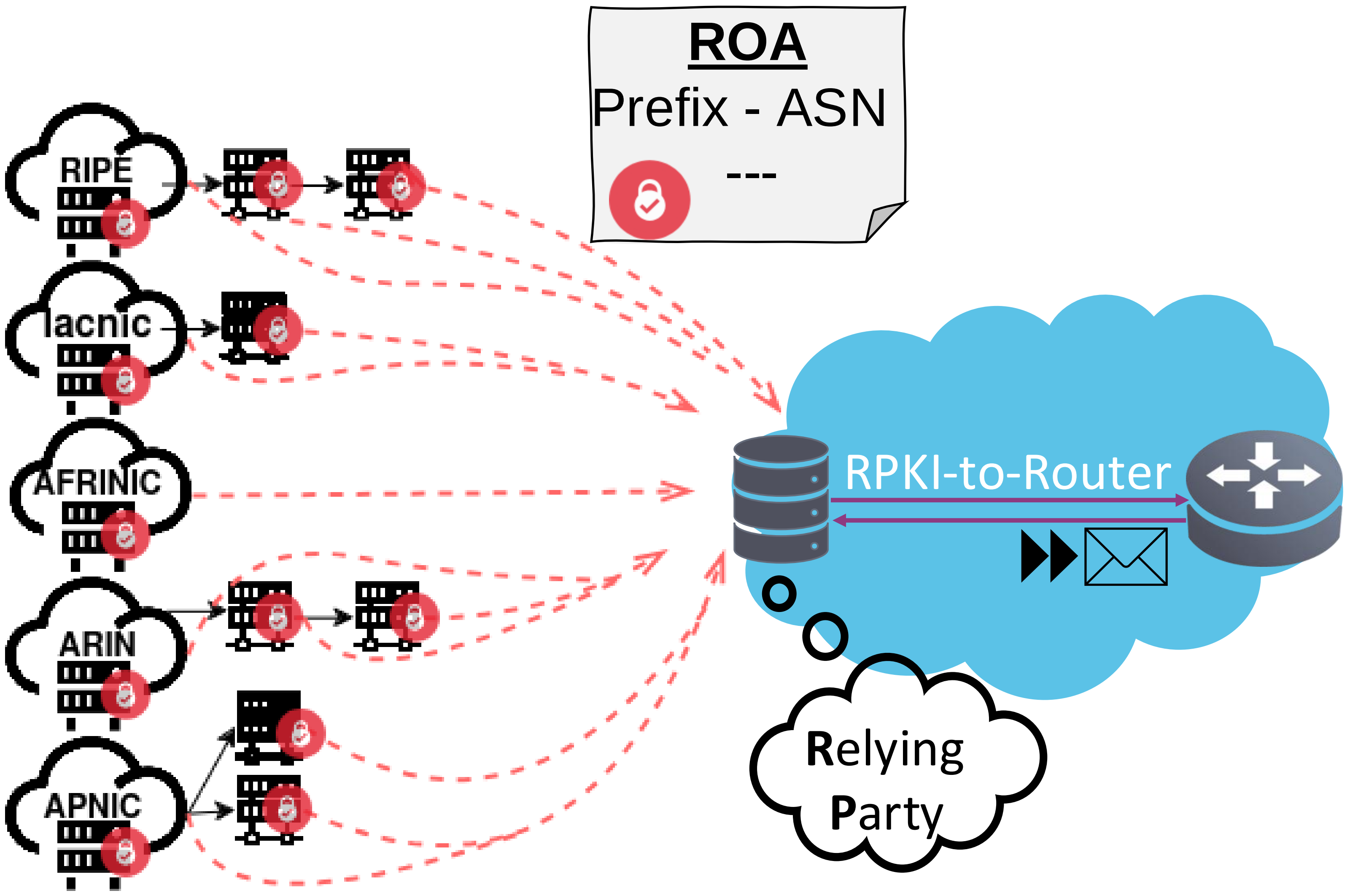
RPKI Repositories

BGP Security with RPKI



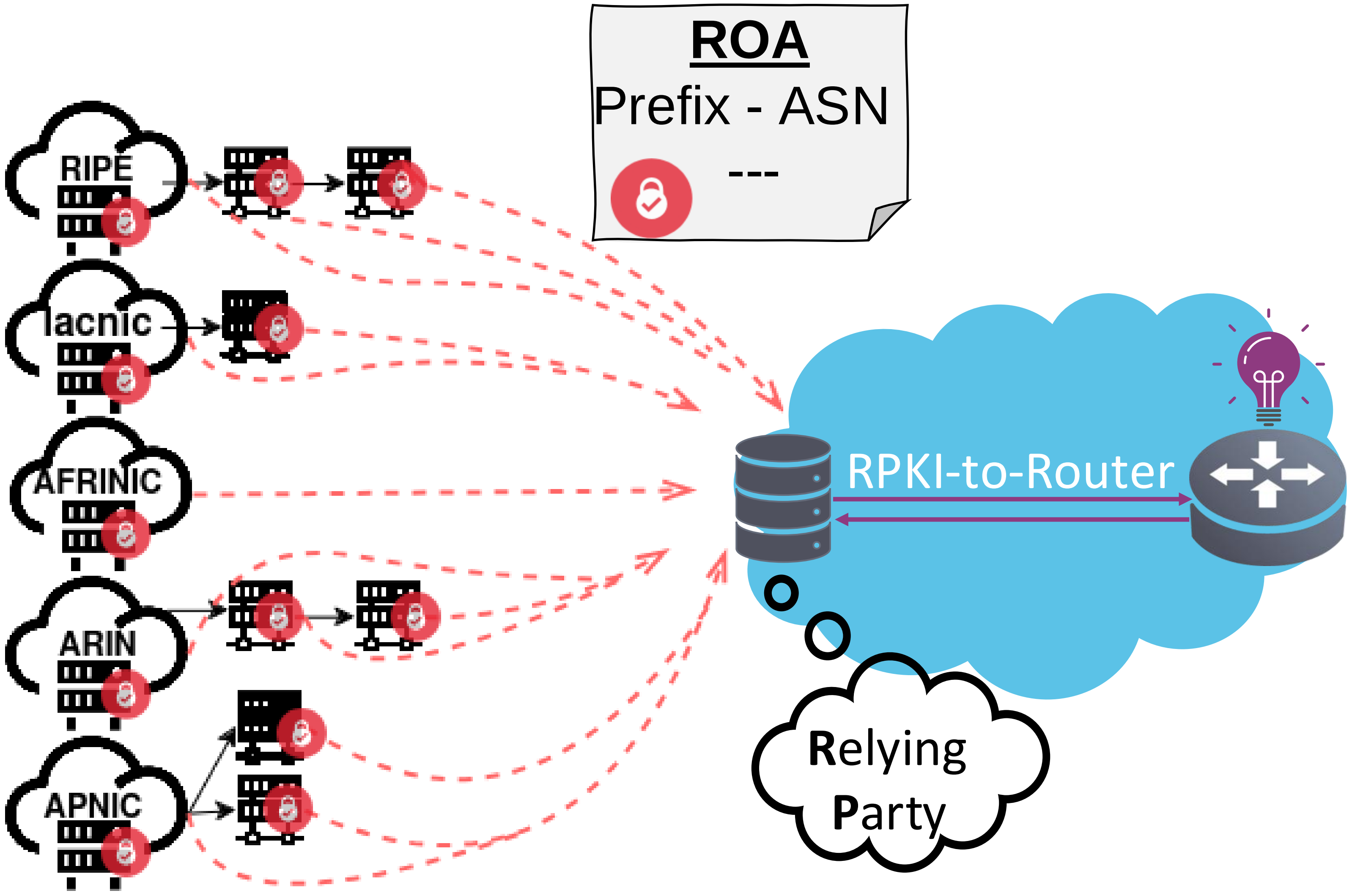
RPKI Repositories

BGP Security with RPKI



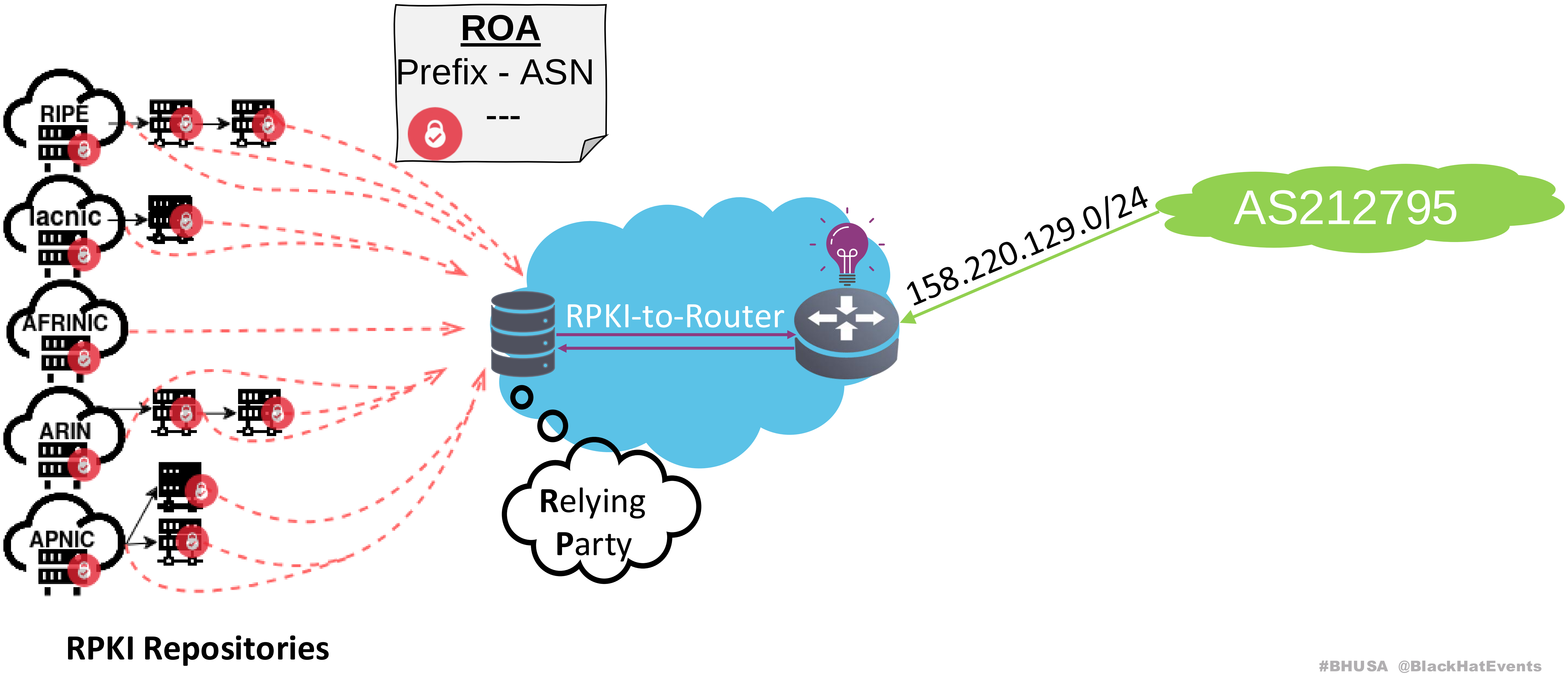
RPKI Repositories

BGP Security with RPKI

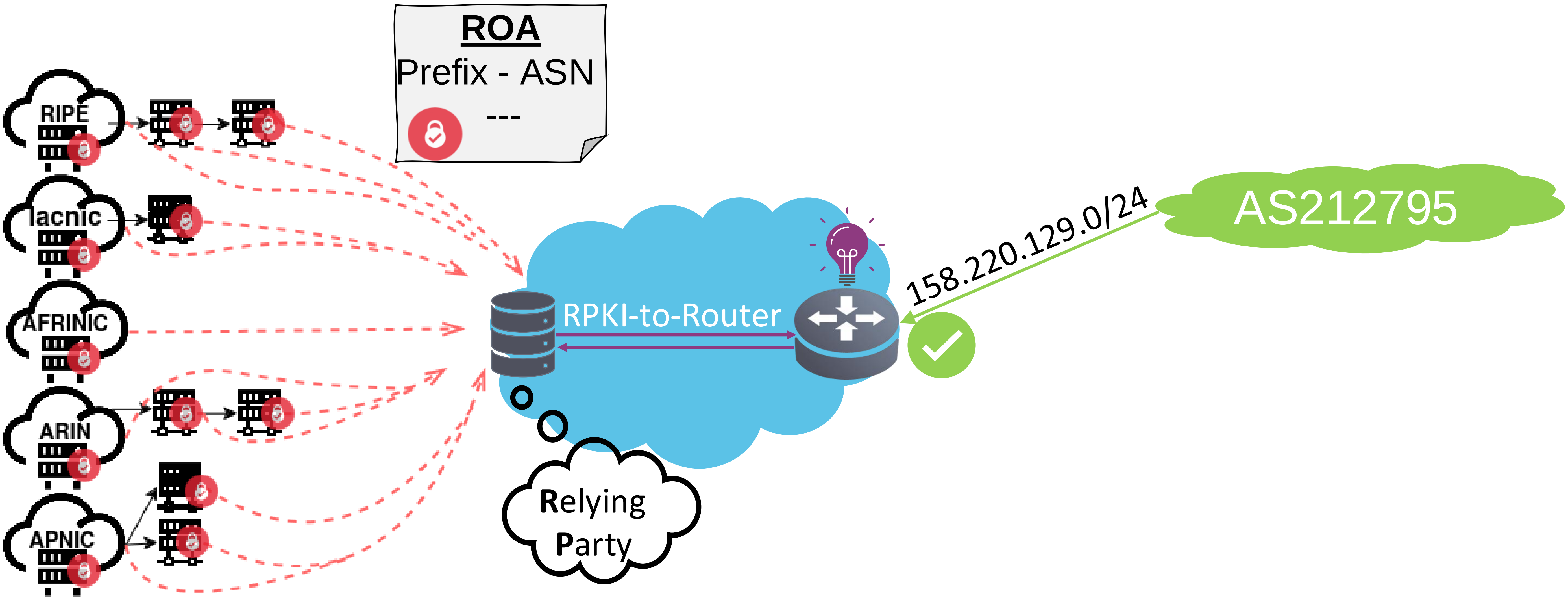


RPKI Repositories

BGP Security with RPKI

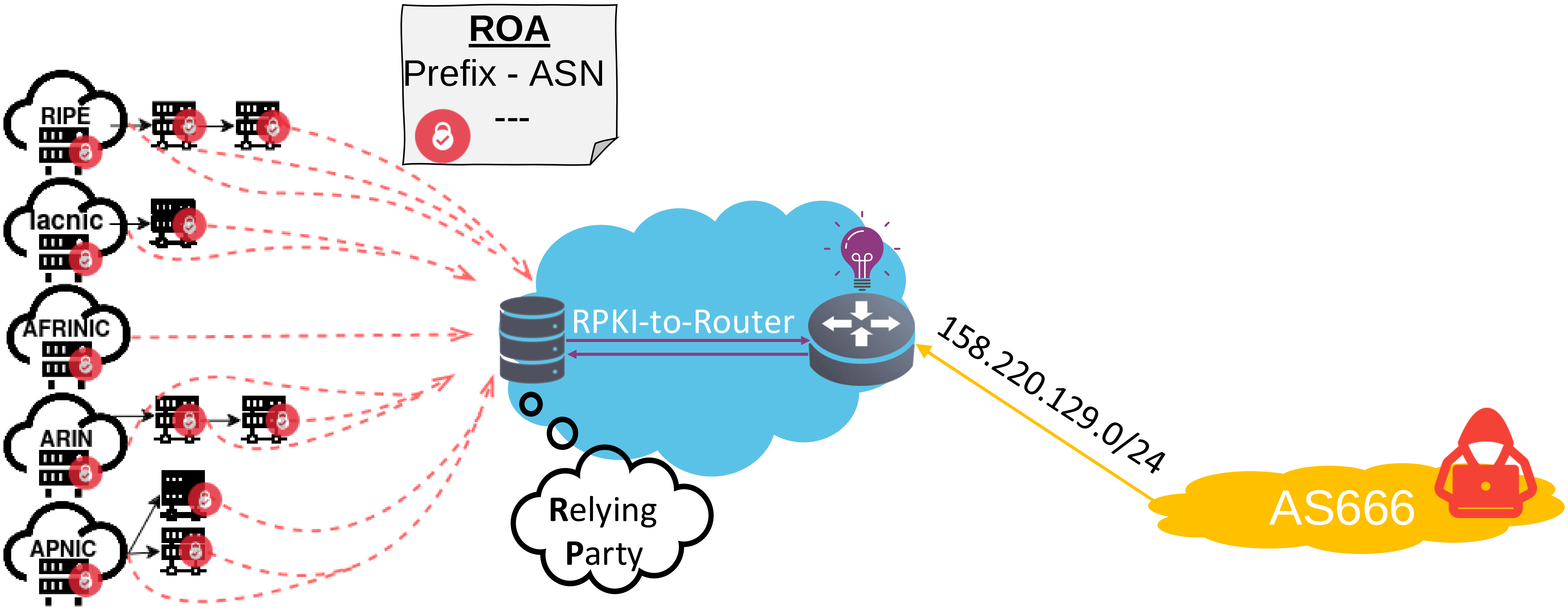


BGP Security with RPKI



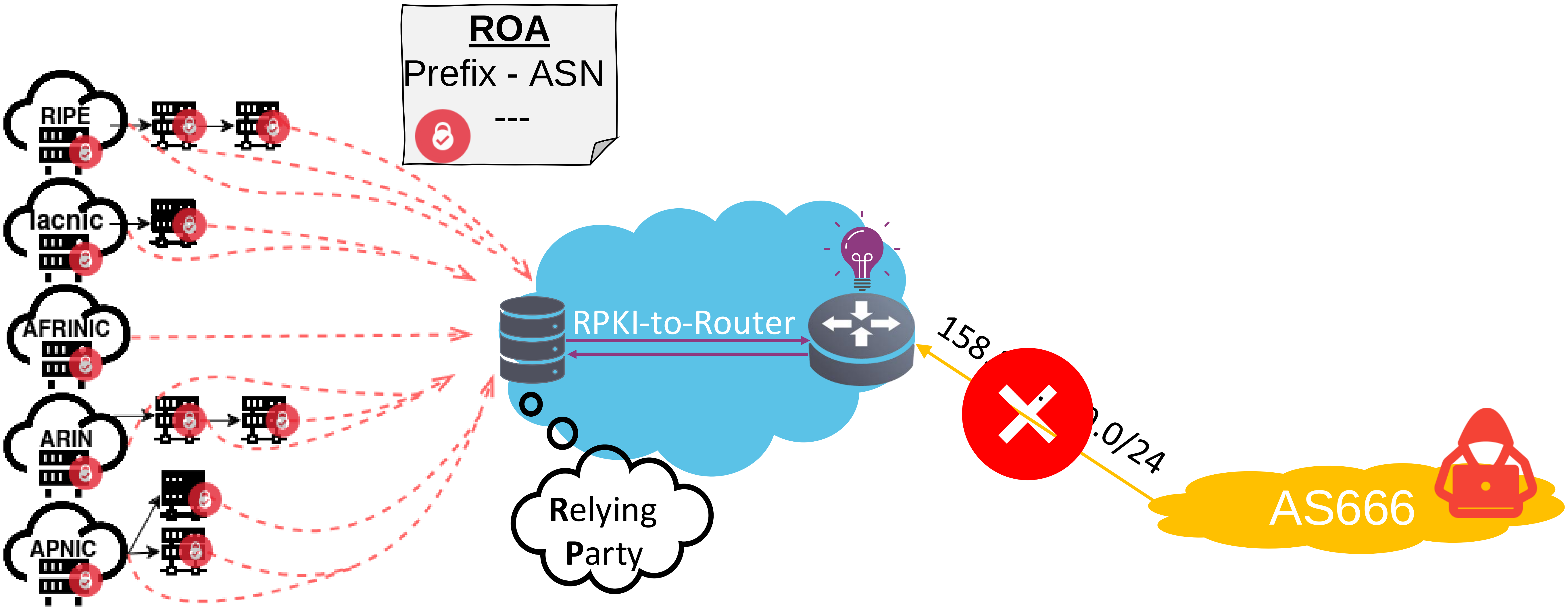
RPKI Repositories

BGP Security with RPKI



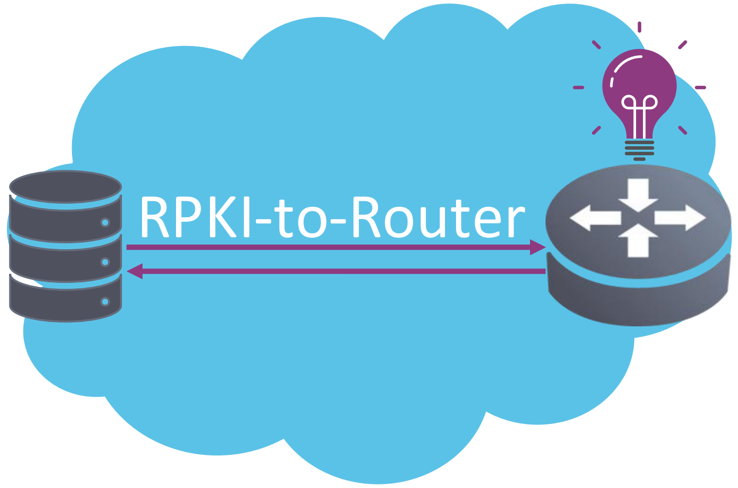
RPKI Repositories

BGP Security with RPKI

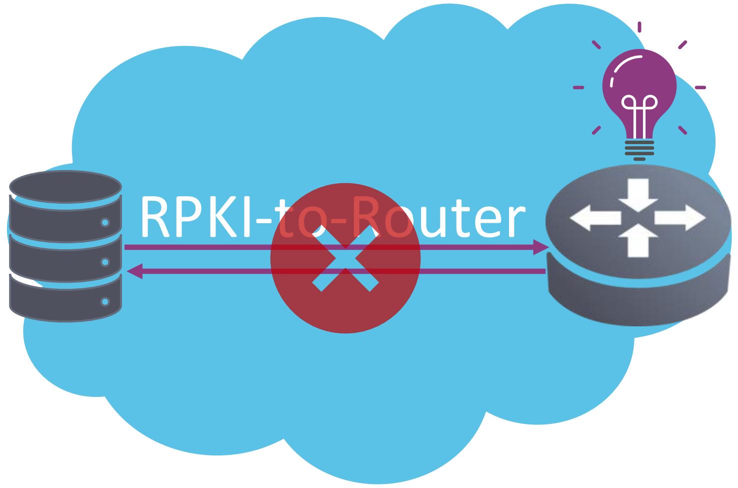


RPKI Repositories

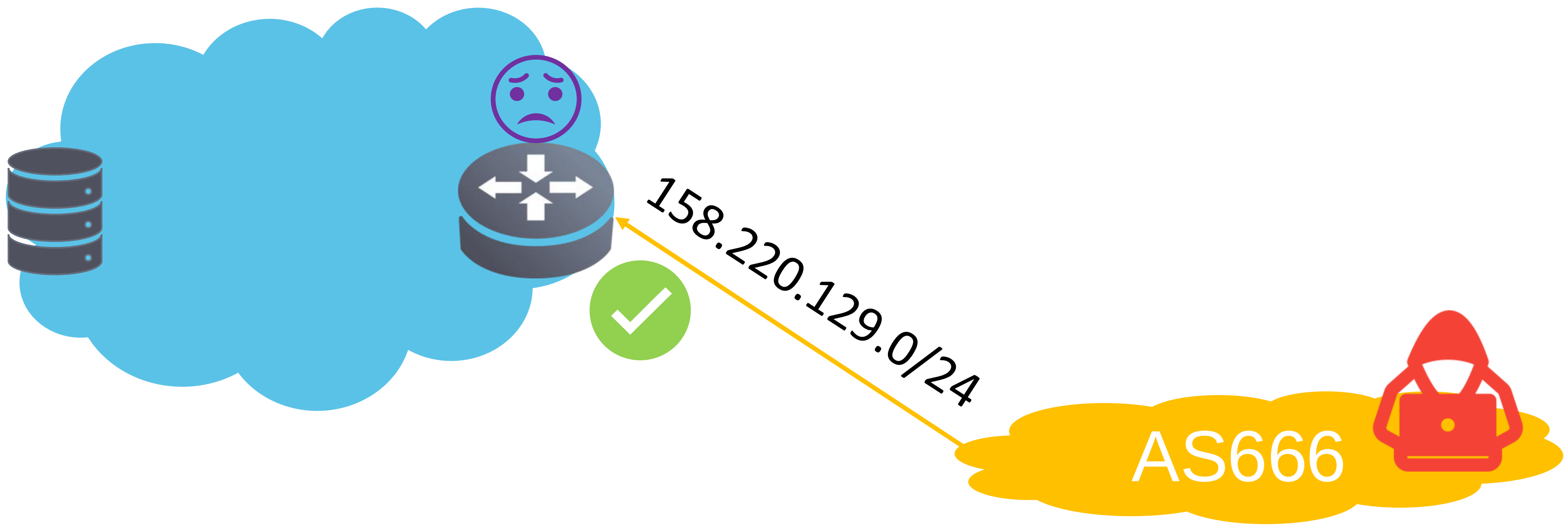
Why is DoS-ing RPs a big deal?



Why is DoS-ing RPs a big deal?



Why is DoS-ing RPs a big deal?



So we decided to tinker with the protocol…

So we decided to tinker with the protocol…

- Relaying Party Impl. 1: crash when objects malformed

```
        Self::_create(data, &mut target).map_err(|err| {
            error!(
                "Fatal: failed to write file {}: {}", path.display(), err
            );
            Failed
        })
```

So we decided to tinker with the protocol…

- Relaying Party Impl. 1: crash when objects malformed

```
        Self::_create(data, &mut target).map_err(|err| {
            error!(
                "Fatal: failed to write file {}: {}", path.display(), err
            );
            Failed
        })
```

- Relying Party Impl. 2: crash when index out-of-bounds

```
    if iterationsUntilStable > *MaxIterations {
        log.Fatal("Max iterations has been reached.
        This number can be adjusted with -max.iterations")
    }
```

So we decided to tinker with the protocol...

- Relaying Party Impl. 1: crash when objects malformed

```
Self::_create(data, &mut target).map_err(|err| {
```

=> 84.9% of global Relying Party deployments affected by low-cost low-burden RPKI Downgrade Attacks

- Relying Party Impl. 2: crash when index out-of-bounds

```
if iterationsUntilStable > *MaxIterations {
    log.Fatal("Max iterations has been reached.
    This number can be adjusted with -max.iterations")
}
```

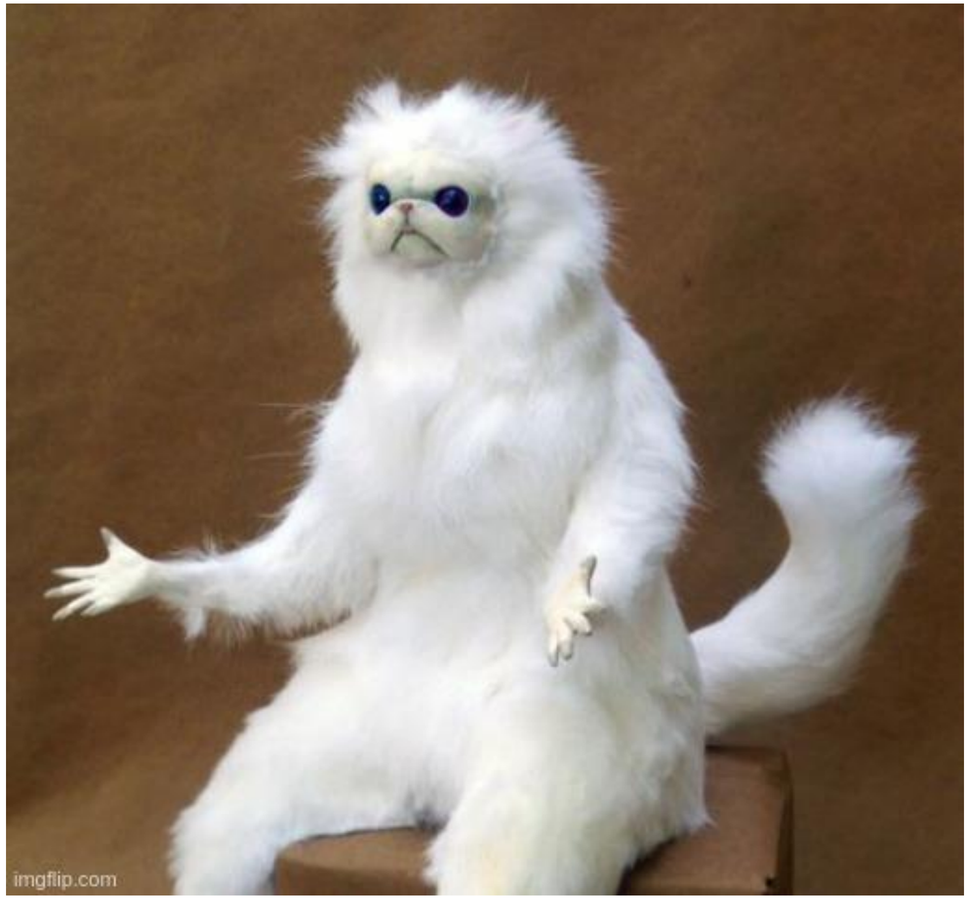
Towards a systematic approach

- RP is interesting target, but how do we test it?
- Fuzzing is a promising solution for systematic testing
- Simple idea:
  - Run many random inputs against RP
  - Find vulnerabilities 💥
  - **Profit** (*optional*)

Towards a systematic approach

- RP is interesting target, but how do we test it?
- Fuzzing is a promising solution for systematic testing

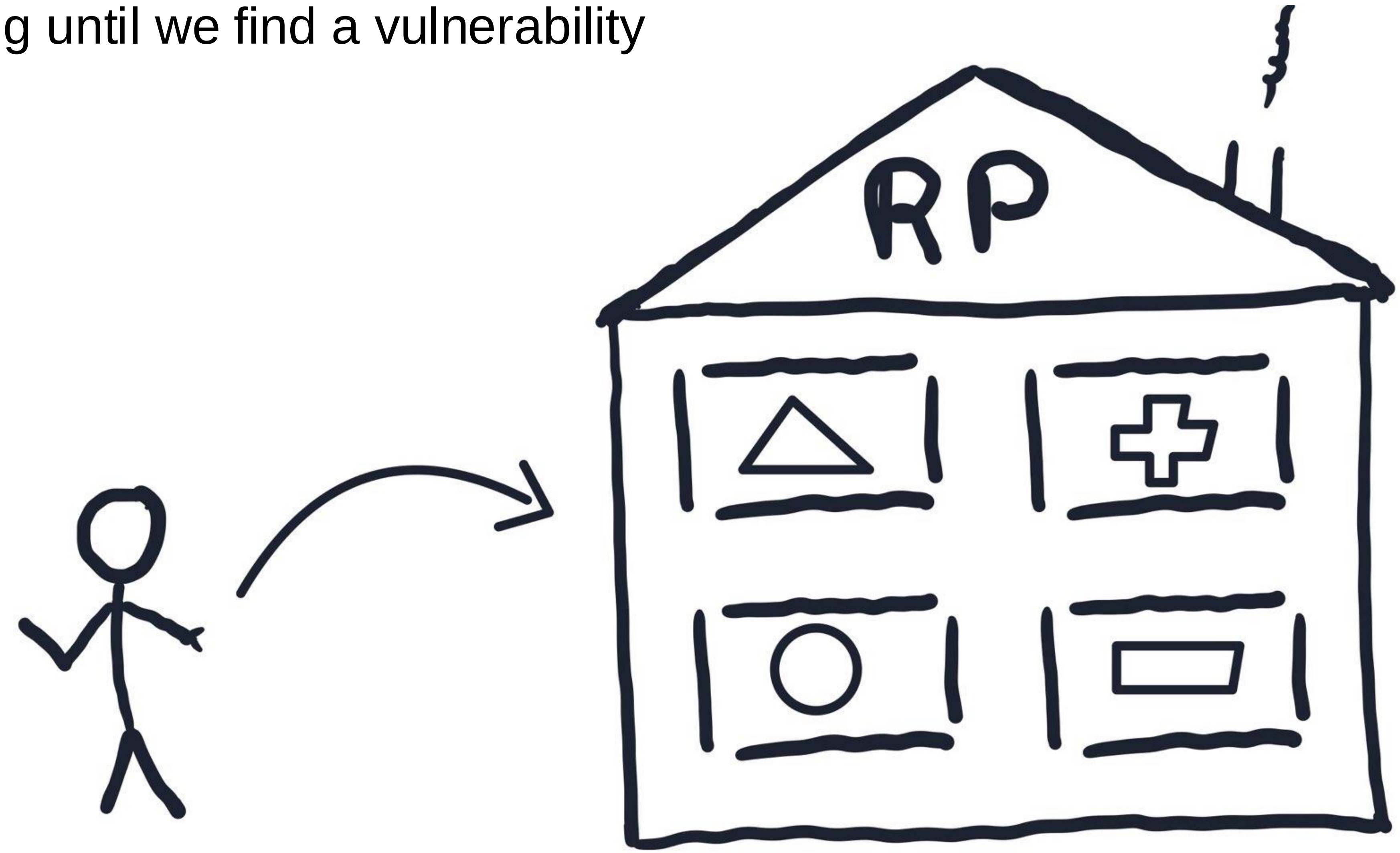
- Simple idea:
  - Run many random inputs against RP
  - Find vulnerabilities 💥
  - **Profit** (*optional*)



If it's so easy, why has nobody done it.... ????

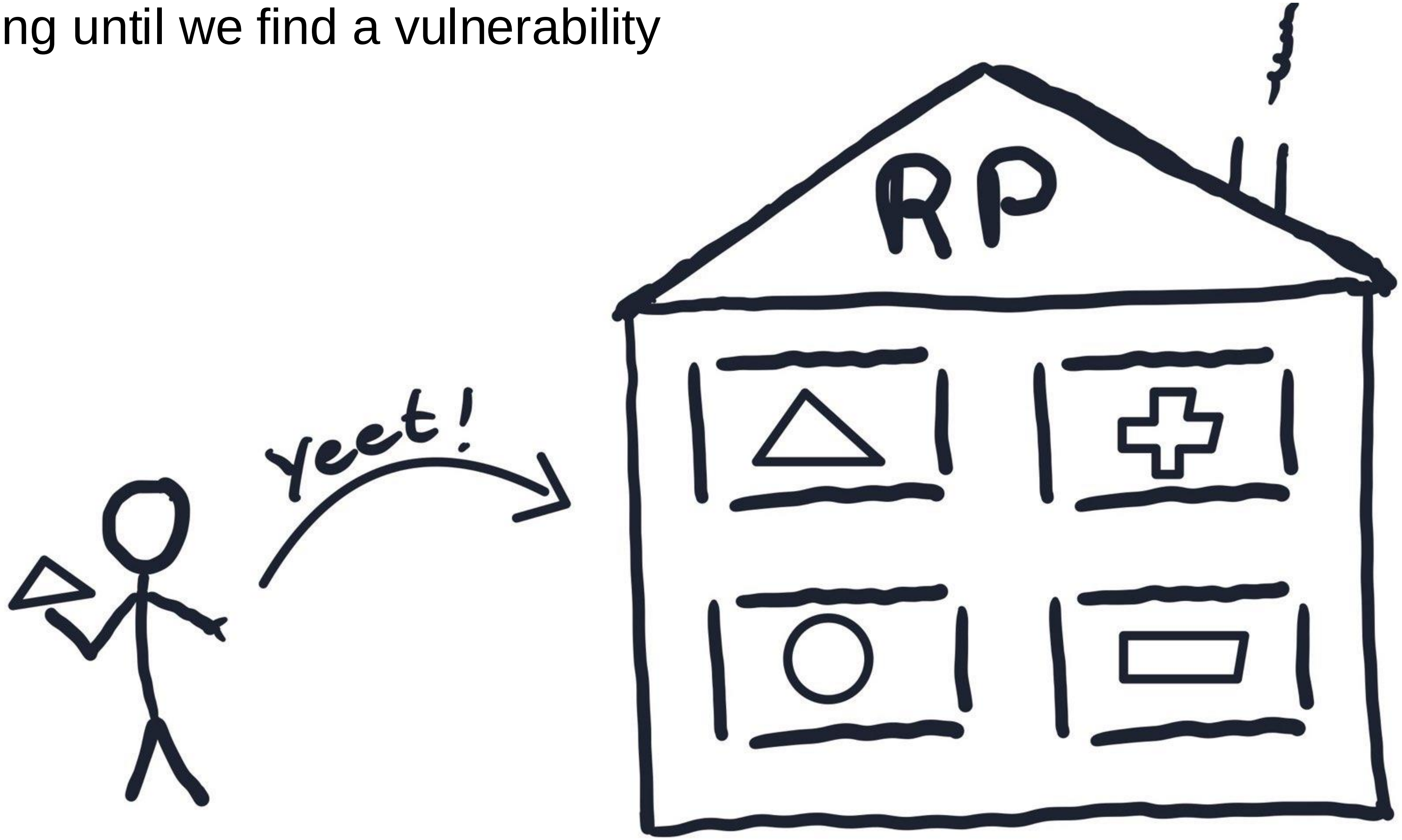
Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability



Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability



Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

Our simple Plan

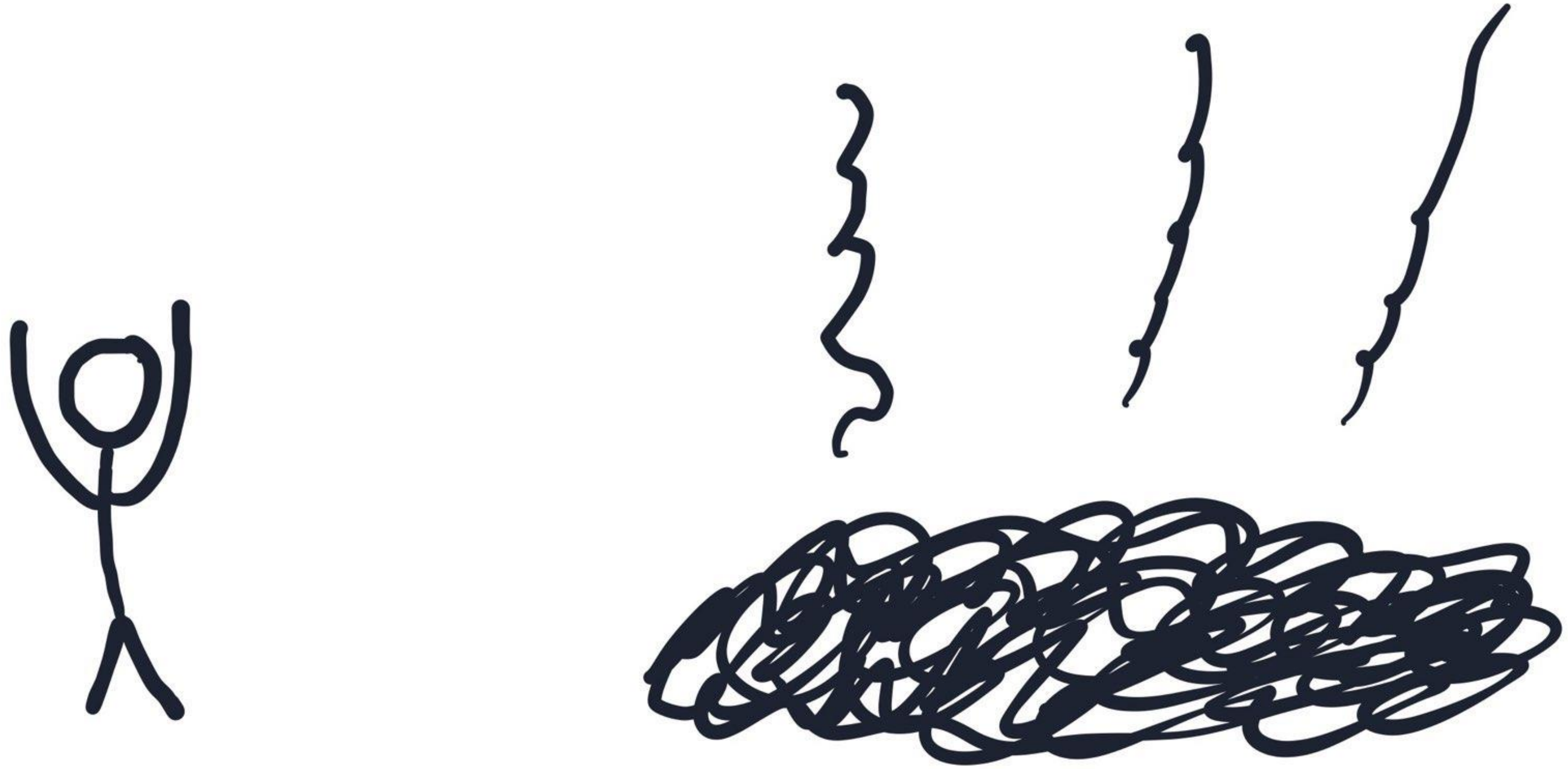
- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

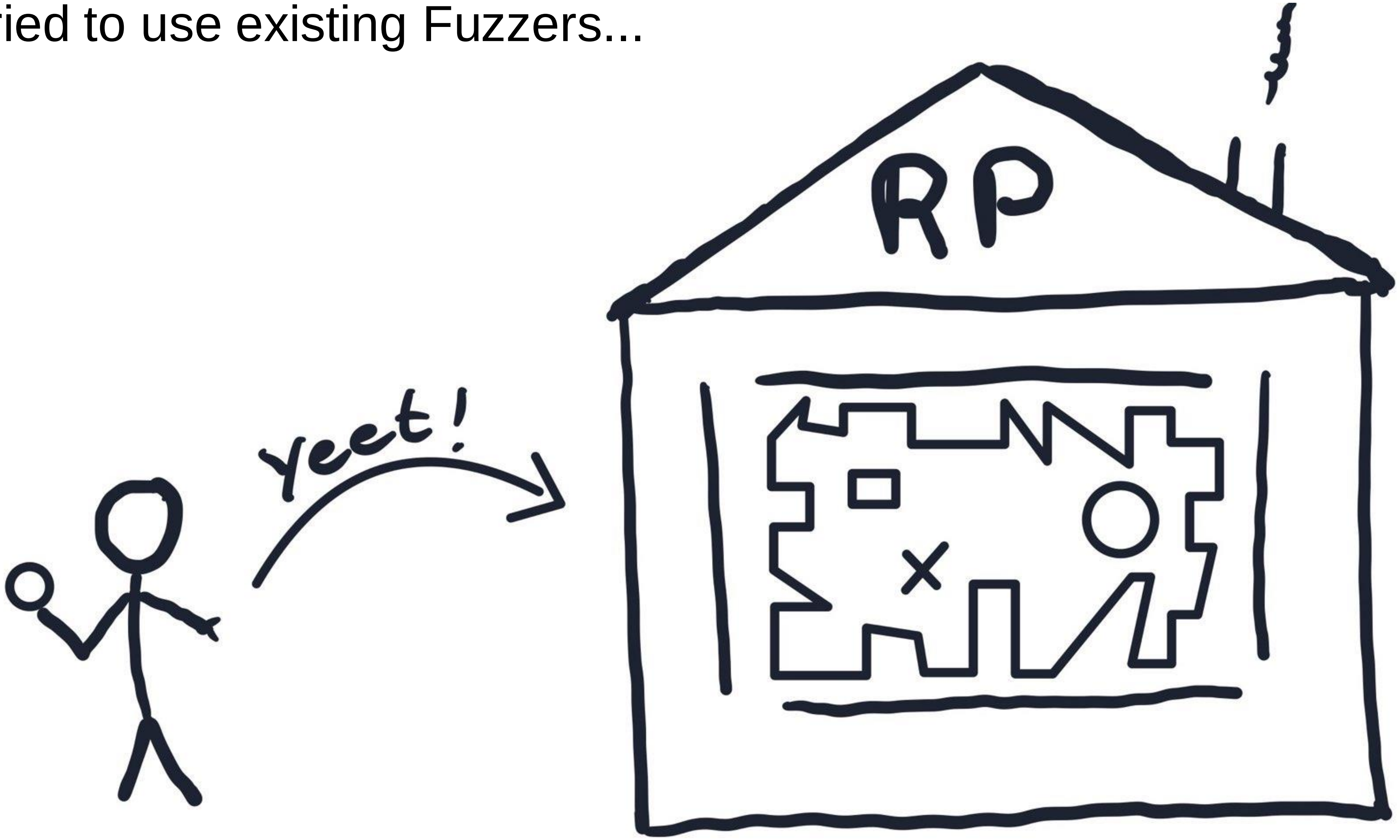


Our simple Plan

- Use existing Fuzzer, generate inputs, find crashes
- Keep trying until we find a vulnerability

The complex Reality

- RPs require very complex inputs
- We still tried to use existing Fuzzers...

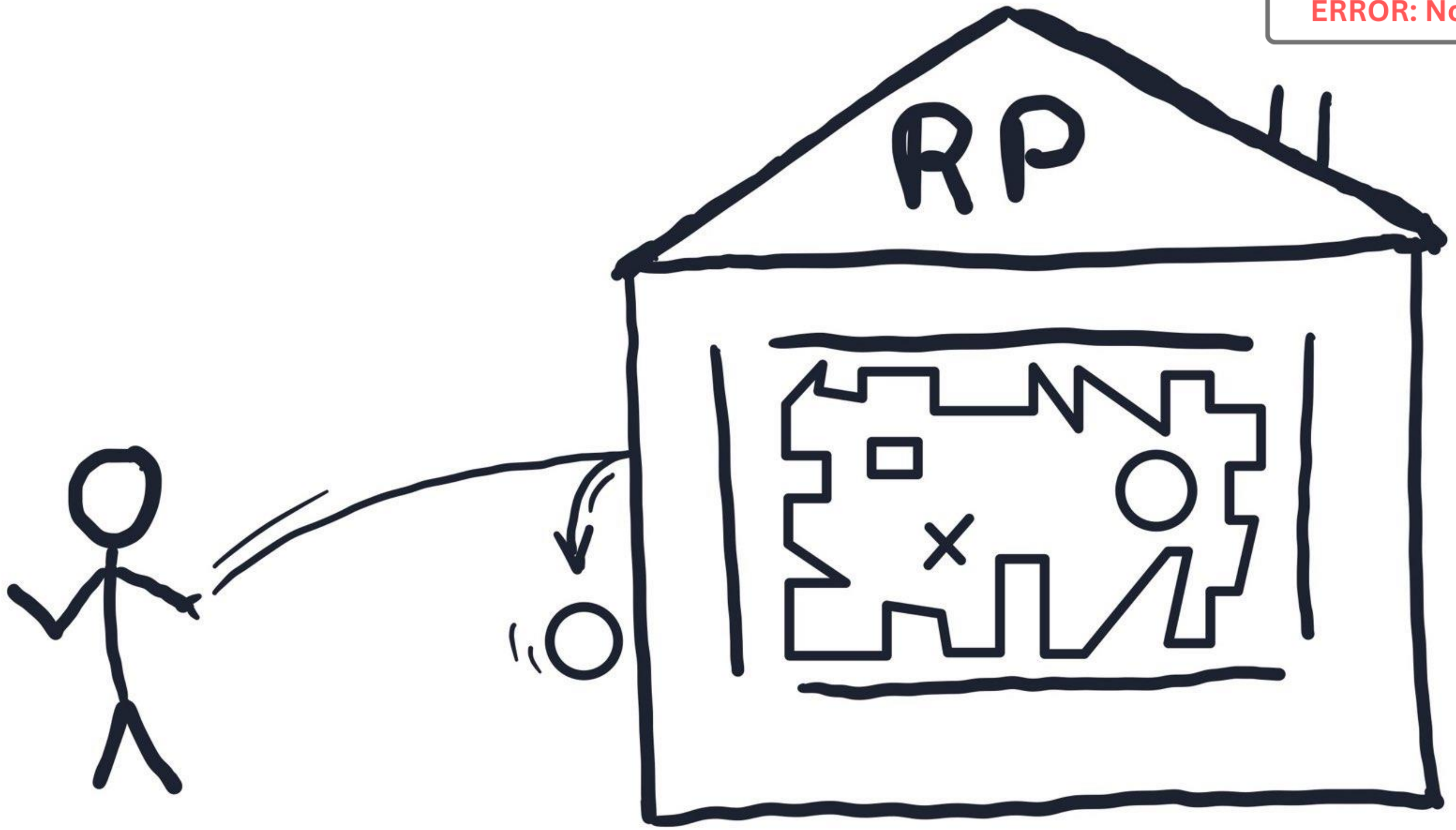


The complex Reality

- RPs require very complex inputs
- We still tried to use existing Fuzzers...

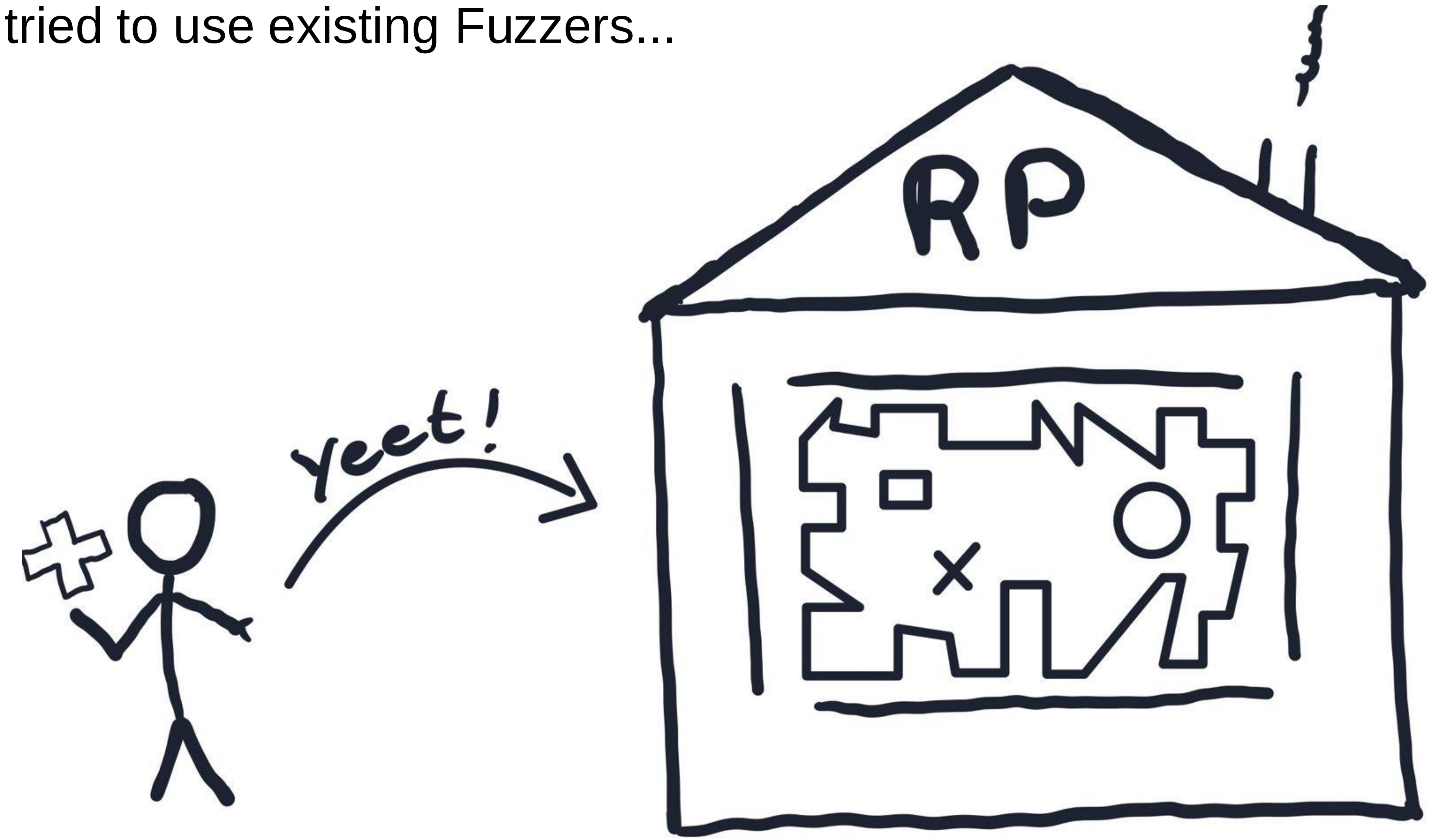
Routinator.log

ERROR: Failed to decode Manifest (my_ca.mft)
ERROR: No valid Manifest found (failed)



The complex Reality

- RPs require very complex inputs
- We still tried to use existing Fuzzers...

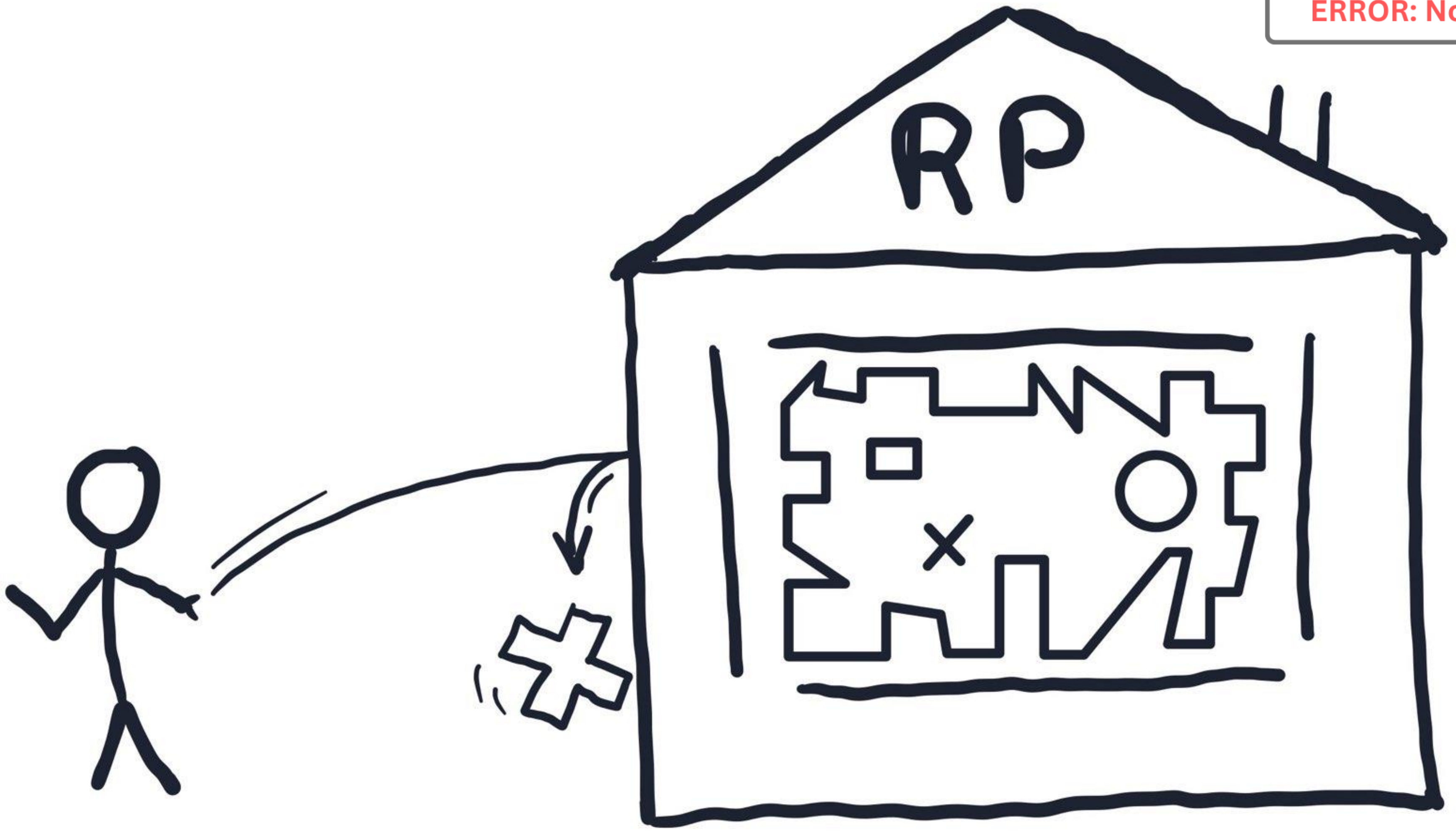


The complex Reality

- RPs require very complex inputs
- We still tried to use existing Fuzzers...

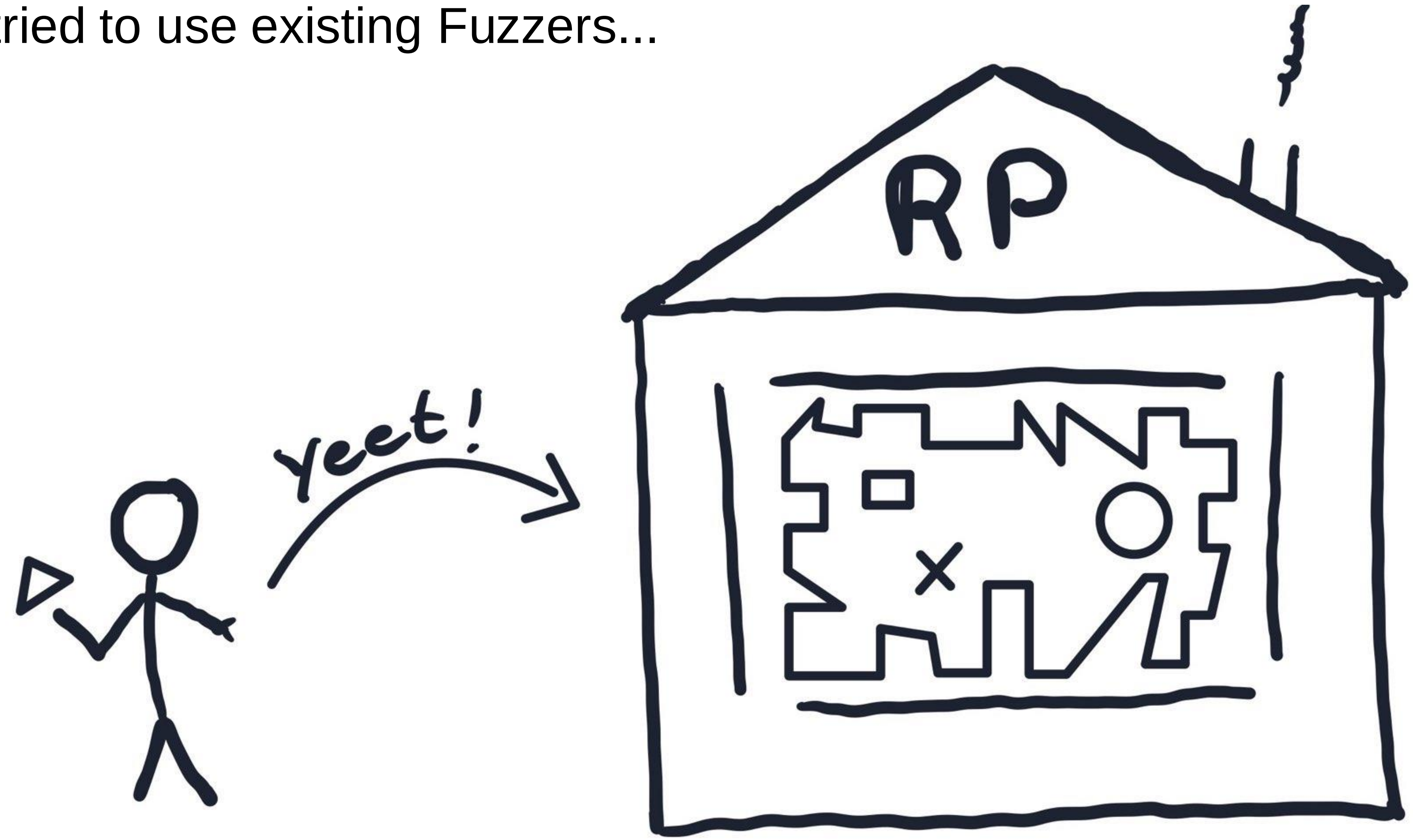
Routinator.log

ERROR: Failed to decode Manifest (my_ca.mft)
ERROR: No valid Manifest found (failed)



The complex Reality

- RPs require very complex inputs
- We still tried to use existing Fuzzers...

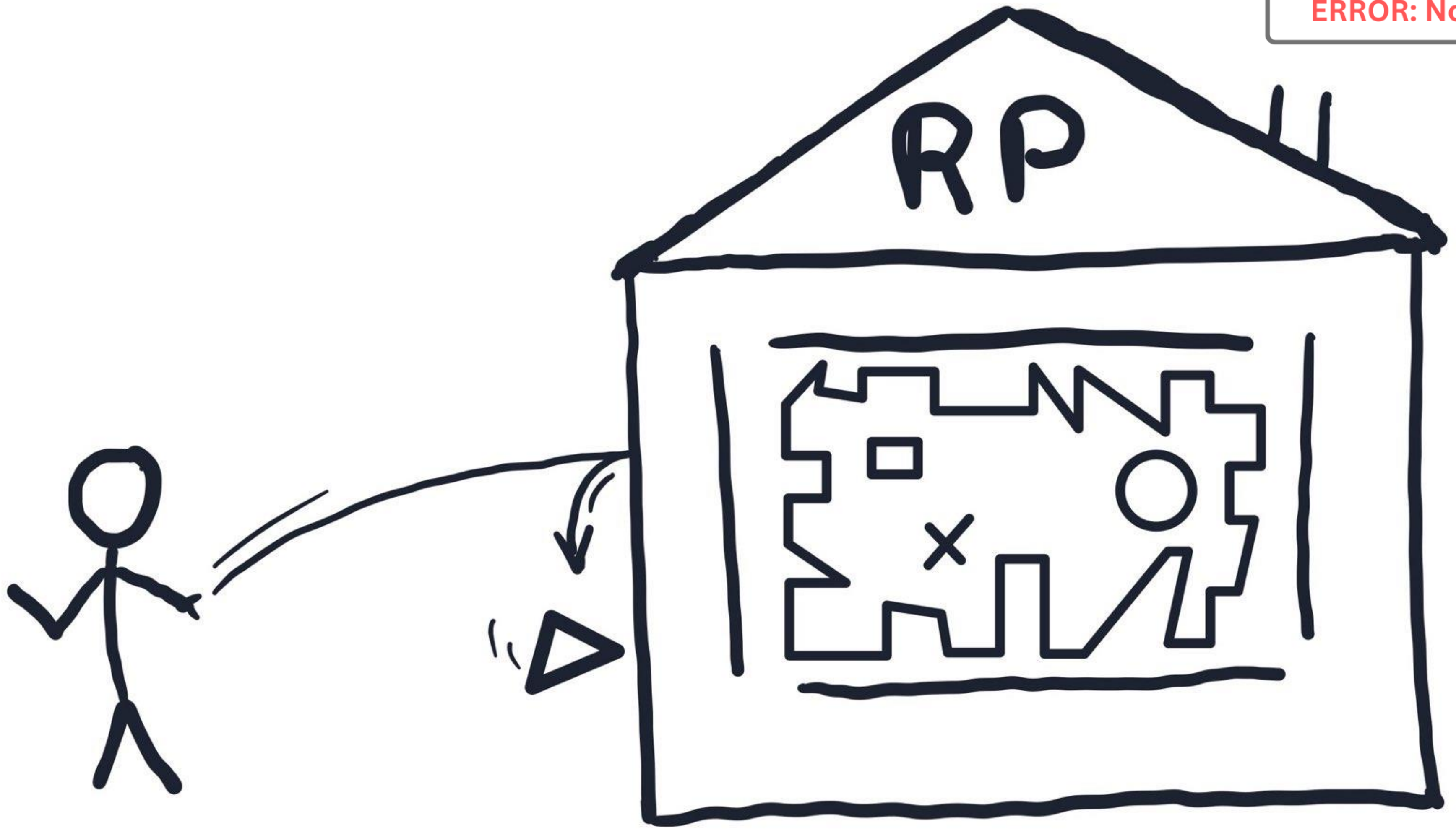


The complex Reality

- RPs require very complex inputs
- We still tried to use existing Fuzzers...

Routinator.log

ERROR: Failed to decode Manifest (my_ca.mft)
ERROR: No valid Manifest found (failed)

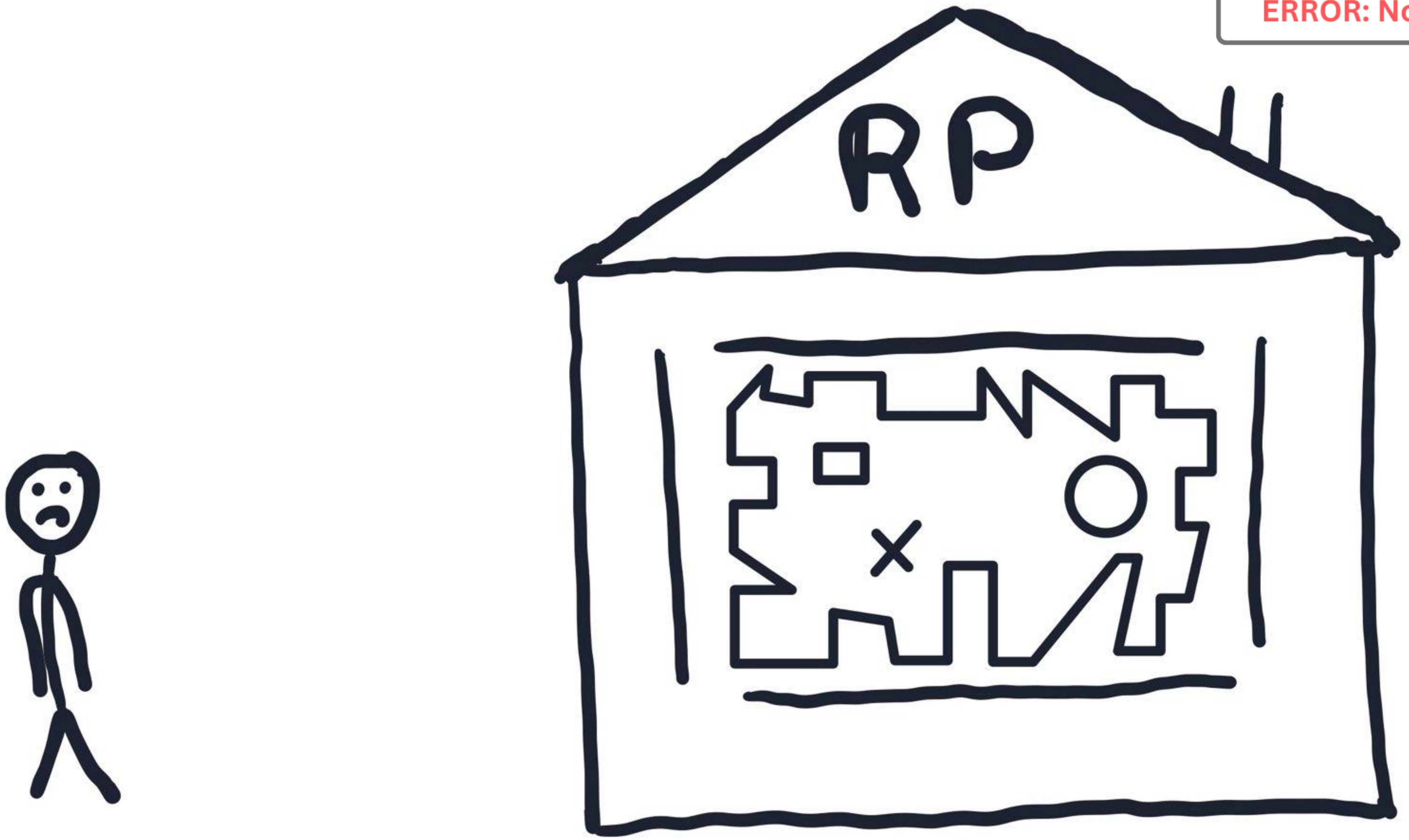


The complex Reality

- RPs require very complex inputs
- We still tried to use existing Fuzzers...

Routinator.log

ERROR: Failed to decode Manifest (my_ca.mft)
ERROR: No valid Manifest found (failed)

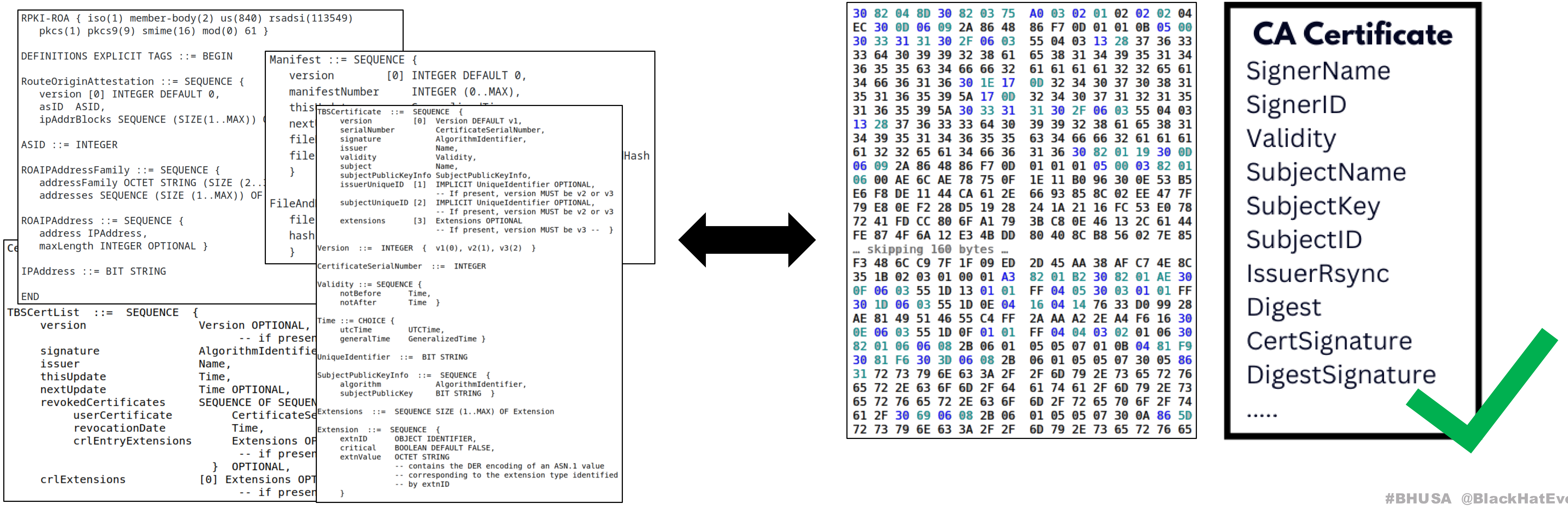


Why is this so difficult

- RPKI objects are complex (ASN.1 / X.509 formats)
- Fuzzers struggle with complex objects

Why is this so difficult

- RPKI objects are complex (ASN.1 / X.509 formats)
- Fuzzers struggle with complex objects



It gets worse...

- RPKI uses...

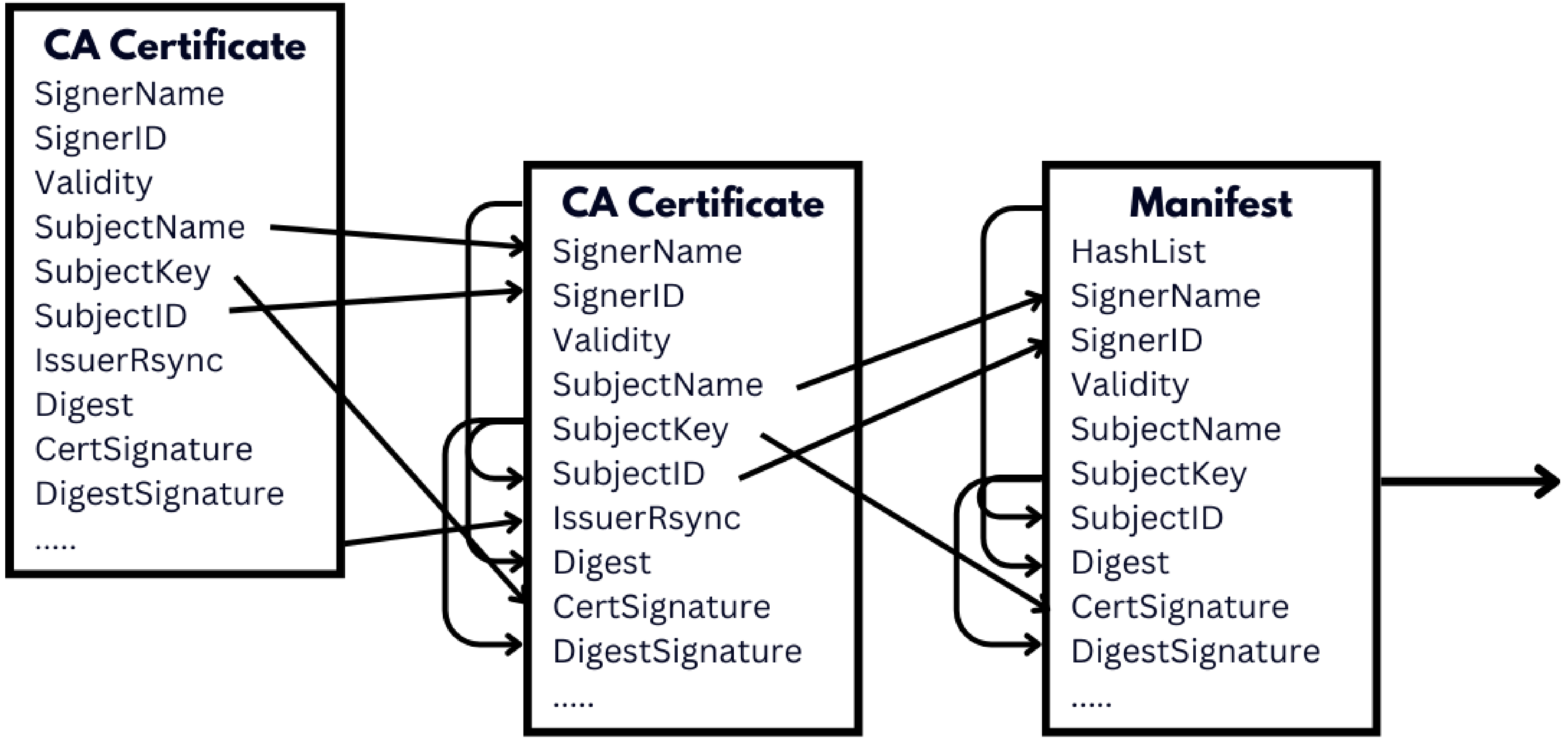
It gets worse...

- RPKI uses...

CRYPTOGRAPHY

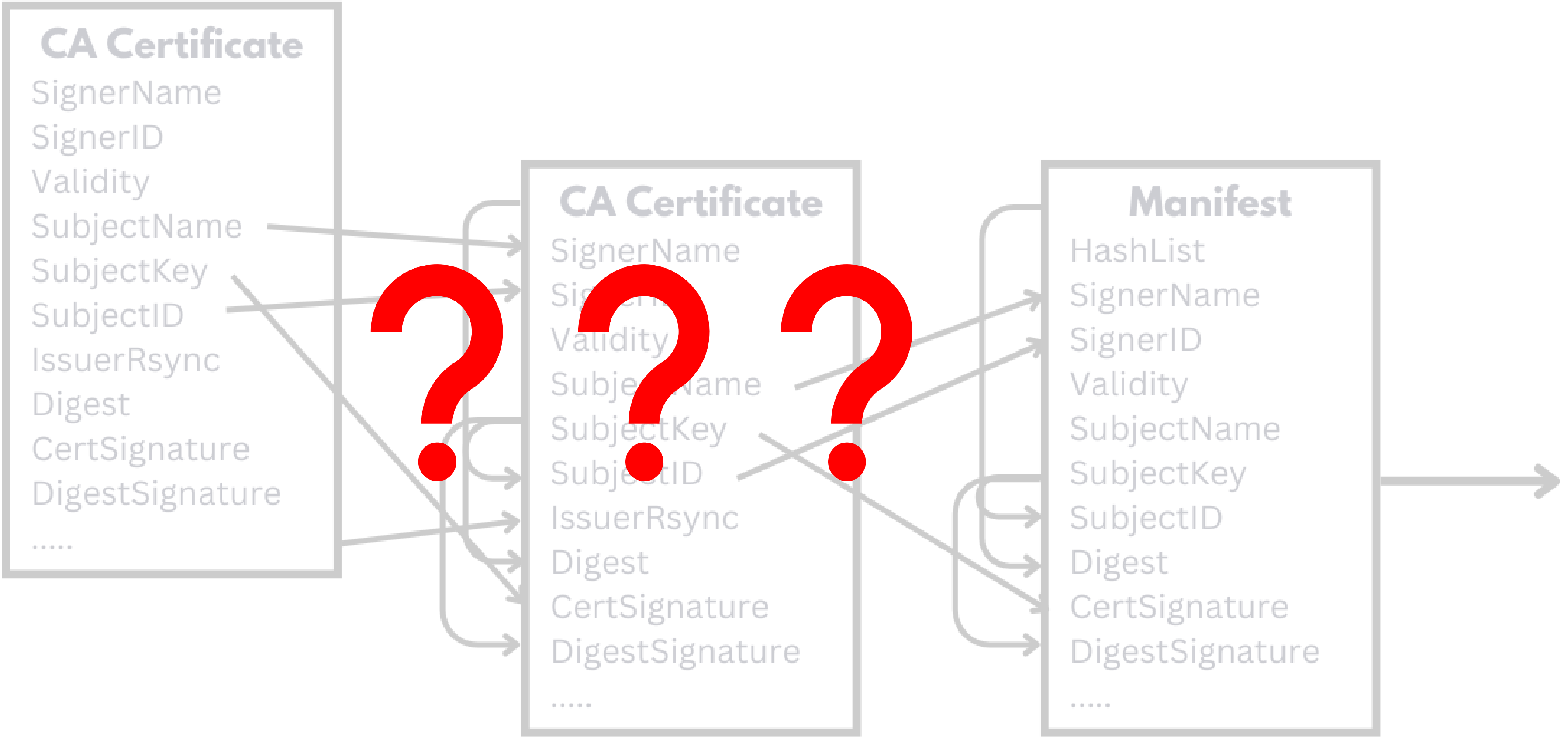
It gets worse…

- RPKI uses cryptography
- Fuzzers struggle with cryptography



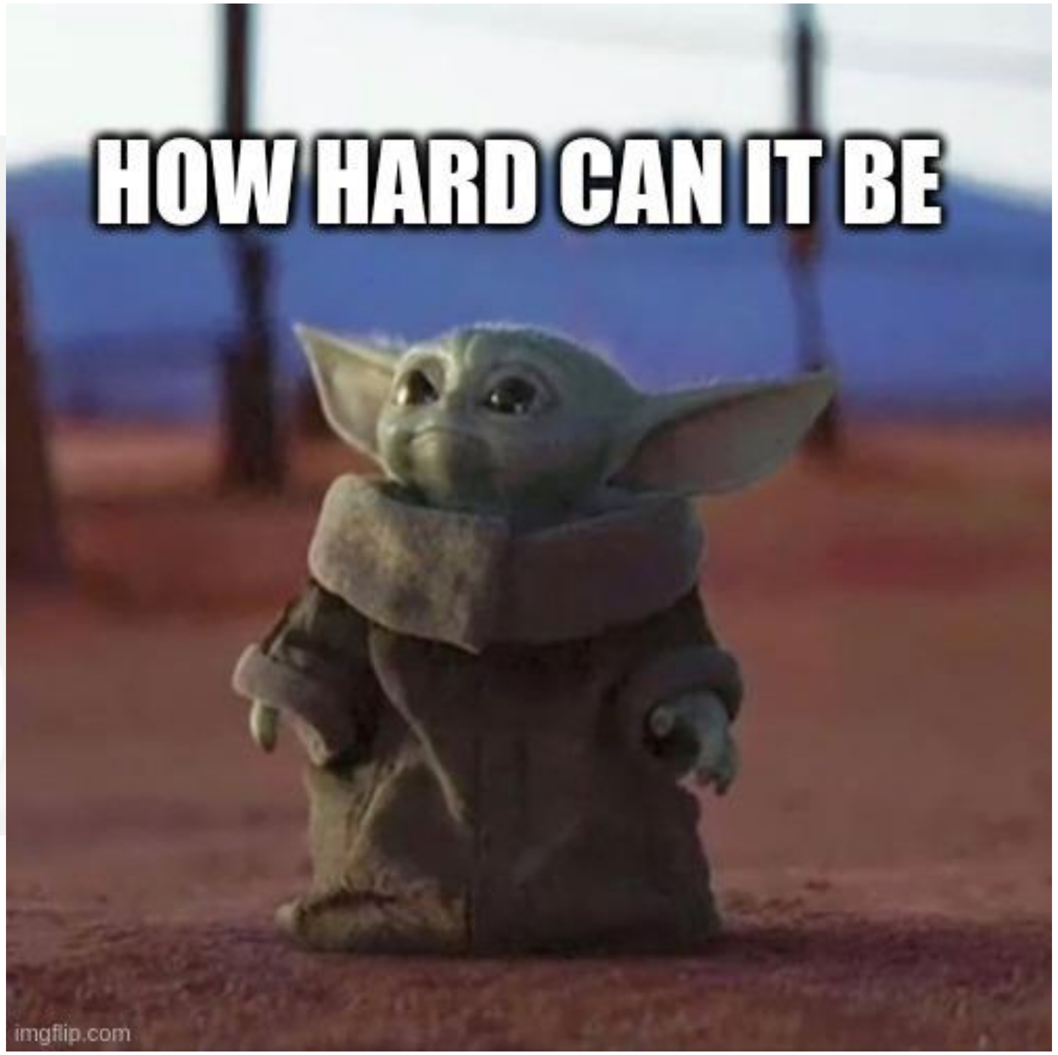
It gets worse...

- RPKI uses cryptography
- Fuzzers struggle with cryptography

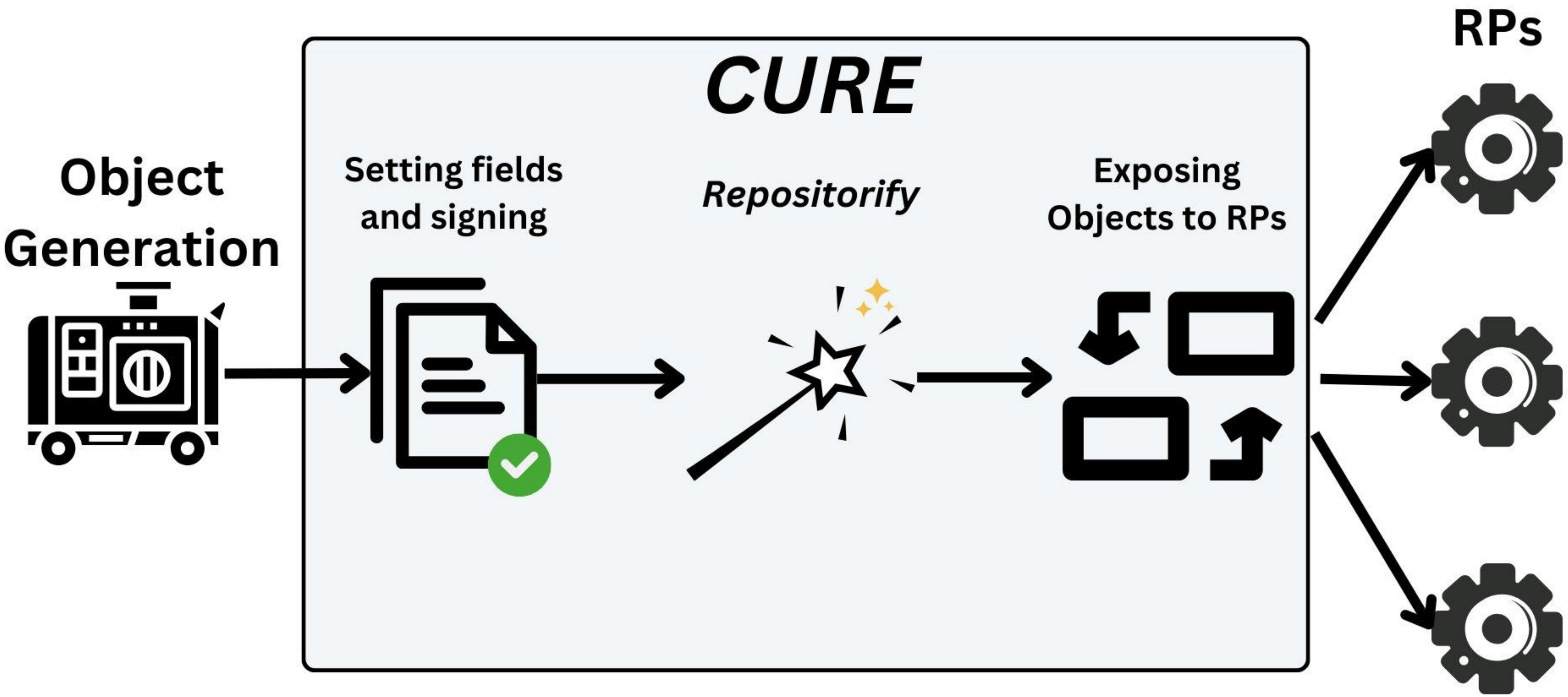


Only one solution...

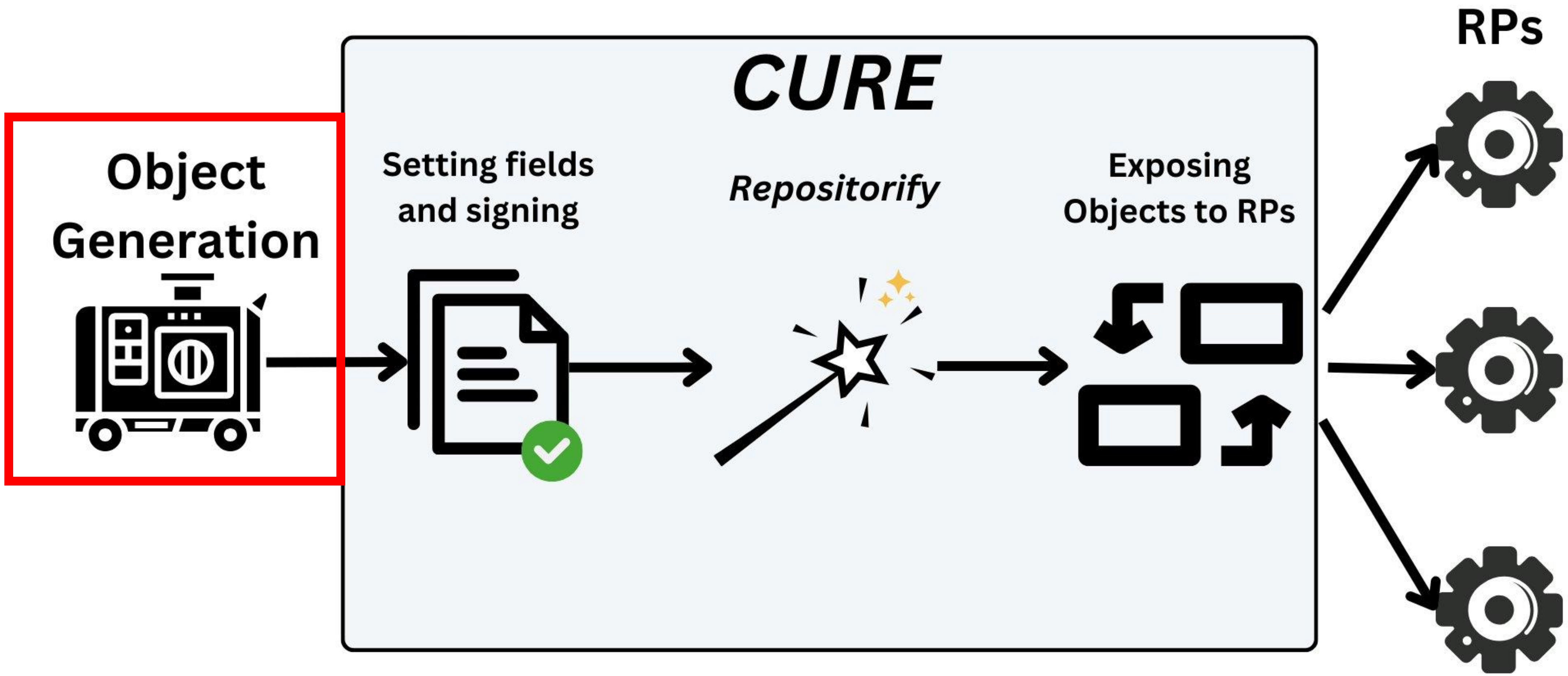
Only one solution...



Building yet another Fuzzer

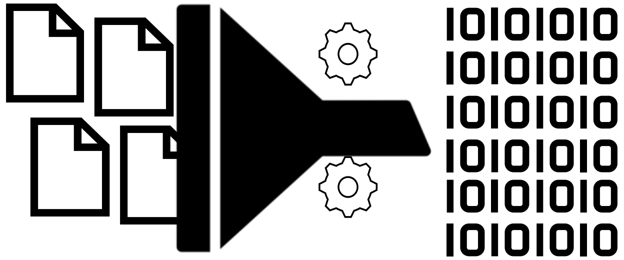


Building yet another Fuzzer



Object Generation

1. Random Byte Mutation



i. feed the randomizer a set of valid objects
ii. splice files & generate random mutations
iii. targets programming, parsing & schematic errors

Object Generation

1. Random Byte Mutation

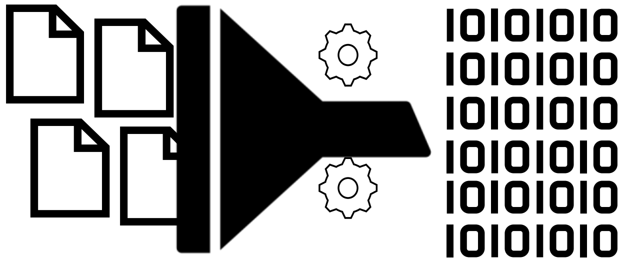
i. feed the randomizer a set of valid objects
ii. splice files & generate random mutations
iii. targets programming, parsing & schematic errors

2. Structure Aware Mutation

i. schema-abiding, correctly encoded objects
ii. manipulate content of fields
iii. targets processing and validation logic

Object Generation

1. Random Byte Mutation



i. feed the randomizer a set of valid objects
ii. splice files & generate random mutations
iii. targets programming, parsing & schematic errors

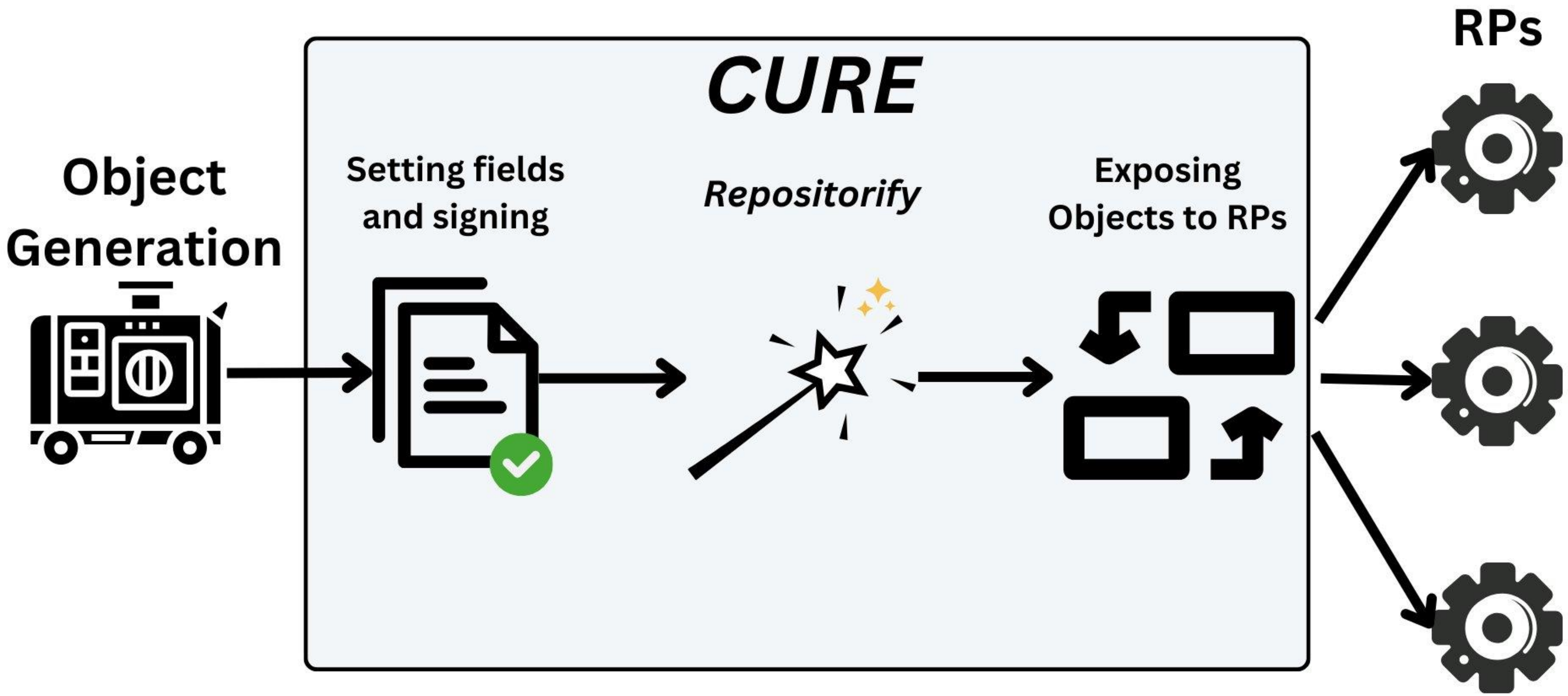
Found Bugs: 7

2. Structure Aware Mutation

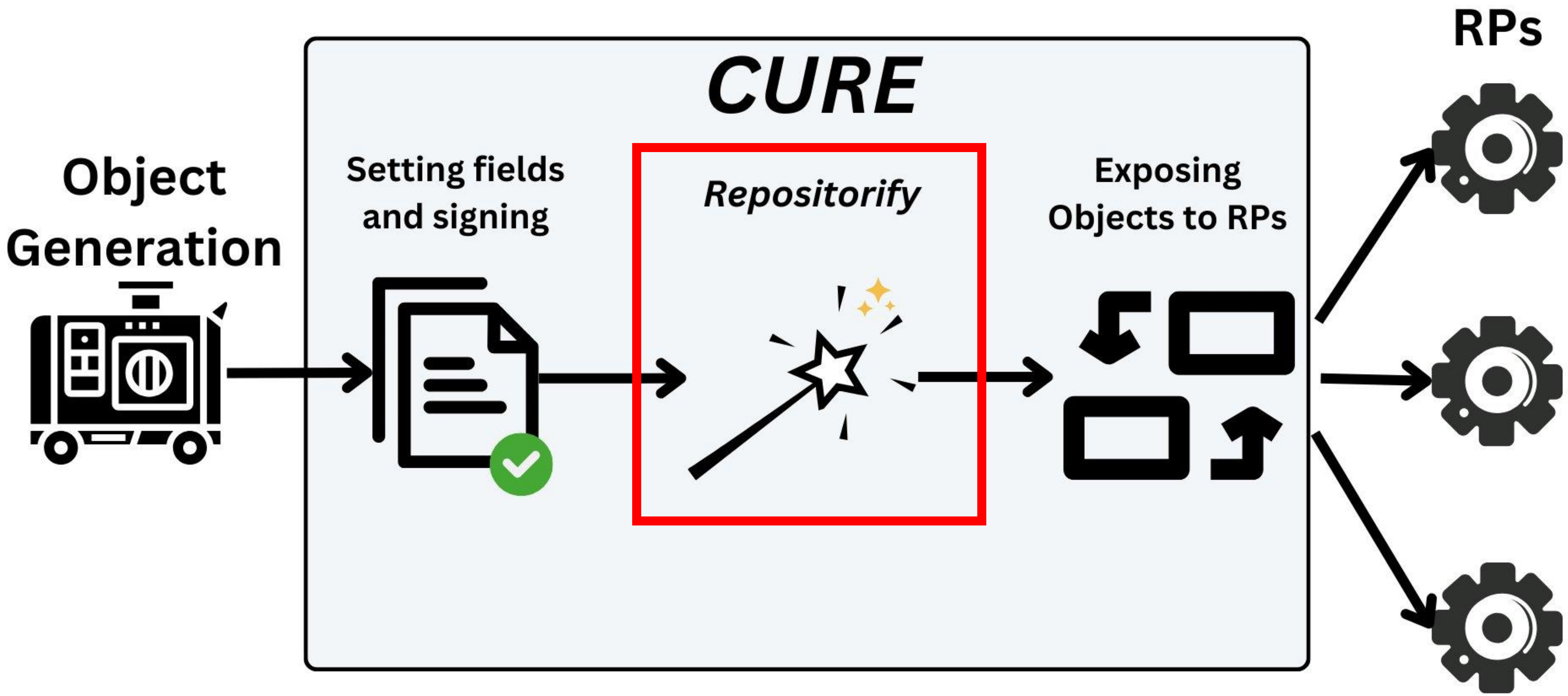
i. schema-abiding, correctly encoded objects
ii. manipulate content of fields
iii. targets processing and validation logic

Found Bugs: 11

Repositorify Module

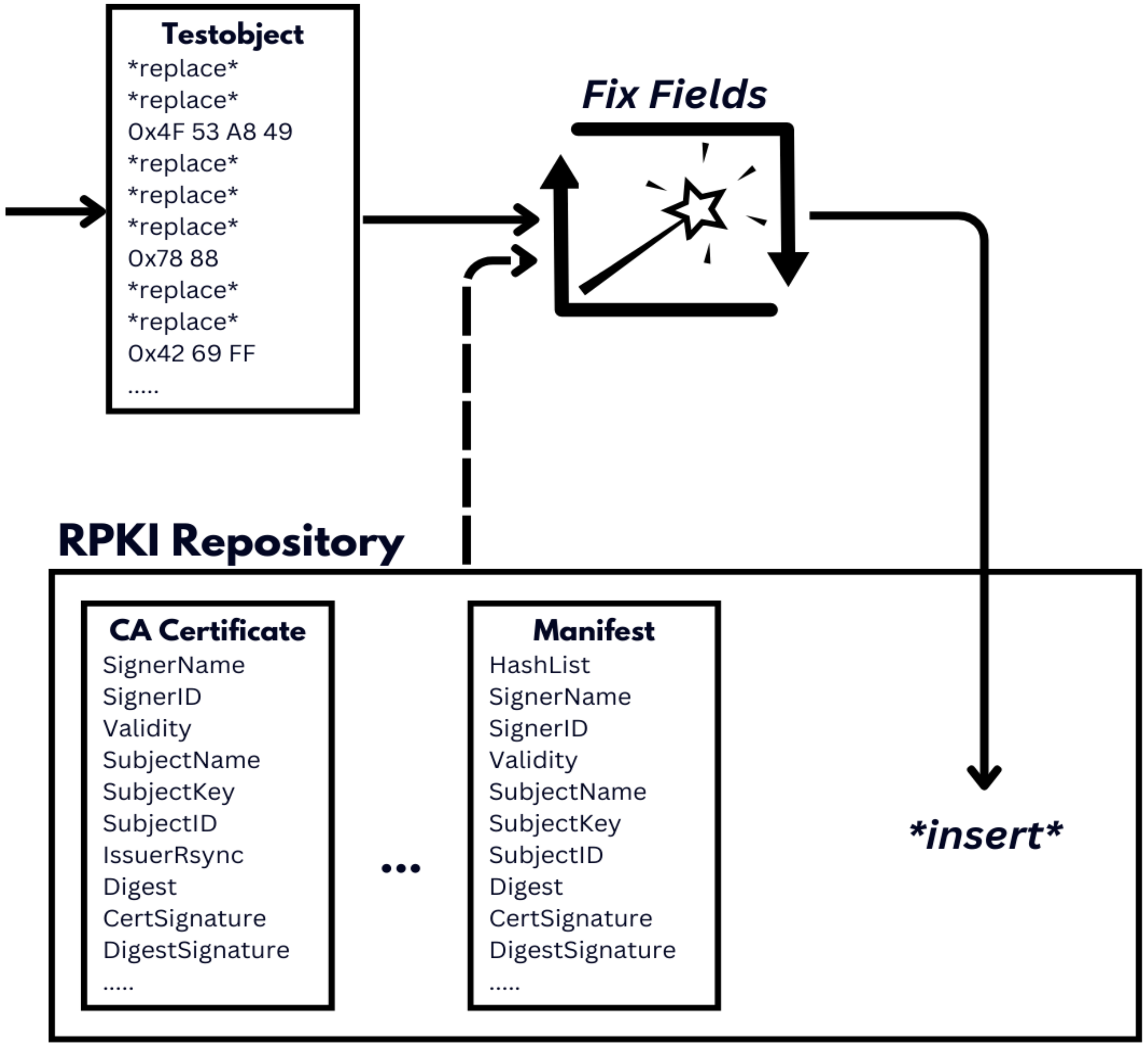


Repositorify Module



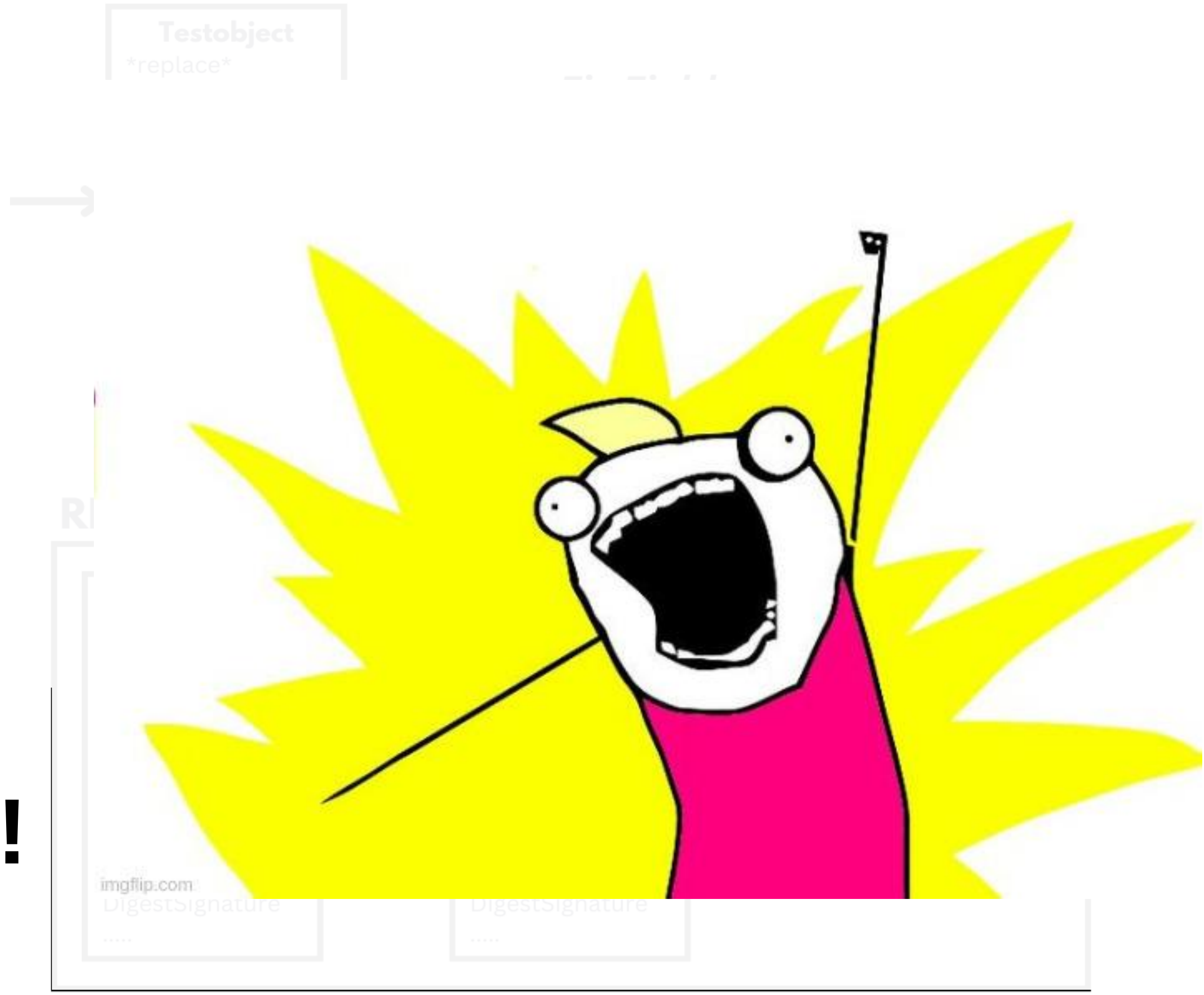
Repositorify Module

- ➢ Create valid RPKI repository
- ➢ Replace fields in objects
  E.g. compute signatures
- ➢ Insert Test-Objects into repository



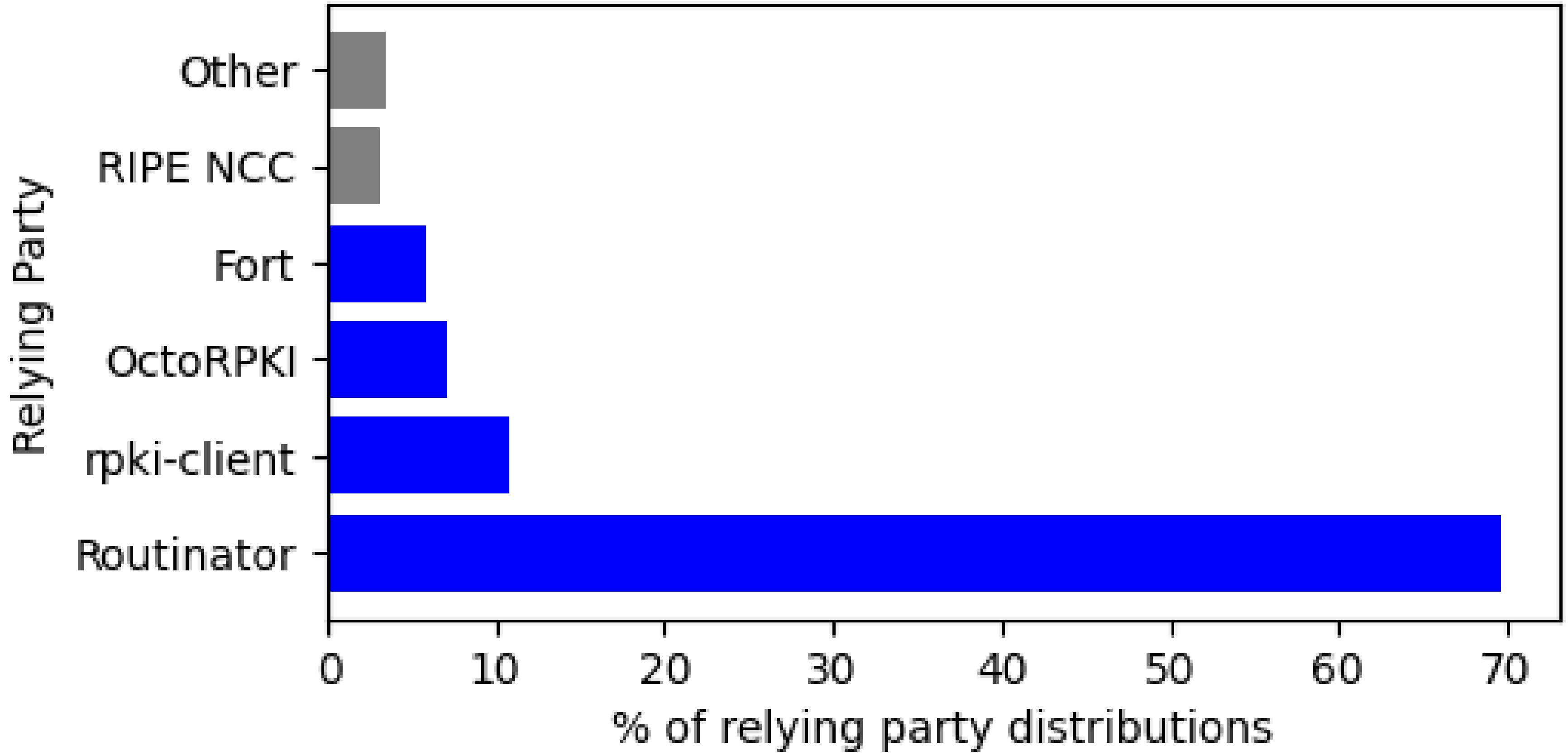
Repositorify Module

- Create valid RPKI repository
- Replace fields in objects
  E.g. compute signatures
- Insert Test-Objects into repository



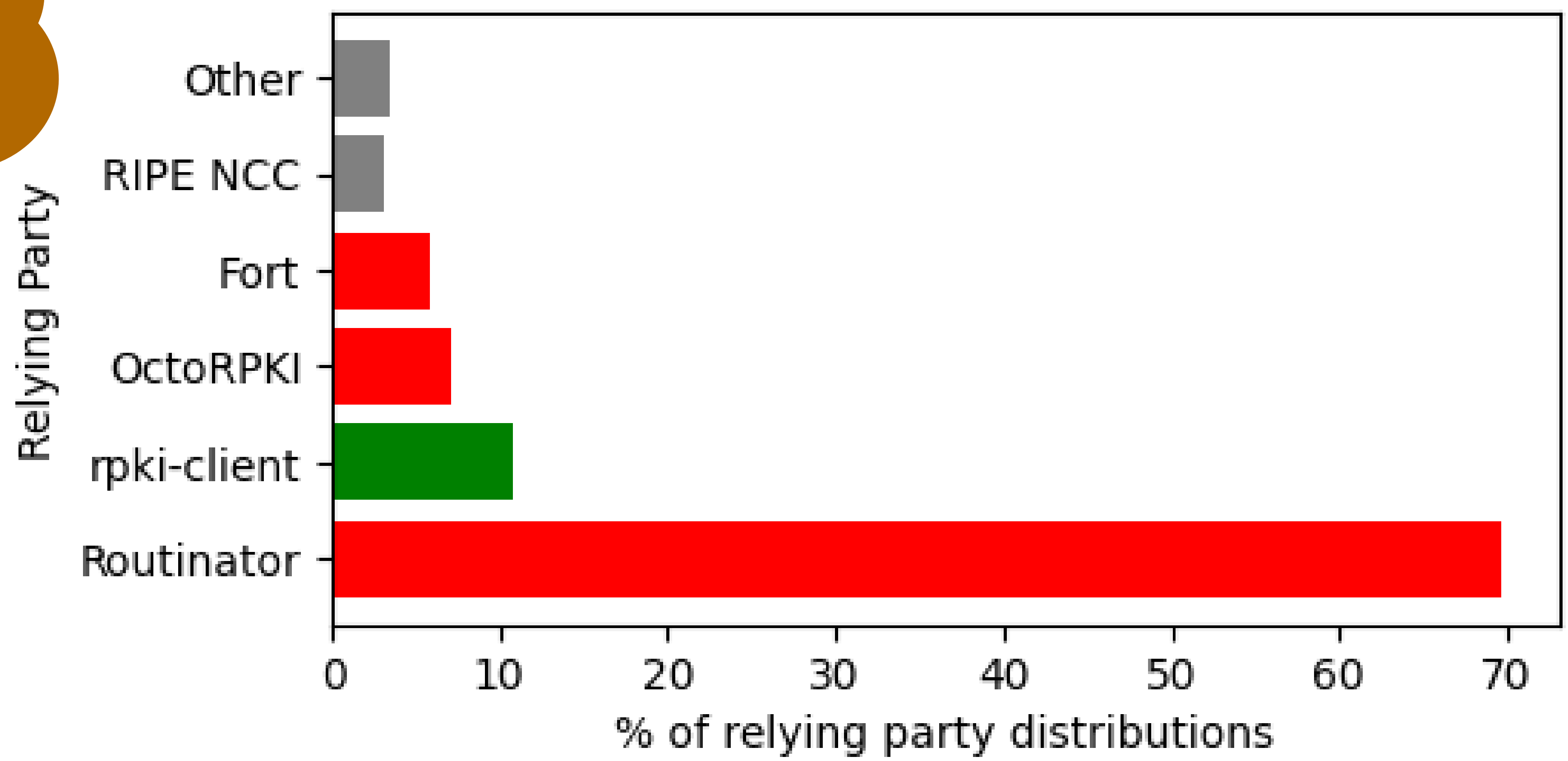
Let's find vulnerabilities!!

Relying Party Distributions



Summary of Results

We found issues on **3 out of 4** maintained RPs



18 total vulnerabilities & 5 CVEs

Vulnerability Type: Path Traversal

- Vulnerable Software: ***Routinator***

- Critical: 9.3 (CVE-2023-39916)

Vulnerability Type: Path Traversal

- Vulnerable Software: ***Routinator***
- Critical: 9.3 (CVE-2023-39916)
- Exploit:
  1. place malicious file anywhere on disk
  2. poison the RPKI data by adding a malicious root certificate pointer

Notification.xml


```
<notification [Header]>
 <snapshot
   uri="https://server.com/data/../../../fake.TAL"
   hash="33f969c5b6fd9ab501f9def2d47f7576ba80
         0a91d09d34a080ed2cf90a86d1ec"
/>
</notification>
```

Vulnerability Type: DoS

- Adversary can create objects of any format

Vulnerability Type: DoS

- Adversary can create objects of any format

- Vulnerable Software:
  - ***Routinator:*** Parsing of ASN.1 Data
  - ***OctoRPKI:*** Processing of Object Fields
  - ***Fort:*** Processing of RTR Requests

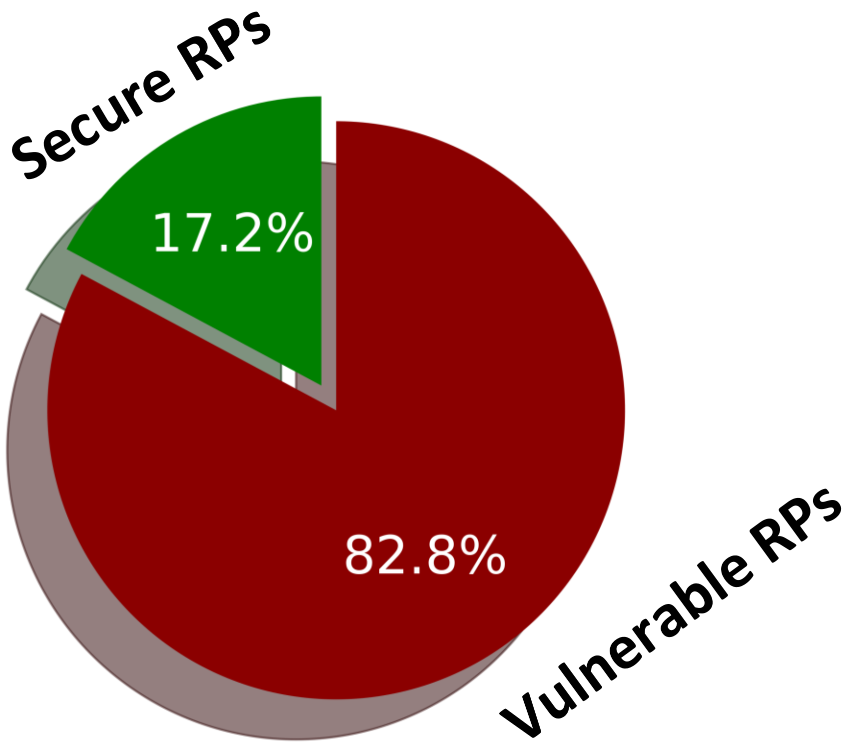
- Exploit:
  Adversary forces RPs in perpetual fail-and-restart mode

Routinator.log

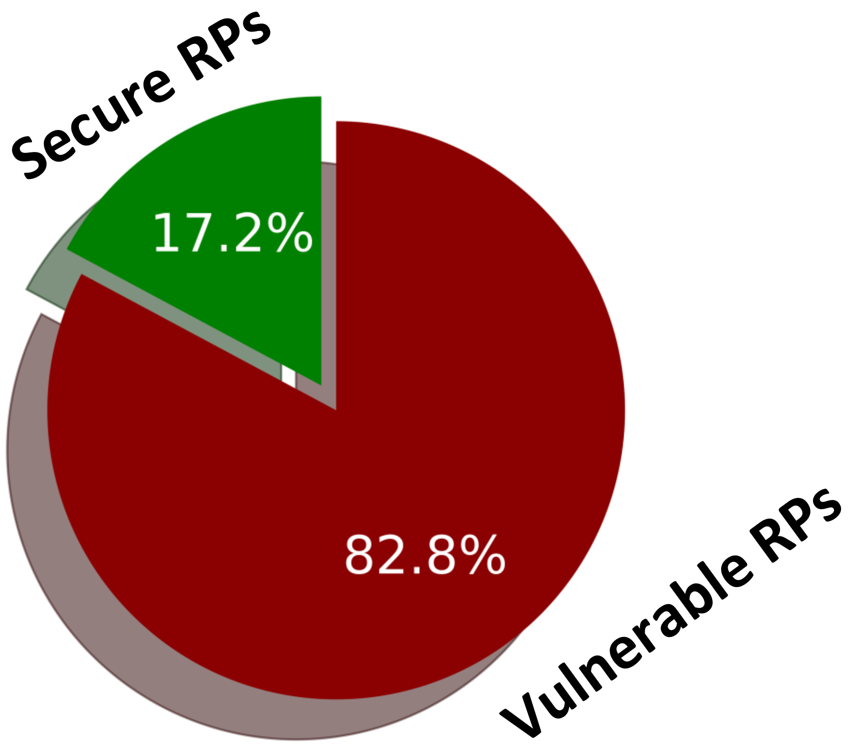
thread '<unnamed>' panicked at 'index out of bounds: the len is 2 but the index is 2',
bcder/src/tag.rs:line:column
note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
Aborted

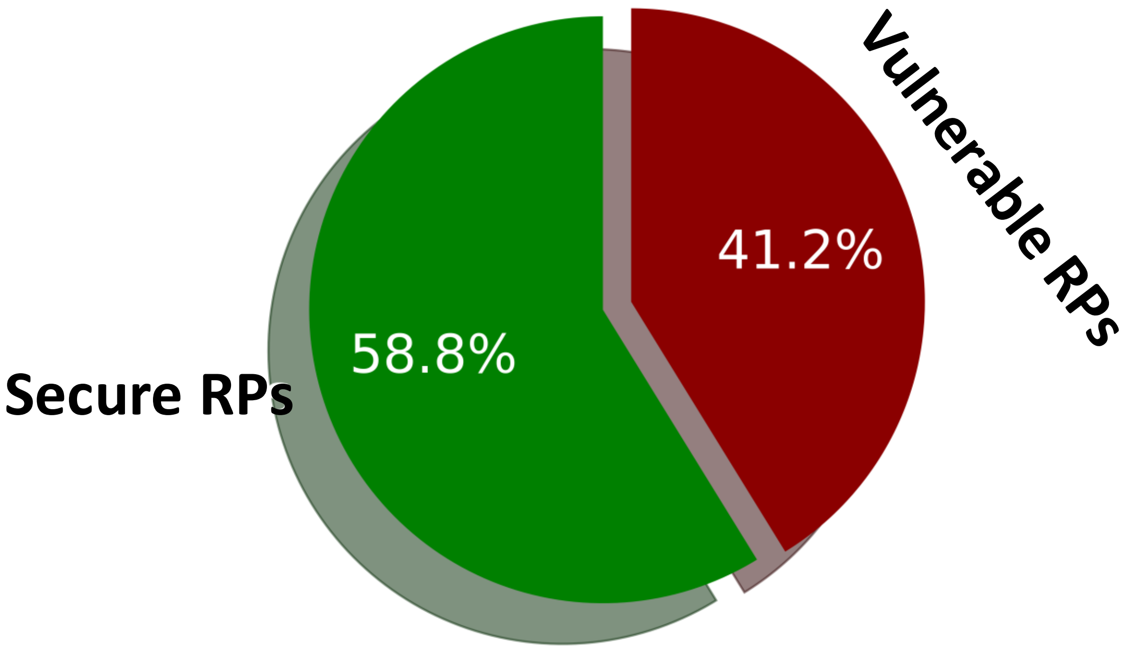
Internet Evaluations

Internet Evaluations (Then)



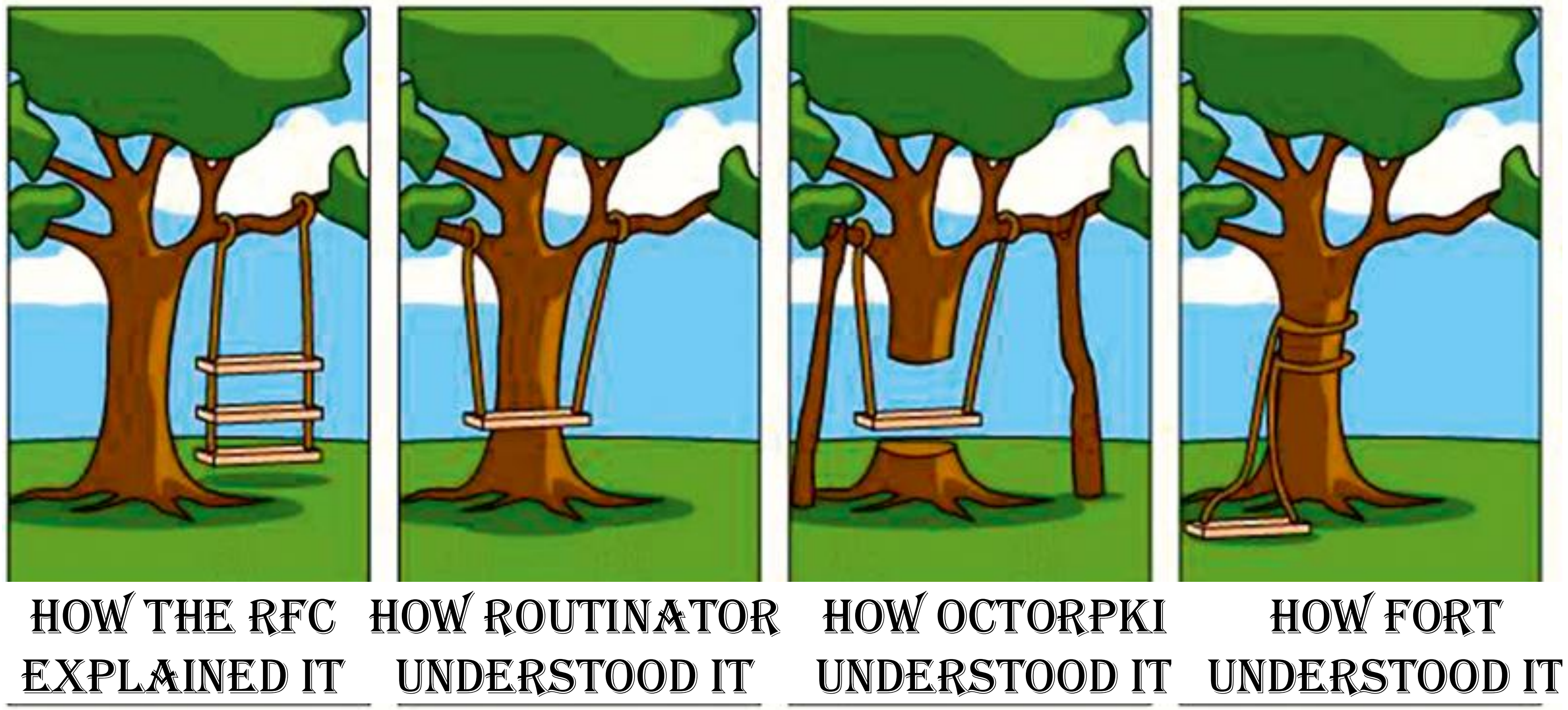
Internet Evaluations (Now)





Results: Global Inconsistencies

Results: Global Inconsistencies



Results: Global Inconsistencies

- **Post-processing ROA Payload:**

Routinator:	441,770	***Fort:***	435,002
OctoRPKI:	434,074	***rpki-client:***	441,777

Results: Global Inconsistencies

- **Post-processing ROA Payload:**

Routinator:	441,770	|	***Fort:***	435,002
OctoRPKI:	434,074	|	***rpki-client:***	441,777

- **Processing inconsistencies in the real-world:**

*6405 unprotected Amazon prefixes in **one implementation** due to the presence of OrganisationName header in certificates*

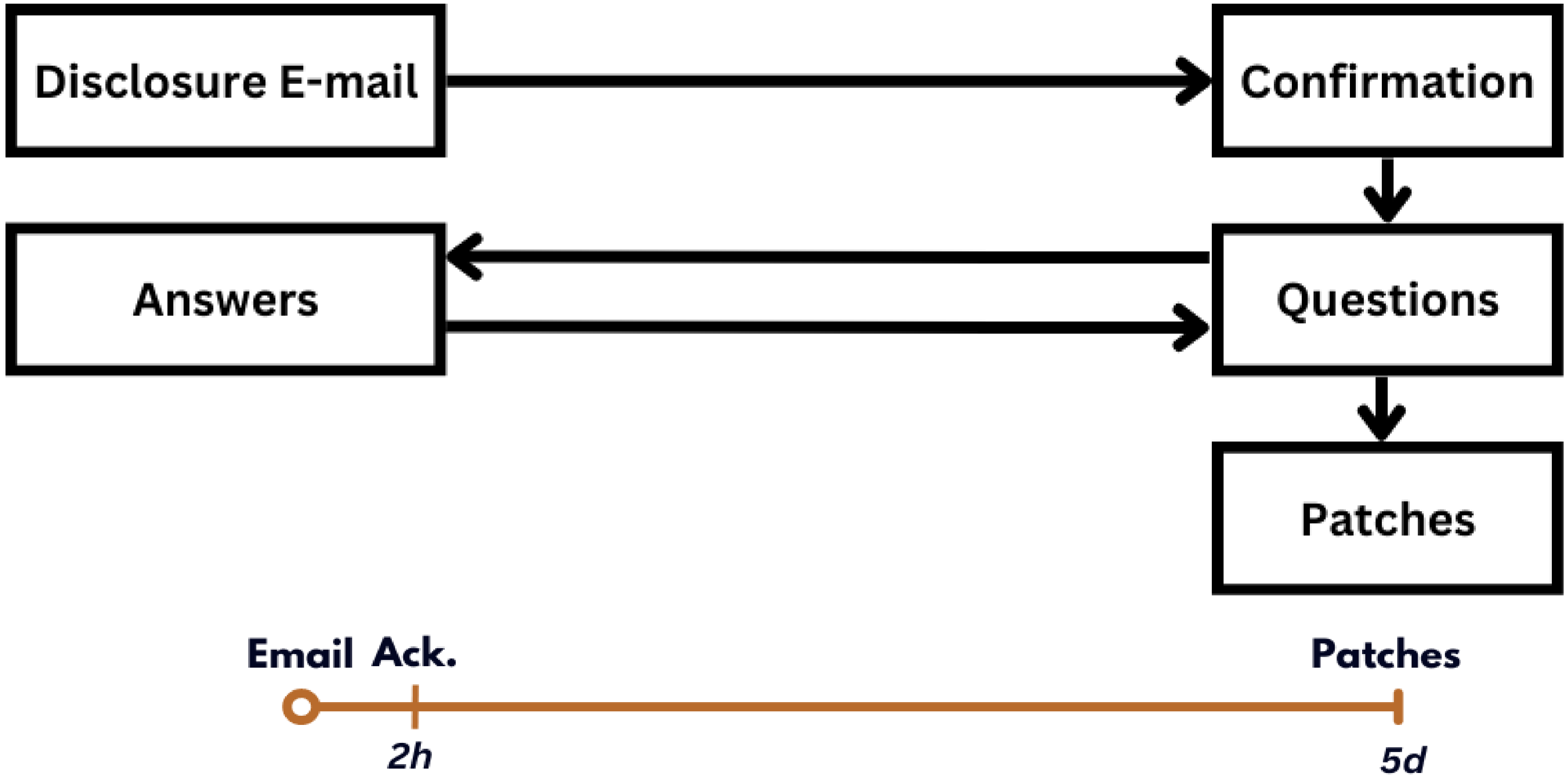
Disclosures

- Of course, we responsibly disclosed all vulnerabilities
- We sent out E-Mail to the vendors and waited for replies

Vulnerabilities in Routinator	***Sent: Jul 19th '23 - 20:25***
Vulnerabilities in OctoRPKI	***Sent: Jul 20th '23 - 11:01***
Vulnerabilities in Fort	***Sent: Jul 20th '23 - 11:56***

The experience differed significantly between vendors...

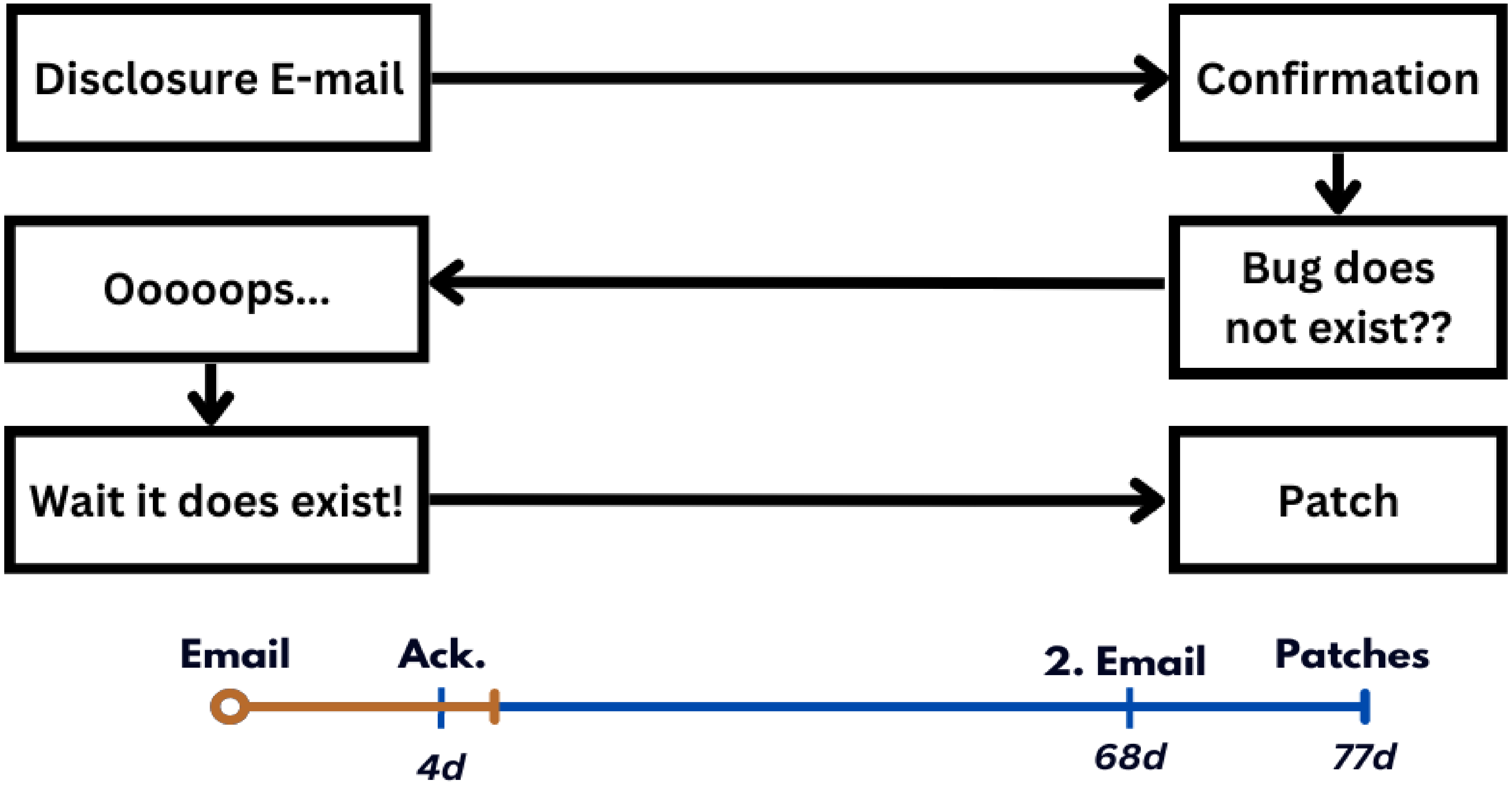
Disclosure – Vendor 1



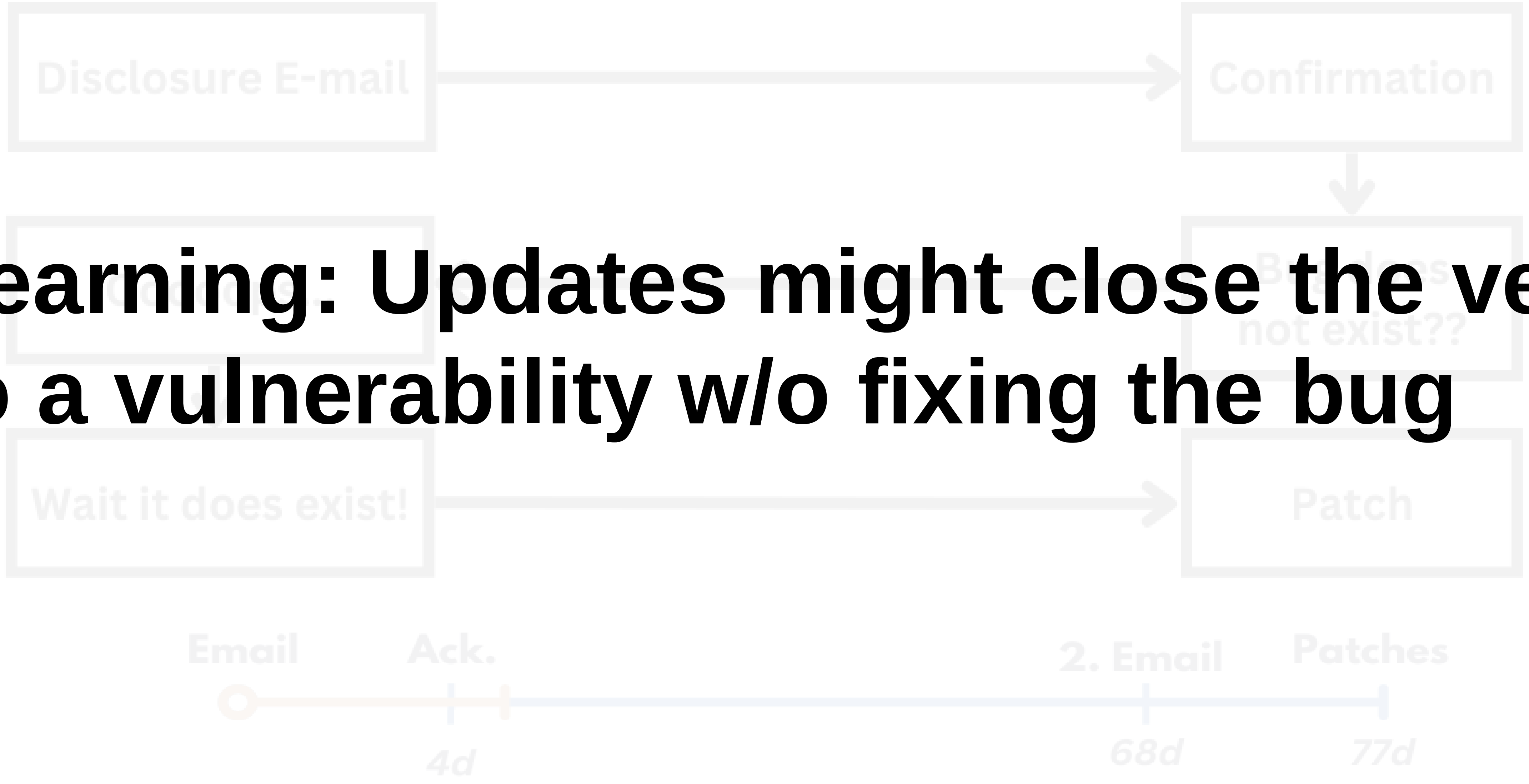
Disclosure – Vendor 1

That was nice!

Disclosure – Vendor 2

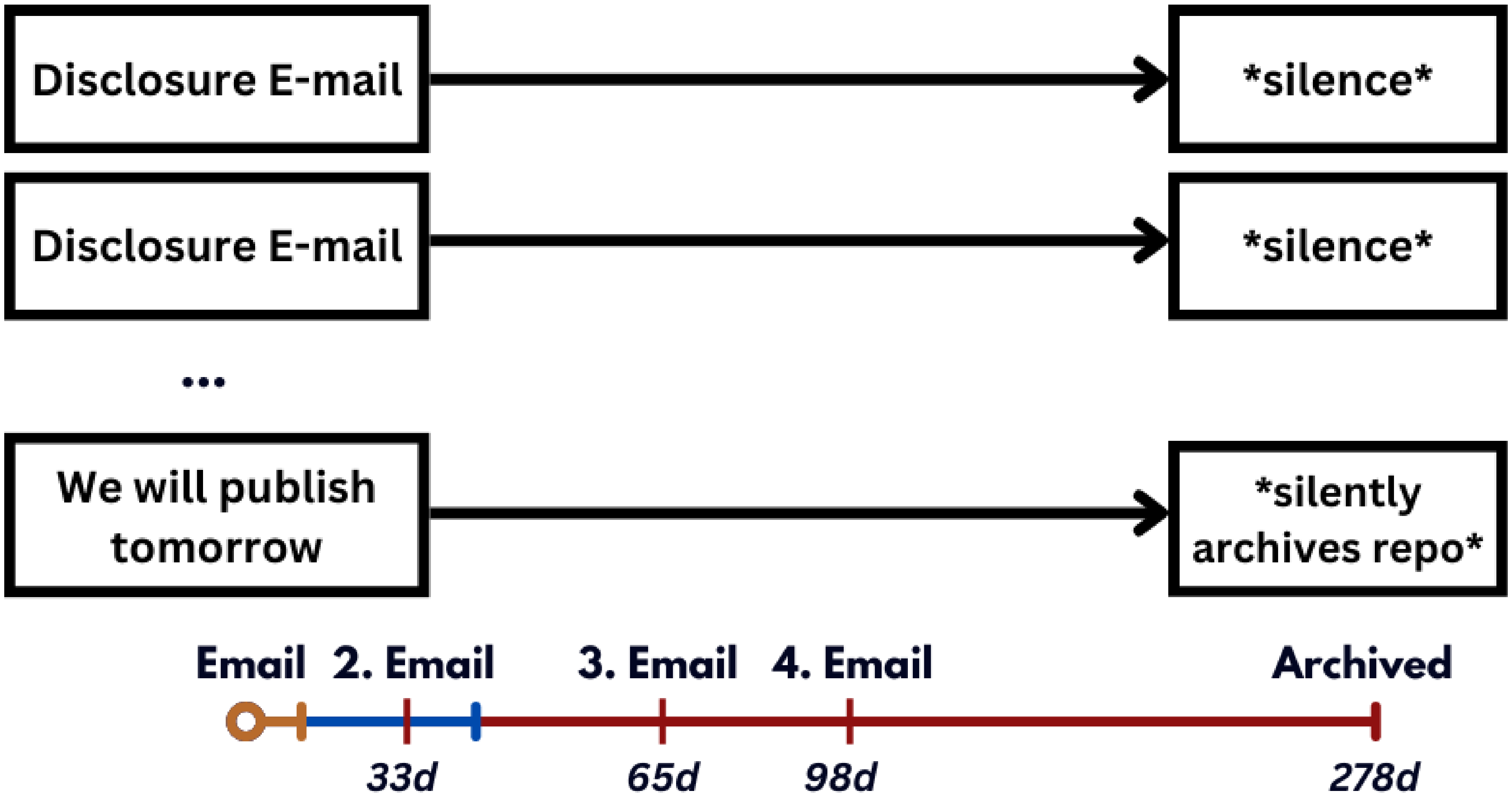


Disclosure – Vendor 2



Learning: Updates might close the vector to a vulnerability w/o fixing the bug

Disclosure – Vendor 3



Disclosure – Vendor 3

Disclosure E

nce*

Disclosure E

nce*

...

We will pul
tomorro

ntly
s repo*

Ema

hived

33d 65d 98d 278d

Disclosure – Vendor 3

Learning: If you don't get a reply, keep trying... Deprecation is better than nothing

Lessons Learned

- **Takeaway 1:** RPKI is a core internet security protocol! The software maturity is (partially) not production ready.

- **Takeaway 2:** 41.2% of RPs on the internet are still vulnerable! Operators must be more reactive and patch their software.

- **Takeaway 3:** Fuzzing crypto is hard! We need more tools to efficiently fuzz cryptographic protocols.

Thank you!

donika.mirdita@athene-center.de
niklas.vogel@athene-center.de

ATHENE
National Research Center for Applied Cybersecurity