



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

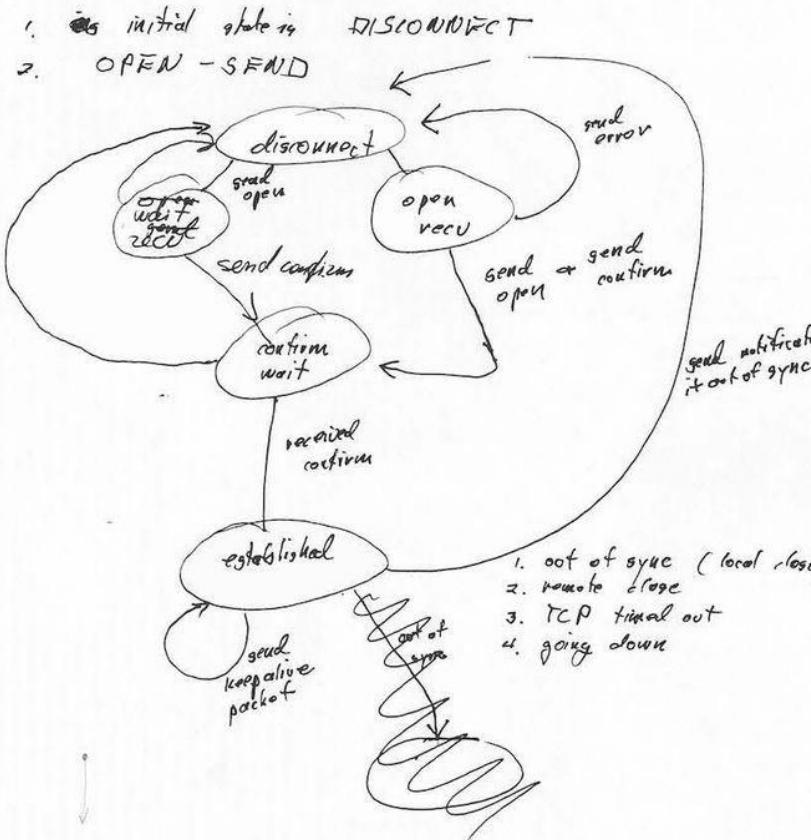
❖ Disclosure Process

BGP as Achille's Heel



BGP as Achille's Heel

State Diagram



loughheed@cisco.com
YAKOV@IBH.COM

4/15 - 3/26 - 1941 (11-7) PST
(914) 945-3896 (8-5) FST

B.G. N.	version number block length version number block type holdown timer	2 bytes 2 bytes 2 bytes 2 bytes 2 bytes (minutes)
types:	open - 1 update - 2 notification - # keepalive - 8	version is currently 1
open:	my AS # link type up - 1 down - 2 interval - 4 I-link - 8 auth type code 0 - none	2 bytes 1 byte (not usual in update direction field)
update:	repeat structure according to block length network # first hop gateway metric count of AS direction AS #	4 bytes 4 bytes 2 bytes 1 byte 1 byte } repeat "count" times 2 bytes }
notification:	opcode data	2 bytes variable

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](#)]

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

INFORMATIONAL
Errata Exist
M. Lepinski
S. Kent
BBN Technologies
February 2012

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

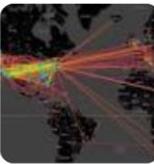
How AWS is helping to secure internet routing

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

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

 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

 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



case study

RPKI

The RPKI Protocol

How AWS

by Fredrik Korsbäck

Harry Coker: Federal Agencies Advance Resource Public Key Infrastructure Adoption

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

BleepingComputer

All Dutch govt net hijacking

The Dutch government will standard on all its systems
9 Apr 2023

Capacity Media

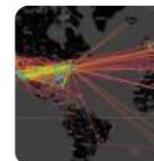
Telia Carrier se

Telia Carrier has announced its own RPKI infrastructure (RPKI)

17 Sept 2019

BleepingComputer

attacks and route leaks



lers, has now deployed RPKI on its

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 BRODIN - 6/6/2024, 11:40 PM



Harry Coker

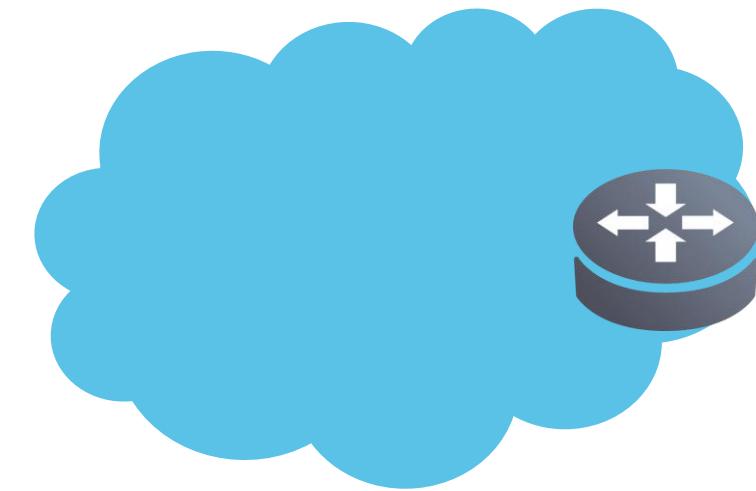
National Cyber Director
Office of the National Cyber Director

rent crooks, spies hijacking victims'

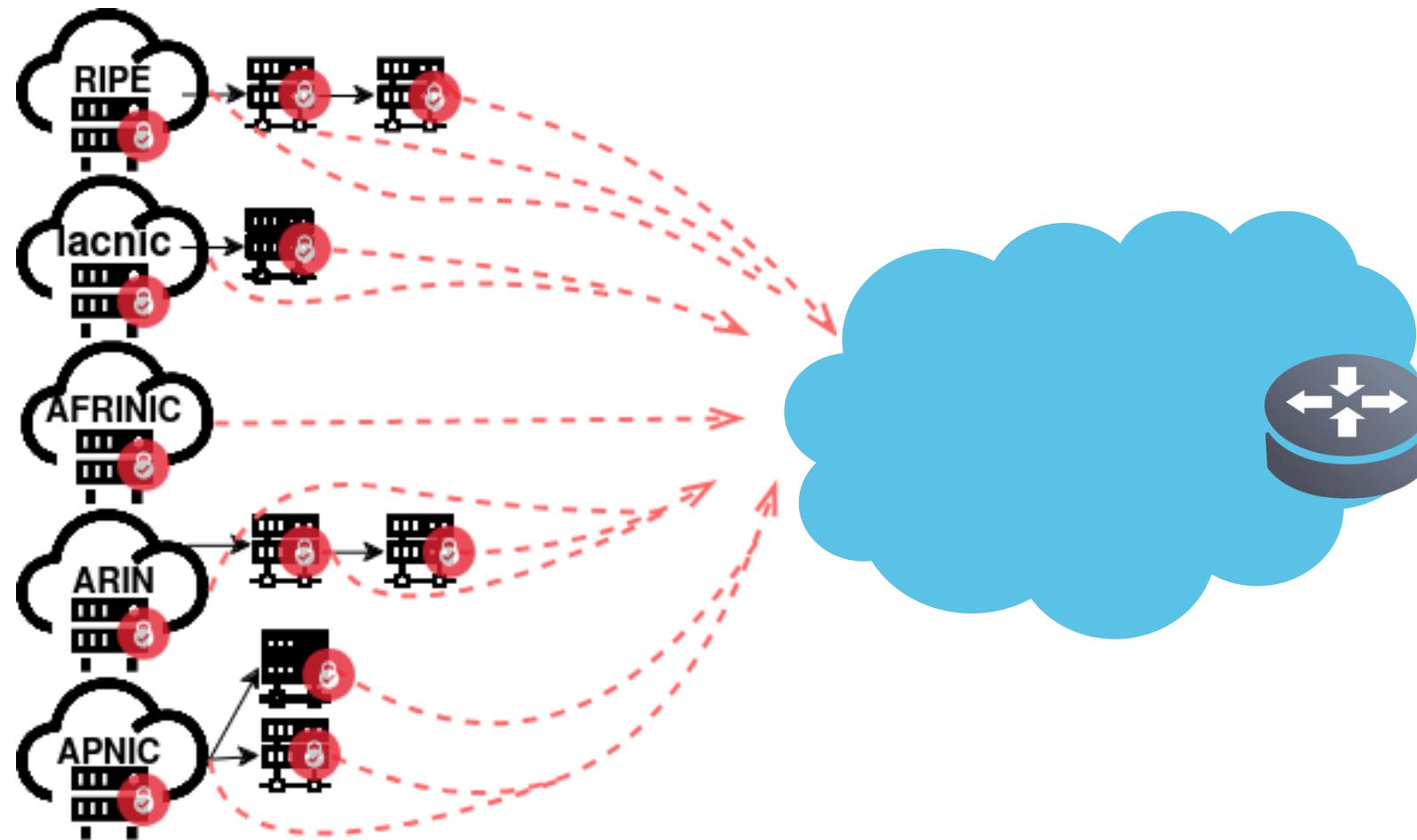
ue 31 Mar 2020 // 12:00 UTC

ix,
iting

BGP Security with RPKI



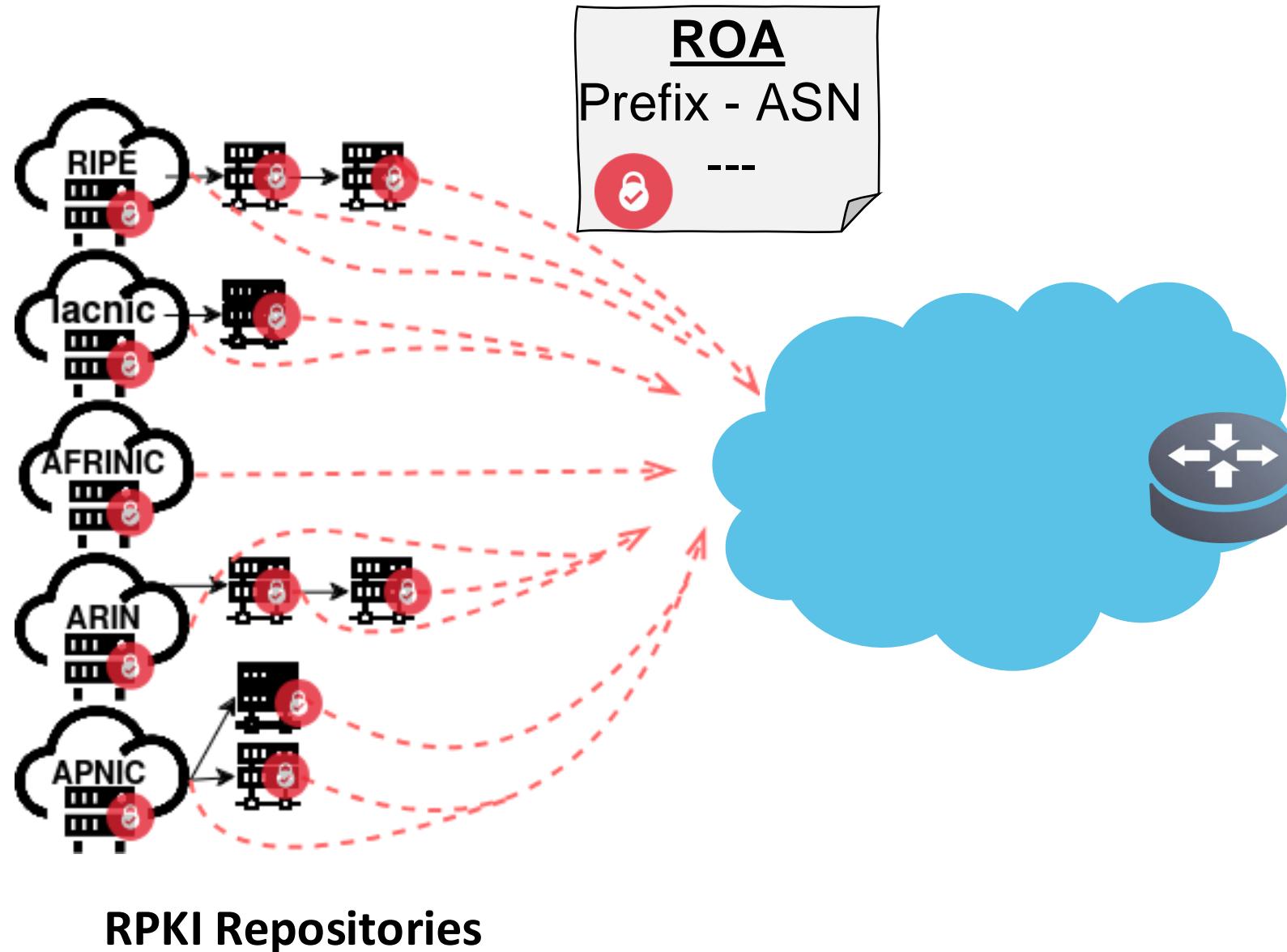
BGP Security with RPKI



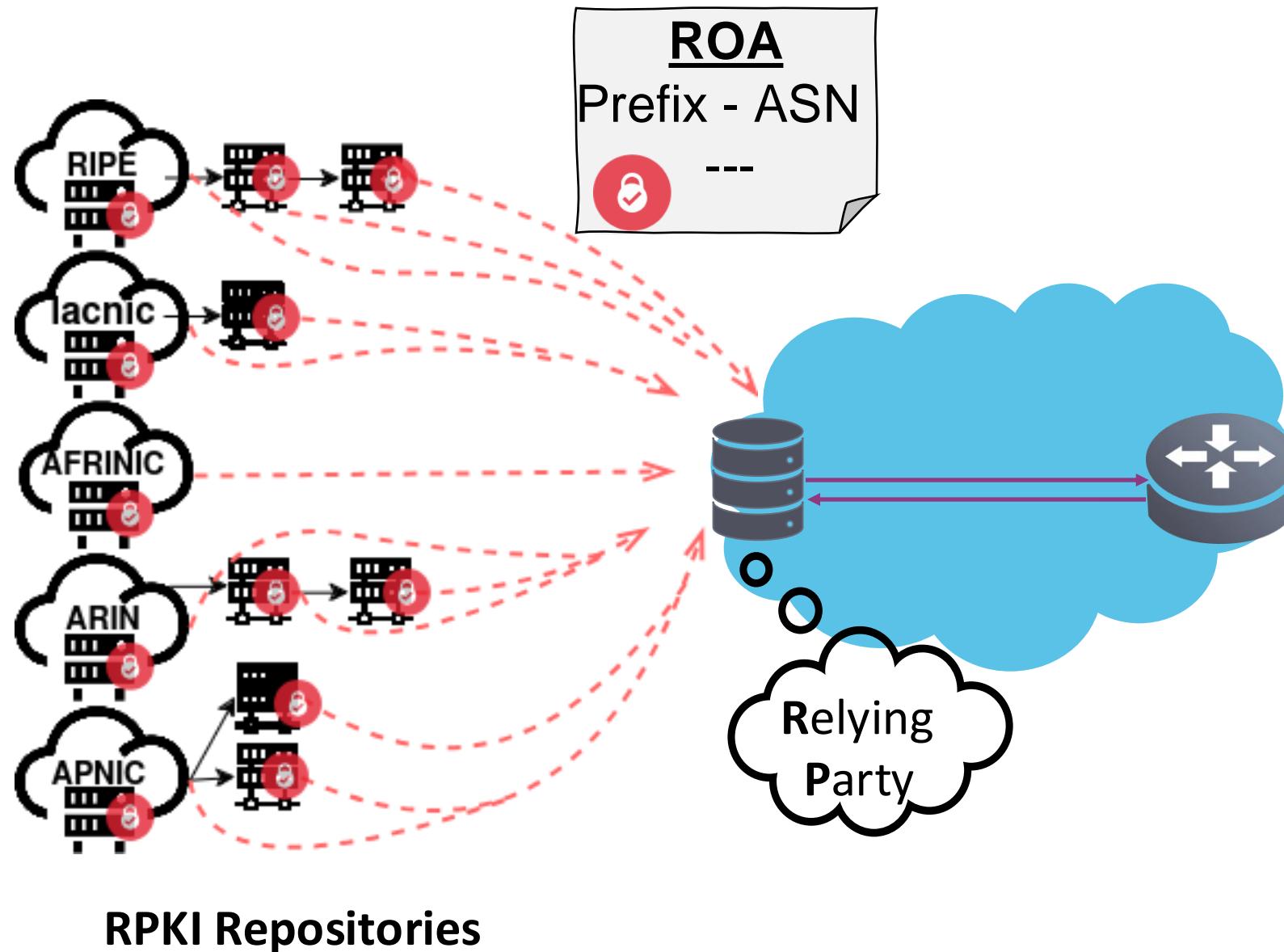
RPKI Repositories

#BHUSA @BlackHatEvents

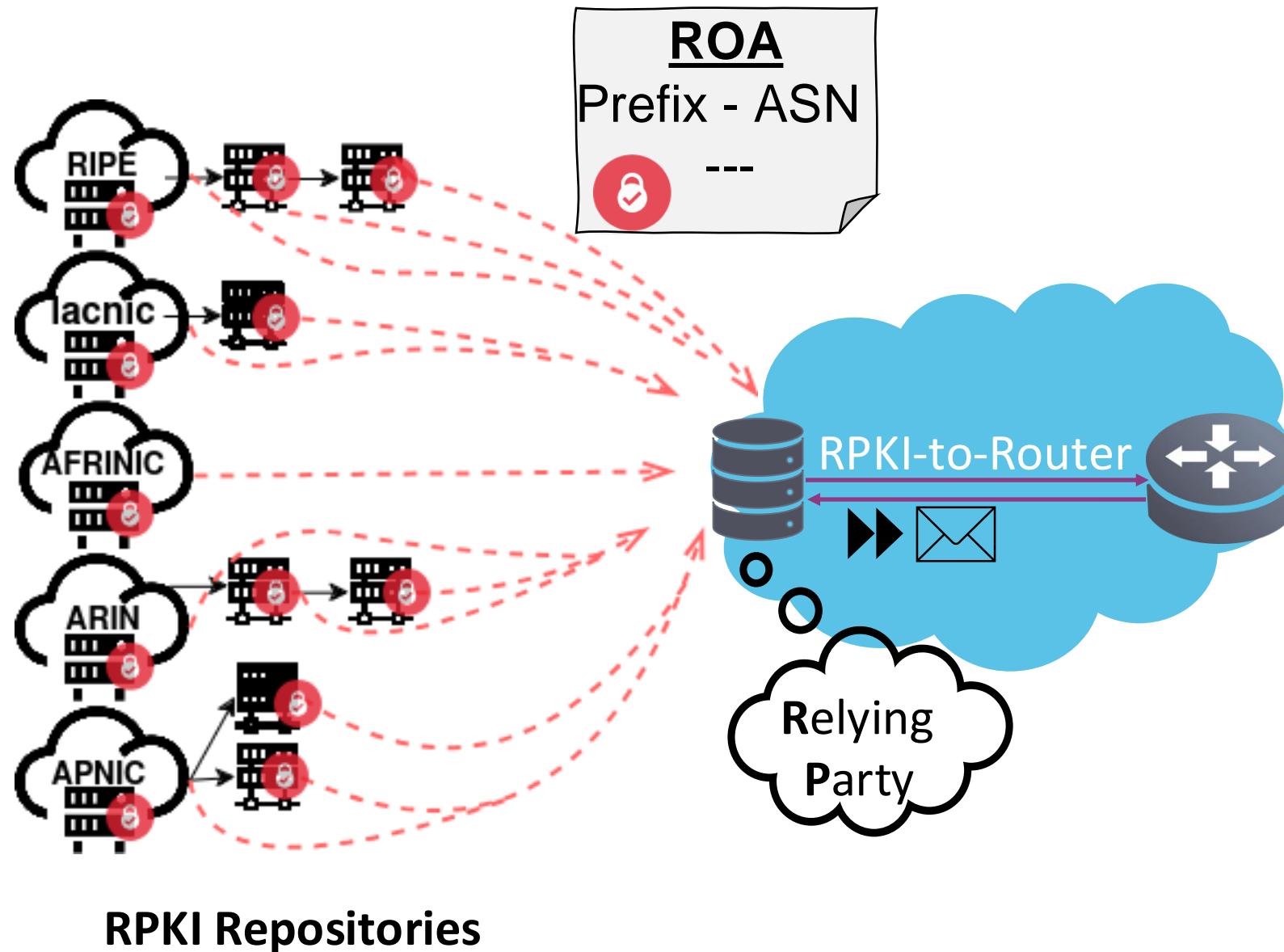
BGP Security with RPKI



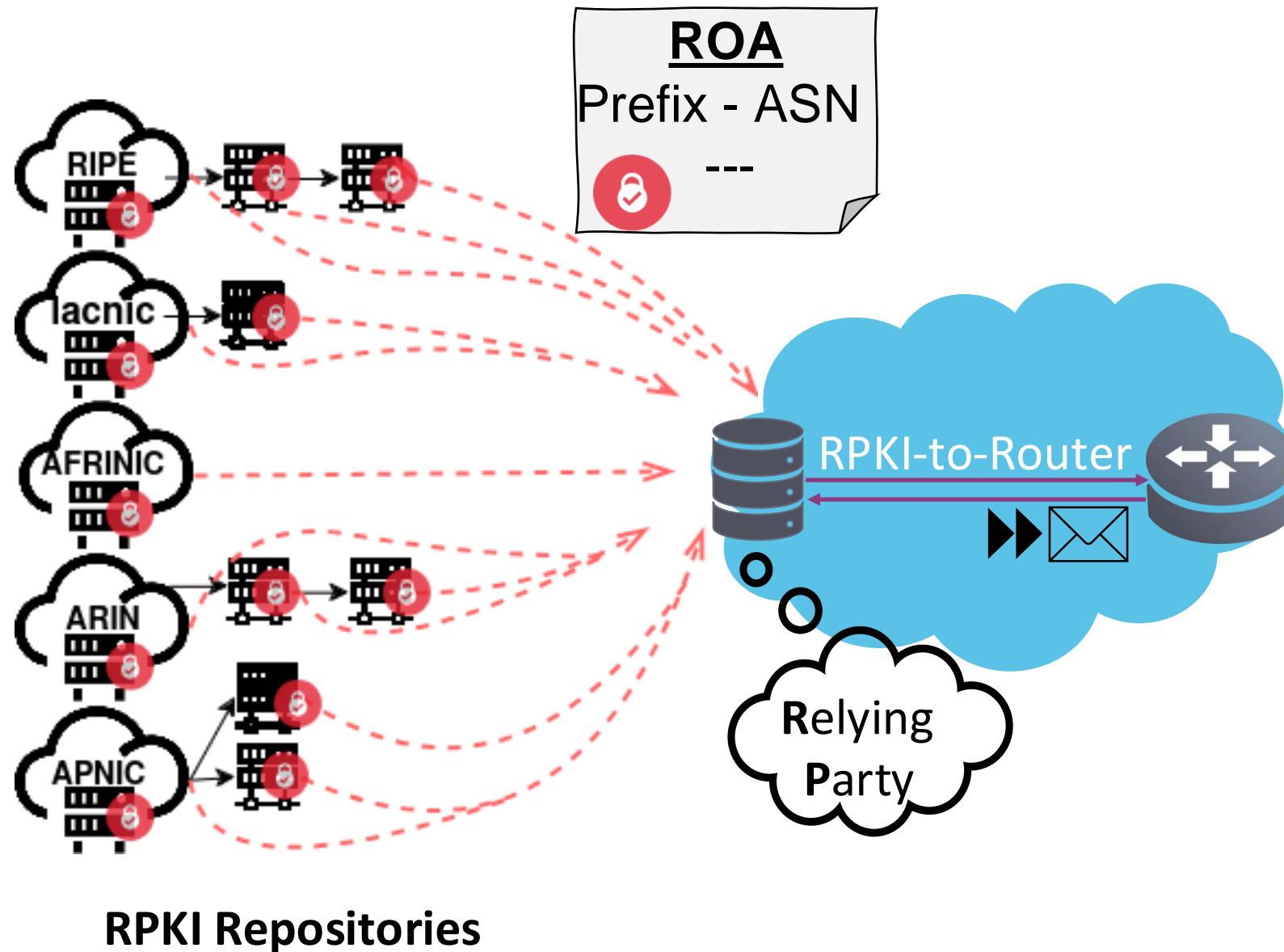
BGP Security with RPKI



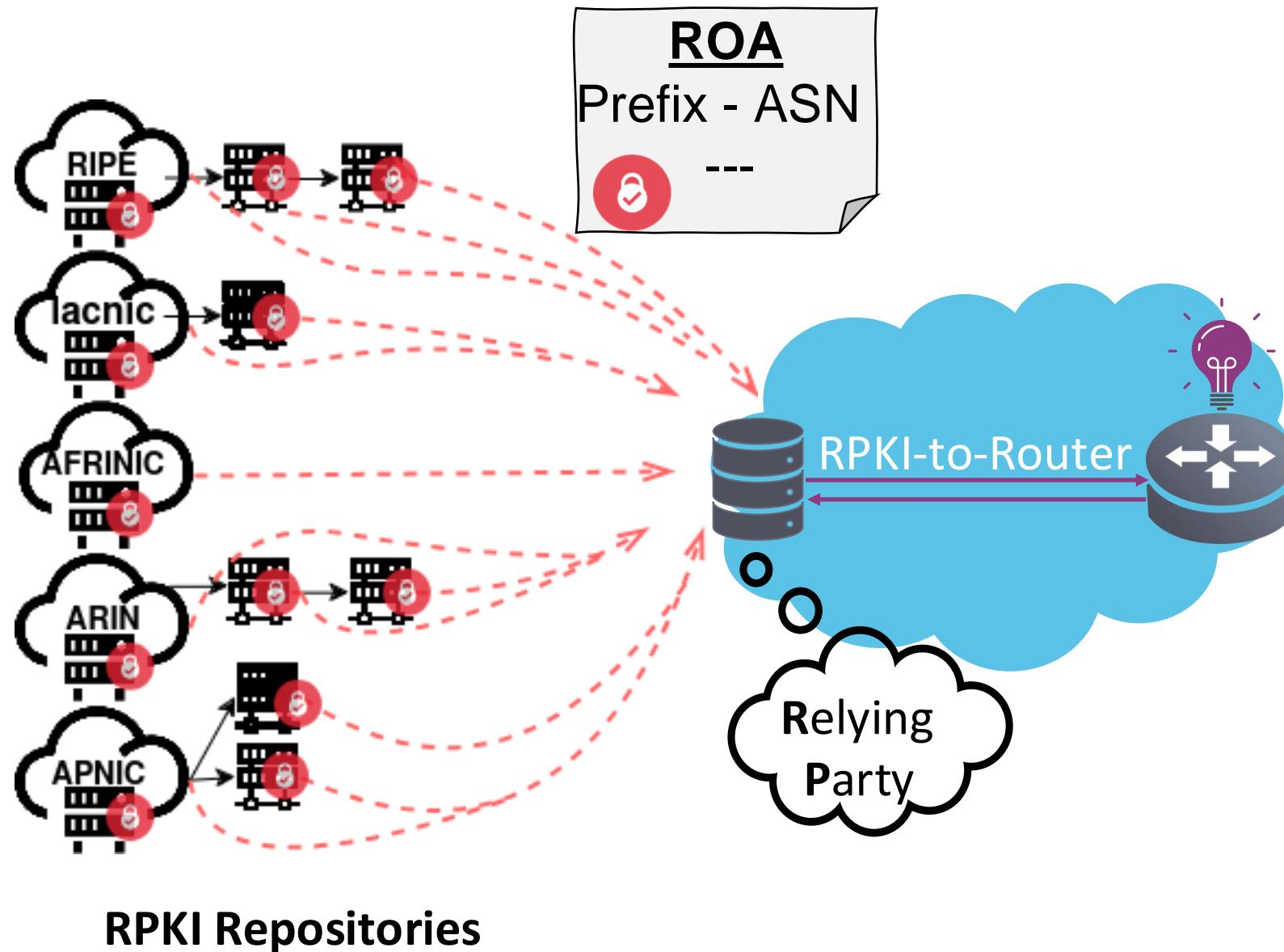
BGP Security with RPKI



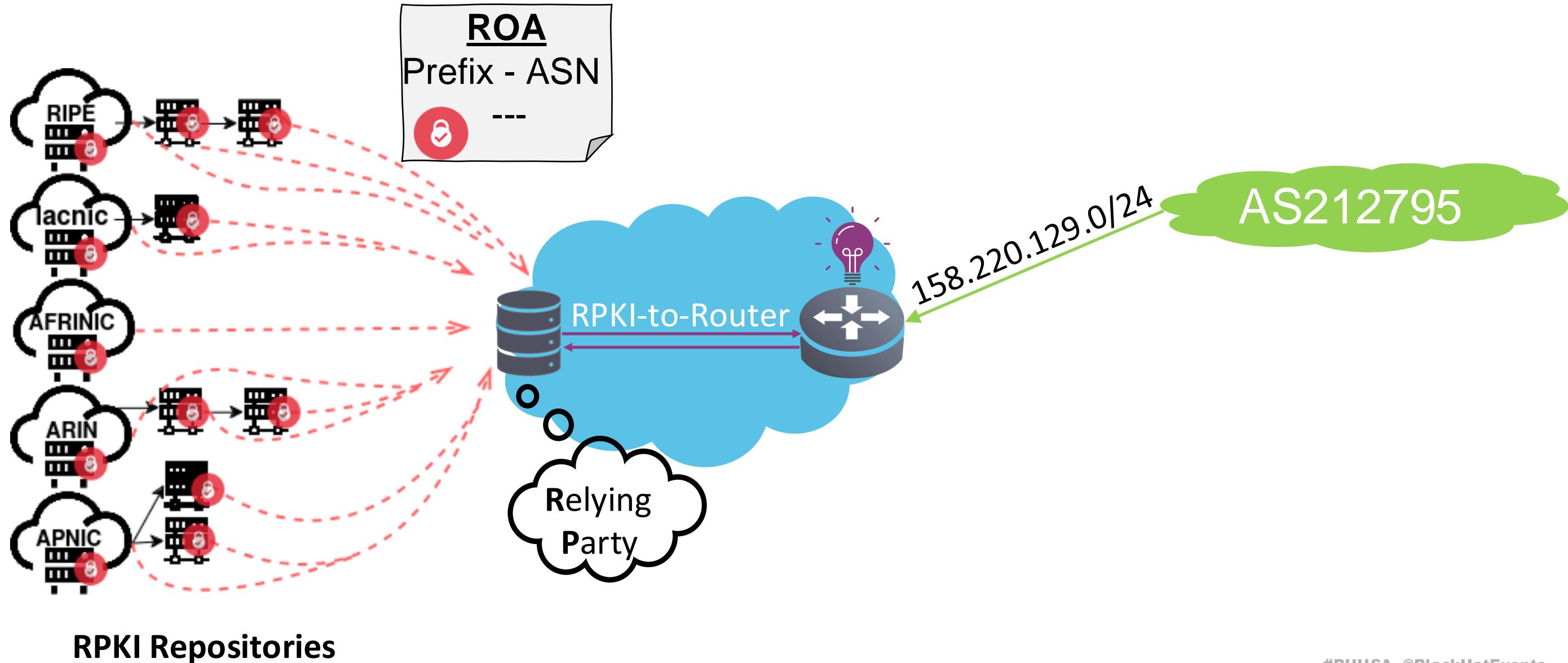
BGP Security with RPKI



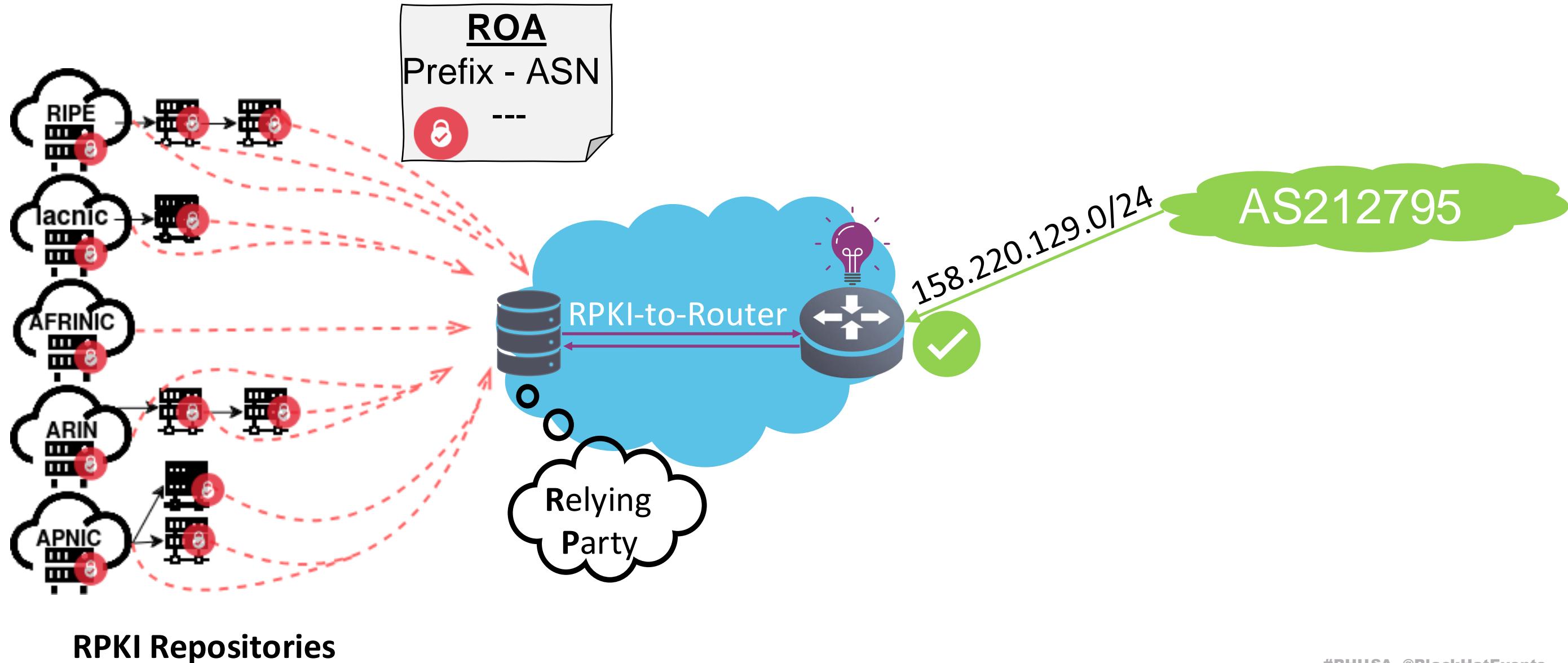
BGP Security with RPKI



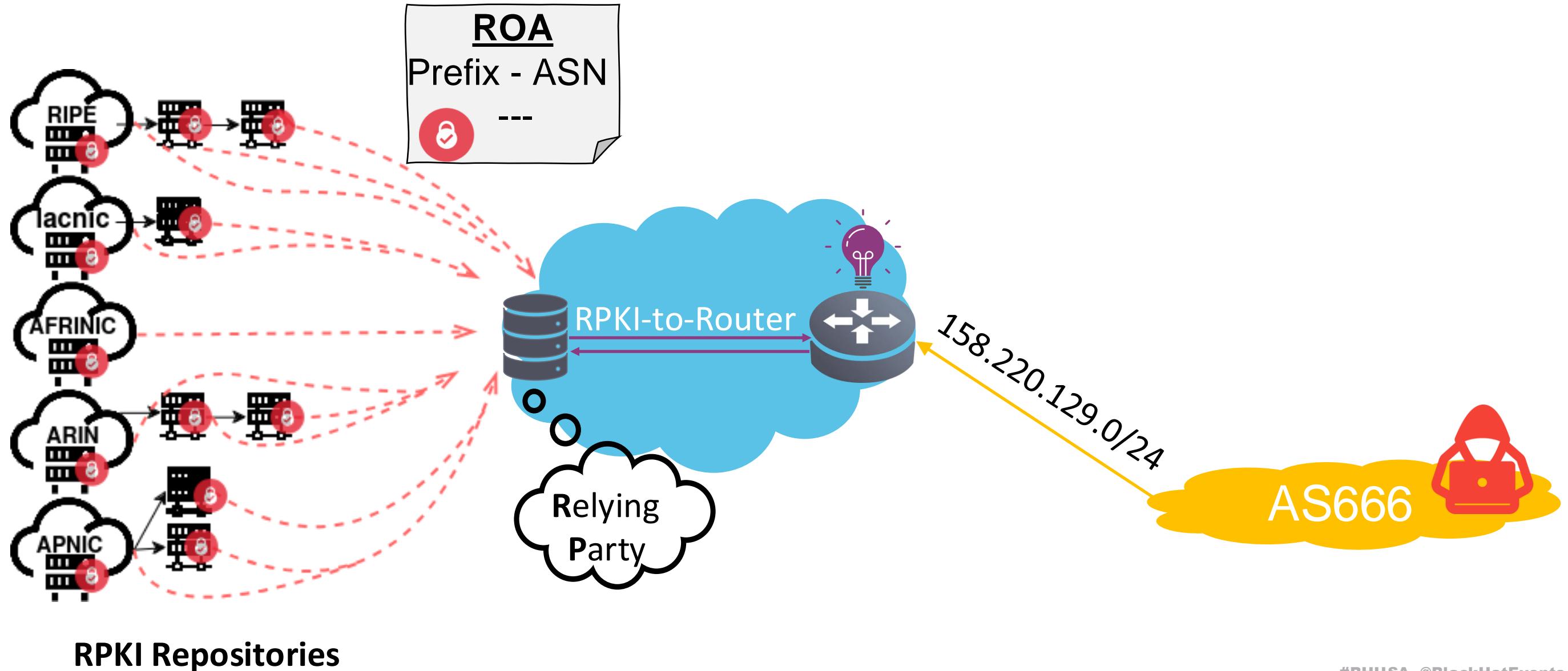
BGP Security with RPKI



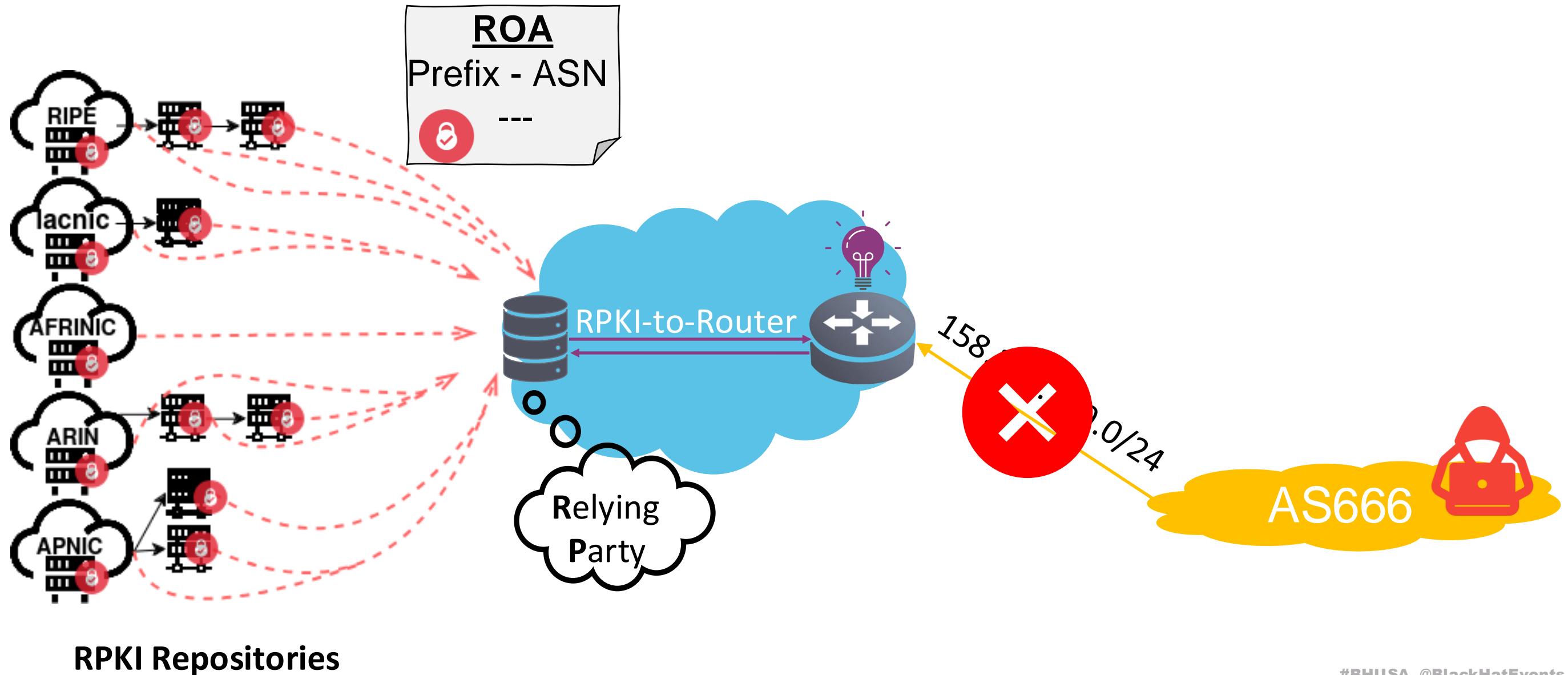
BGP Security with RPKI



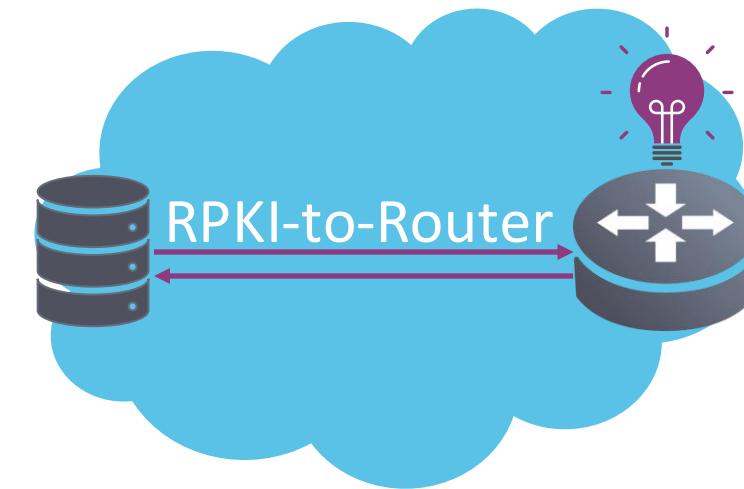
BGP Security with RPKI



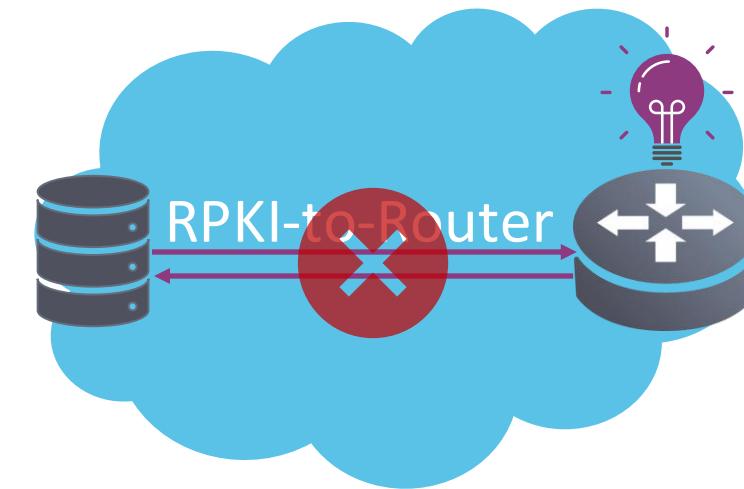
BGP Security with RPKI



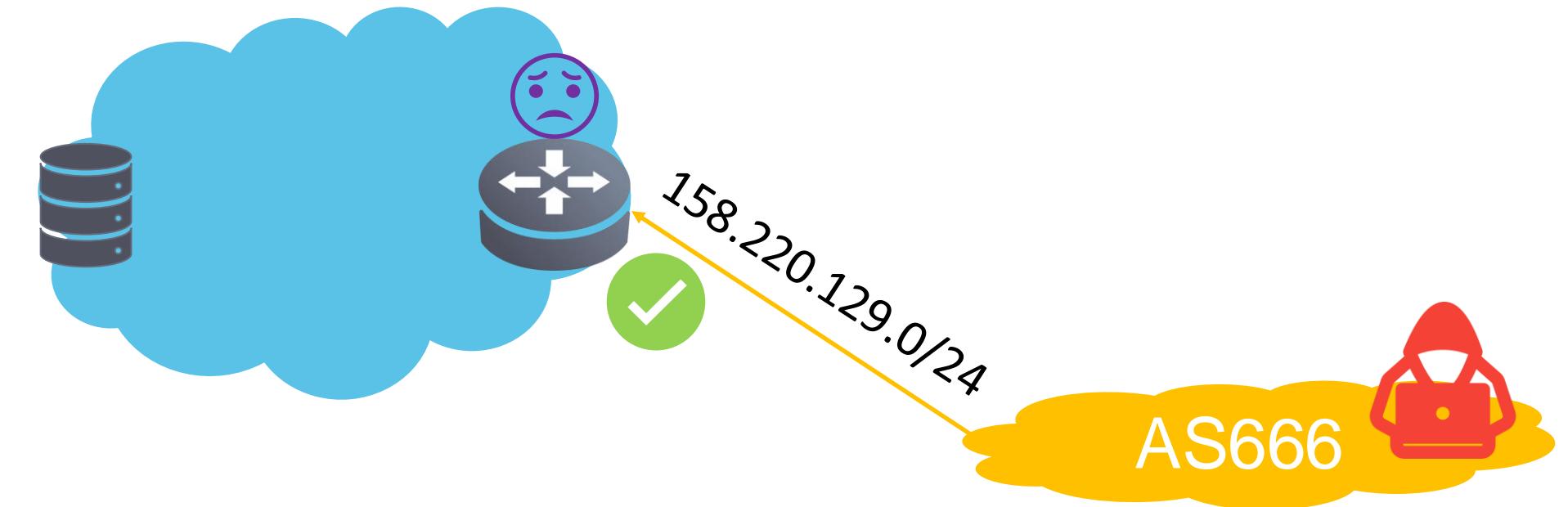
Why is DoS-ing RPs a big deal?



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

```
1973     Self::_create(data, &mut target).map_err(|err| {  
1974         error!(  
1975             "Fatal: failed to write file {}: {}", path.display(), err  
1976         );  
1977         Failed  
1978     })
```

So we decided to tinker with the protocol...



➤ Relaying Party Impl. 1: crash when objects malformed

```
1973     Self::_create(data, &mut target).map_err(|err| {
1974         error!(
1975             "Fatal: failed to write file {}: {}", path.display(), err
1976         );
1977         Failed
1978     })
```

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

```
1317
1318
1319
1320
      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
 - 1973 Self:::create(data, &mut target).map_err(|err| {
 - 1974 Err::RlpError{ reason: "Failed to write file": target.path().display(), err }
 - 1975 })
 - 1976 }
 - 1977 })
 - 1978 })
- => **84.9% of global Relying Party deployments affected by low-cost low-burden RPKI Downgrade Attacks**
- Relaying Party Impl. 2: crash when index out-of-bounds
 - 1317 if iterationsUntilStable > *MaxIterations {
 - 1318 log.Fatal("Max iterations has been reached.
 - 1319 This number can be adjusted with -max.iterations")
 - 1320 }

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.... ????

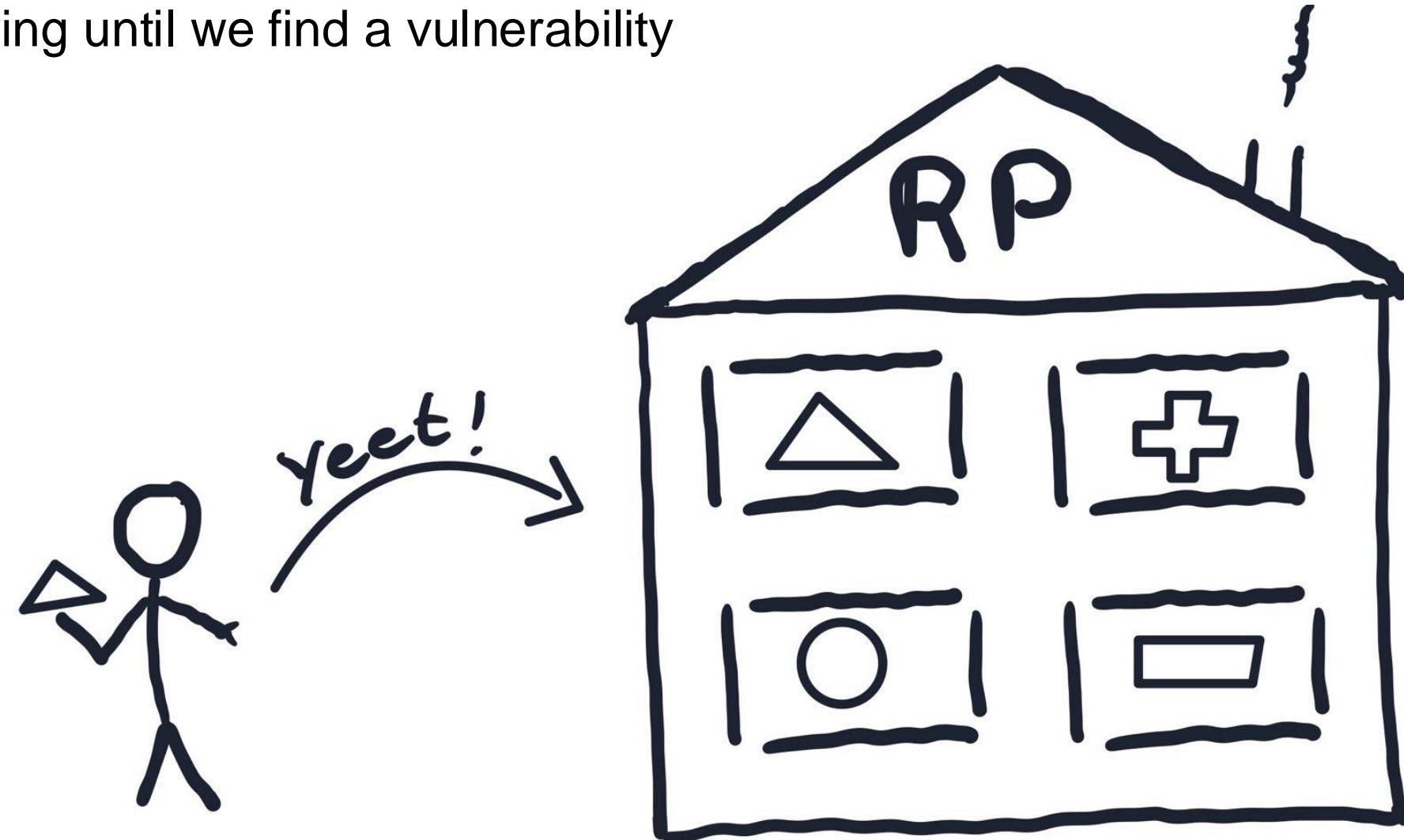
Our simple Plan

- 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



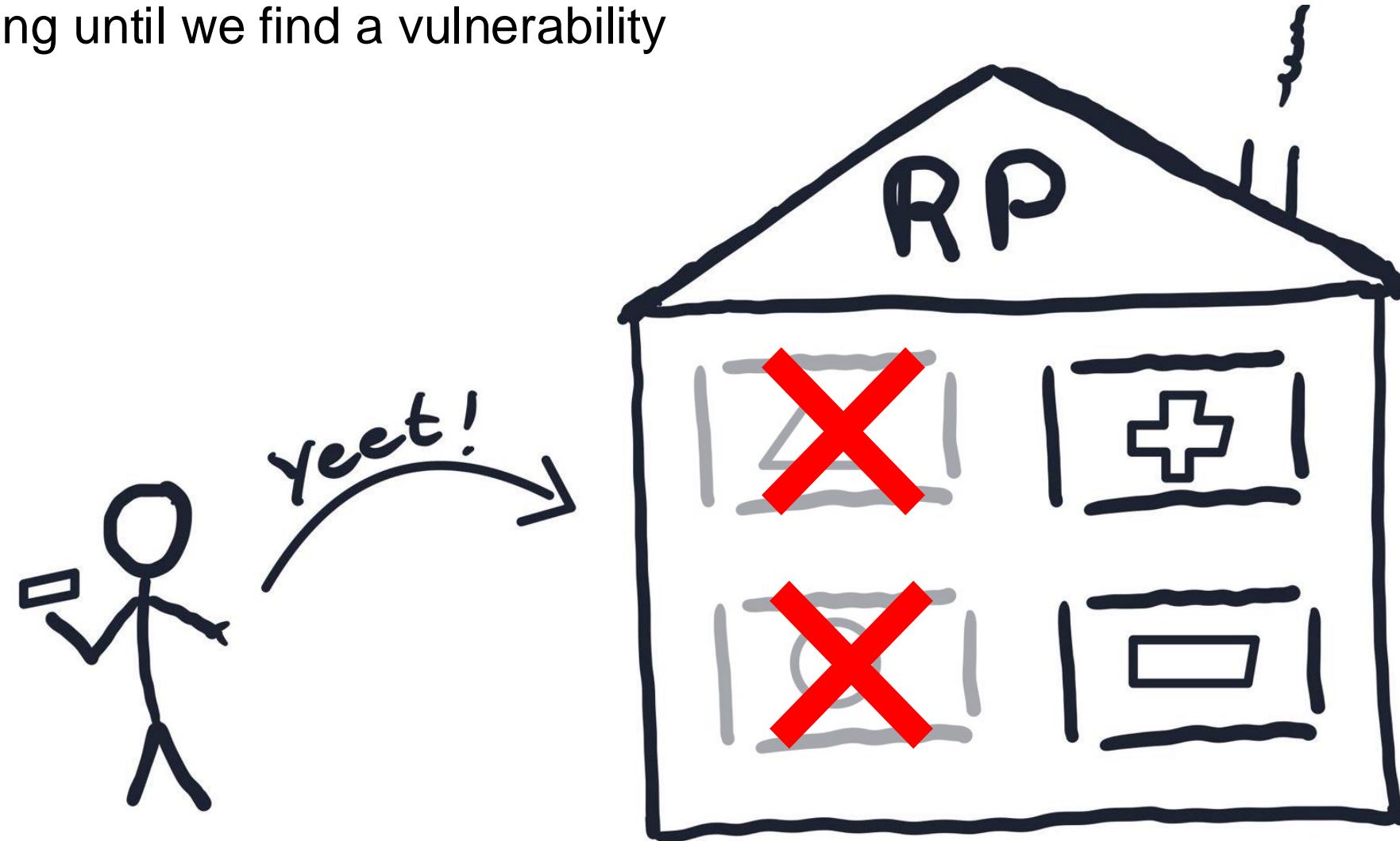
Our simple Plan

- 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



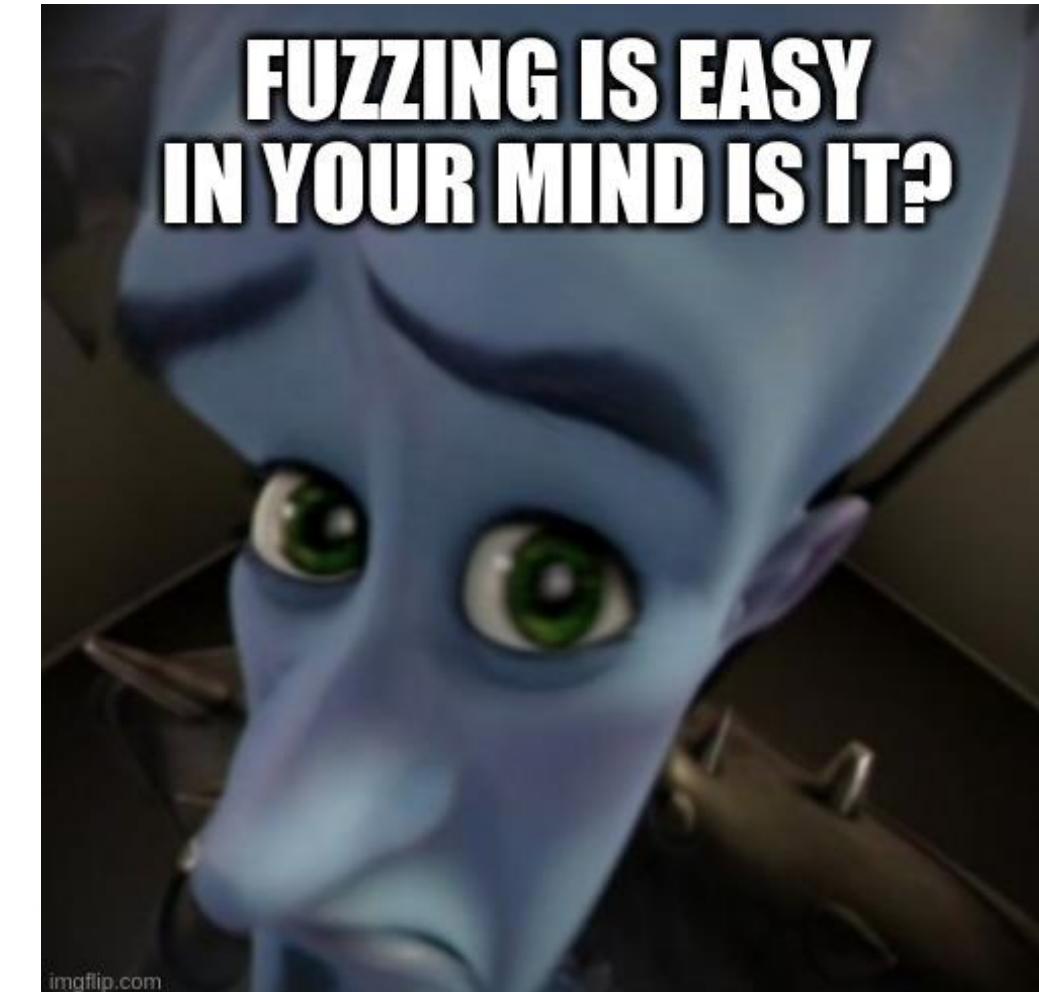
Our simple Plan

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



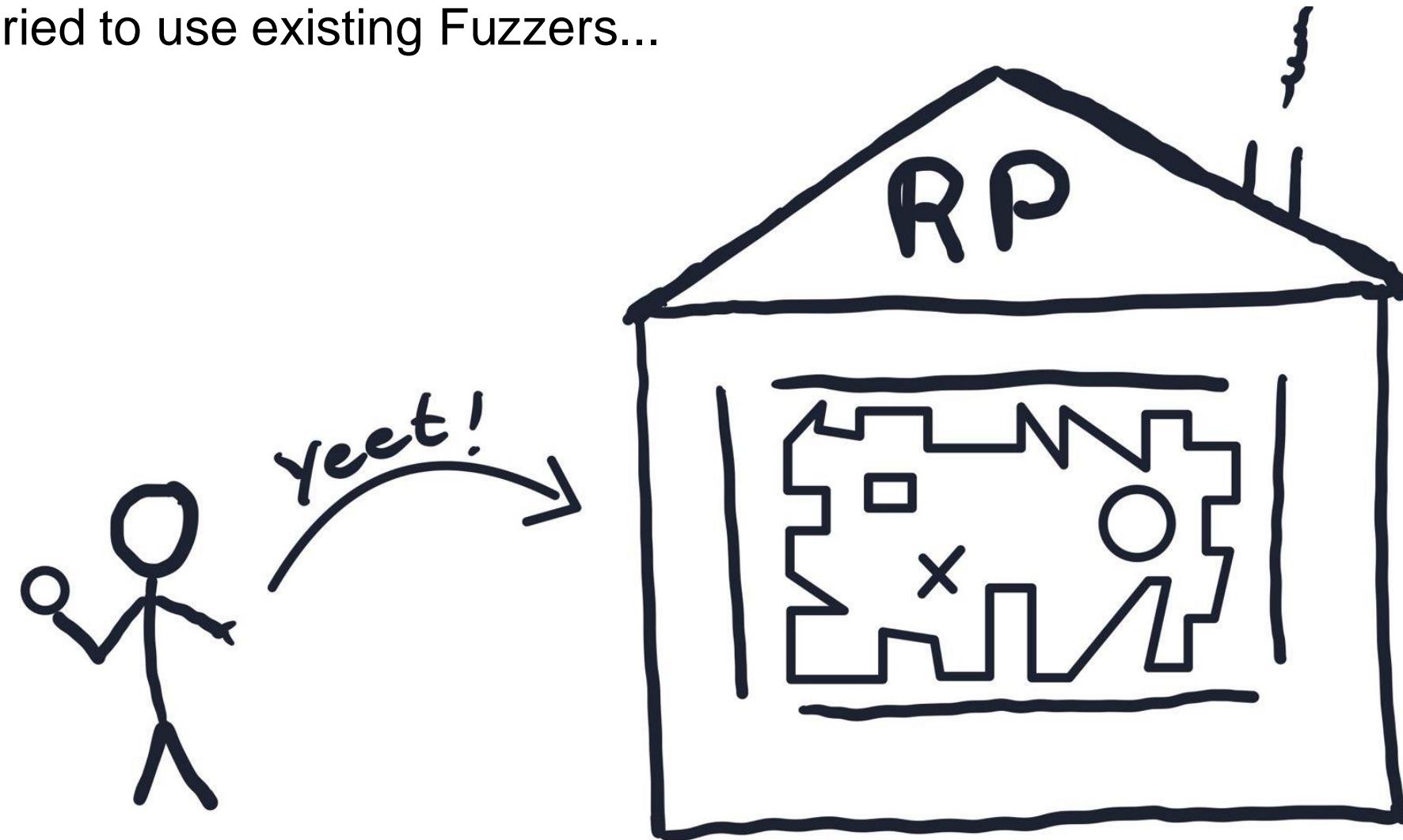
Our simple Plan

- 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...

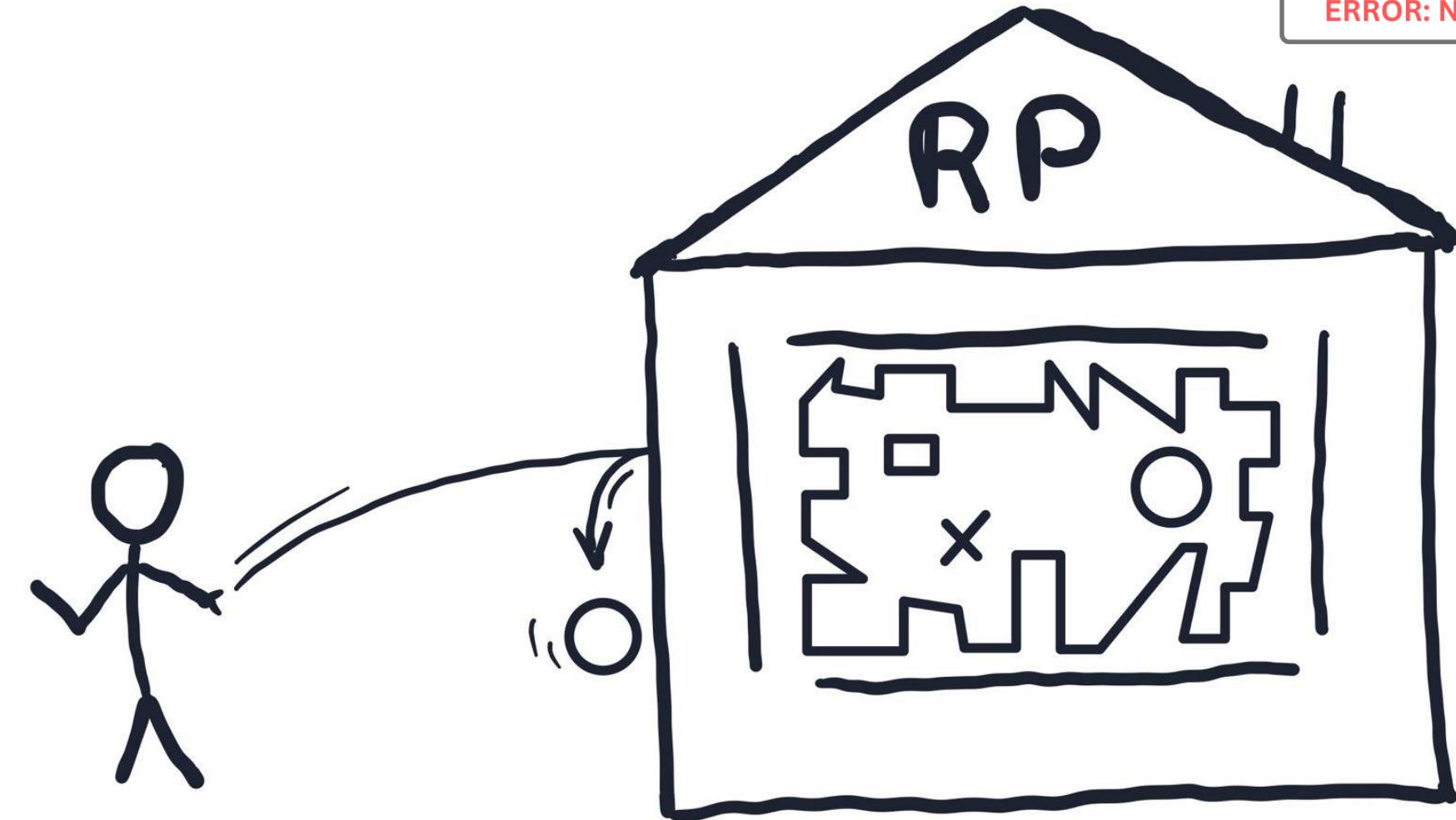


The complex Reality

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

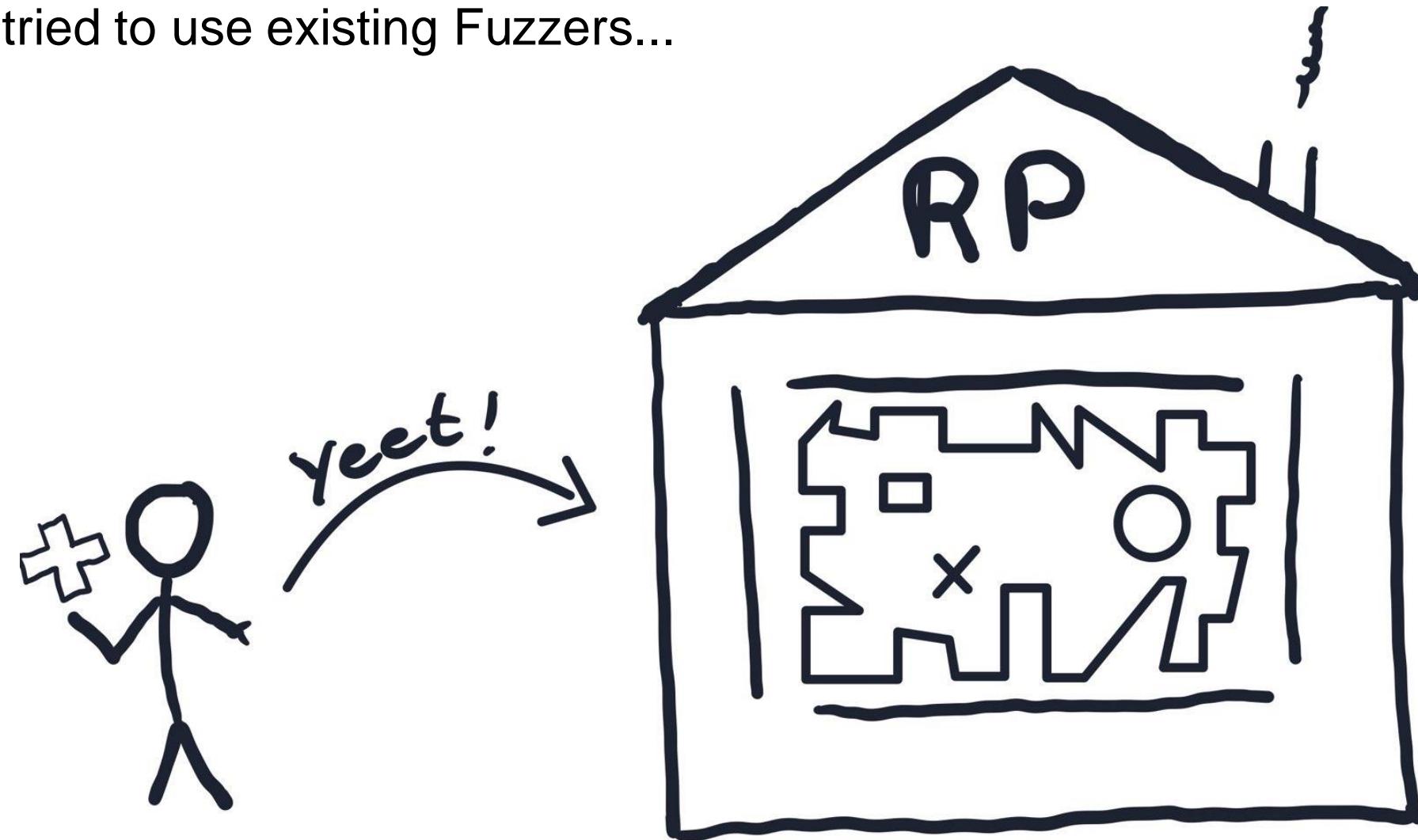
Routinator.log

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...

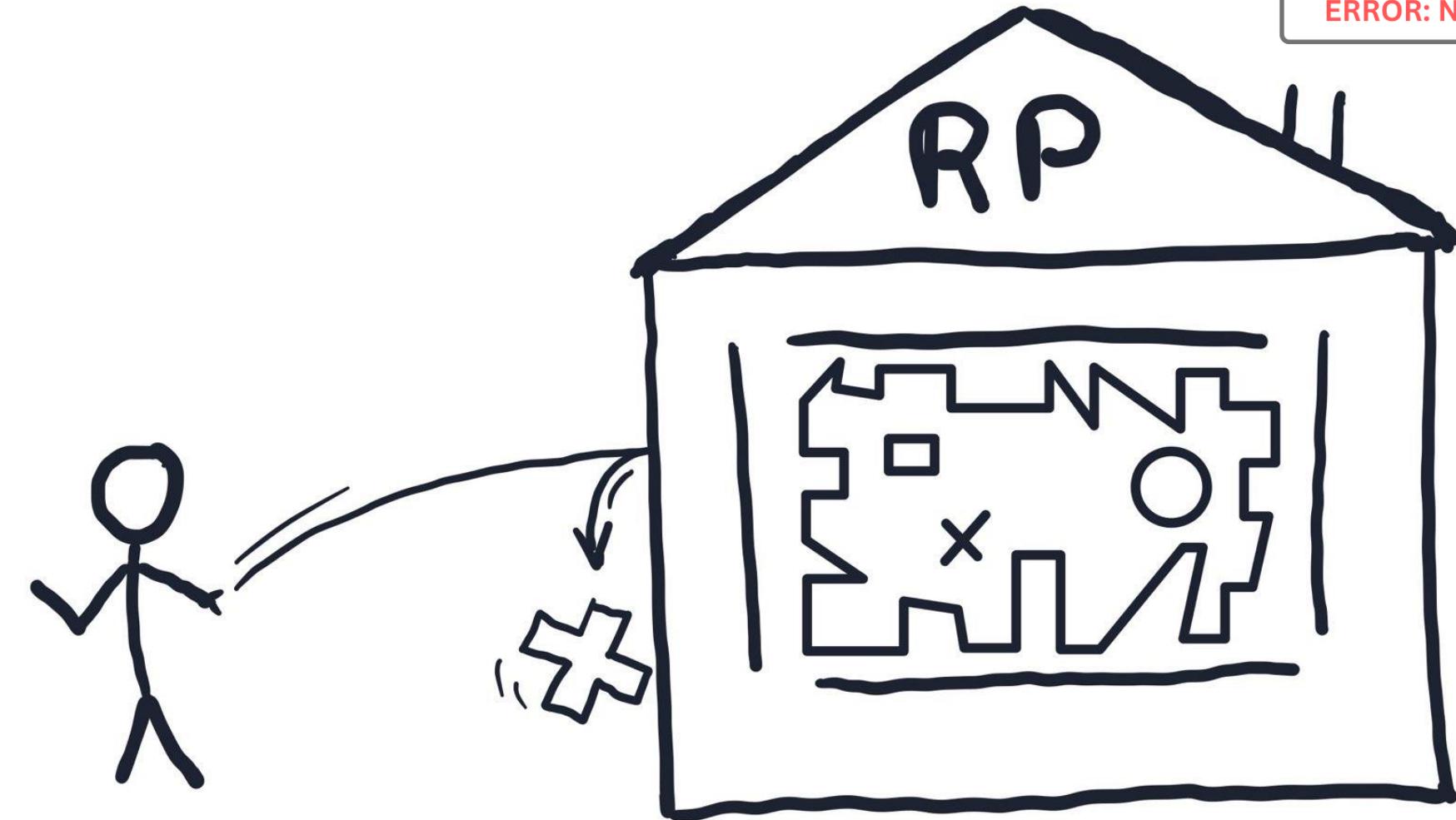


The complex Reality

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

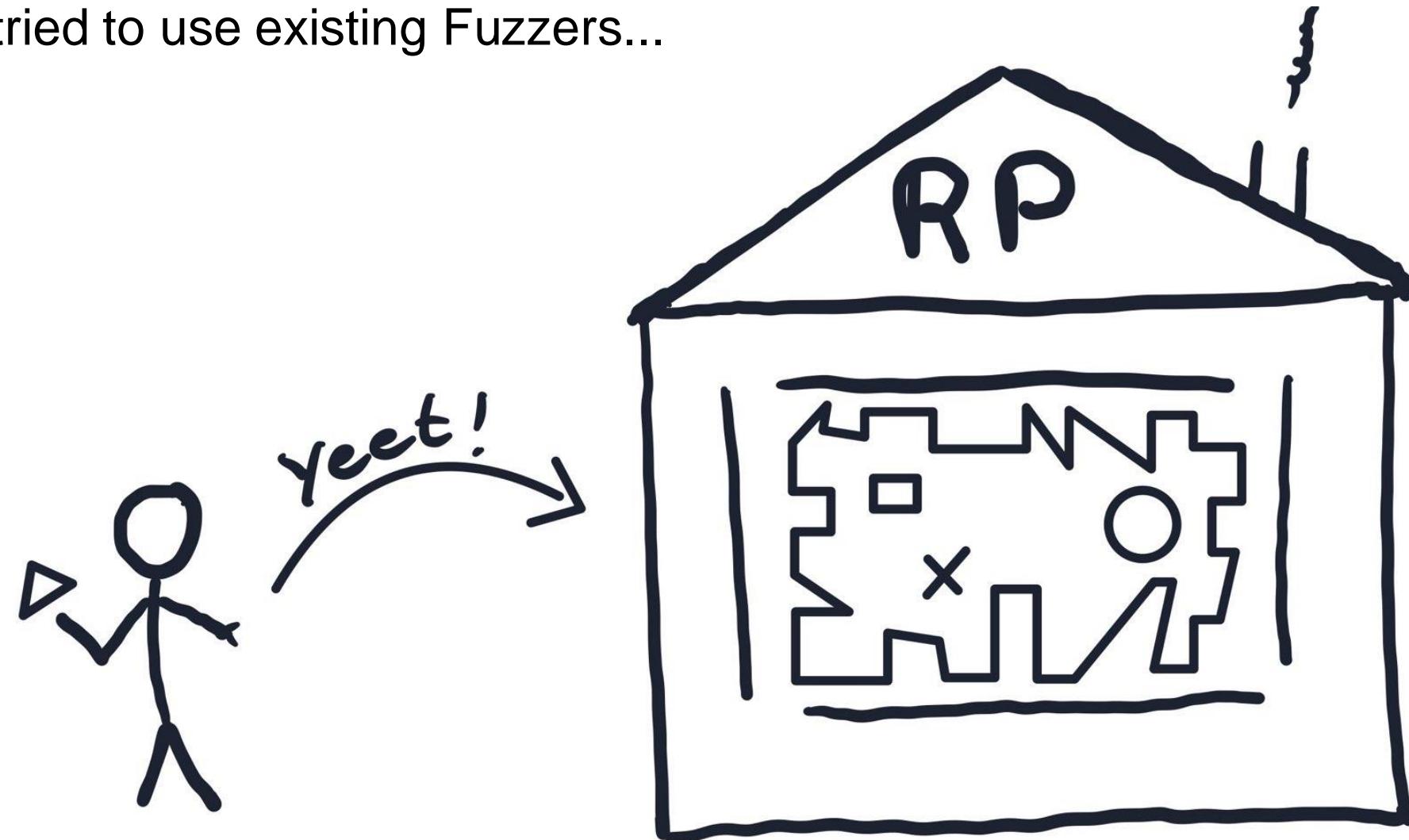
Routinator.log

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...

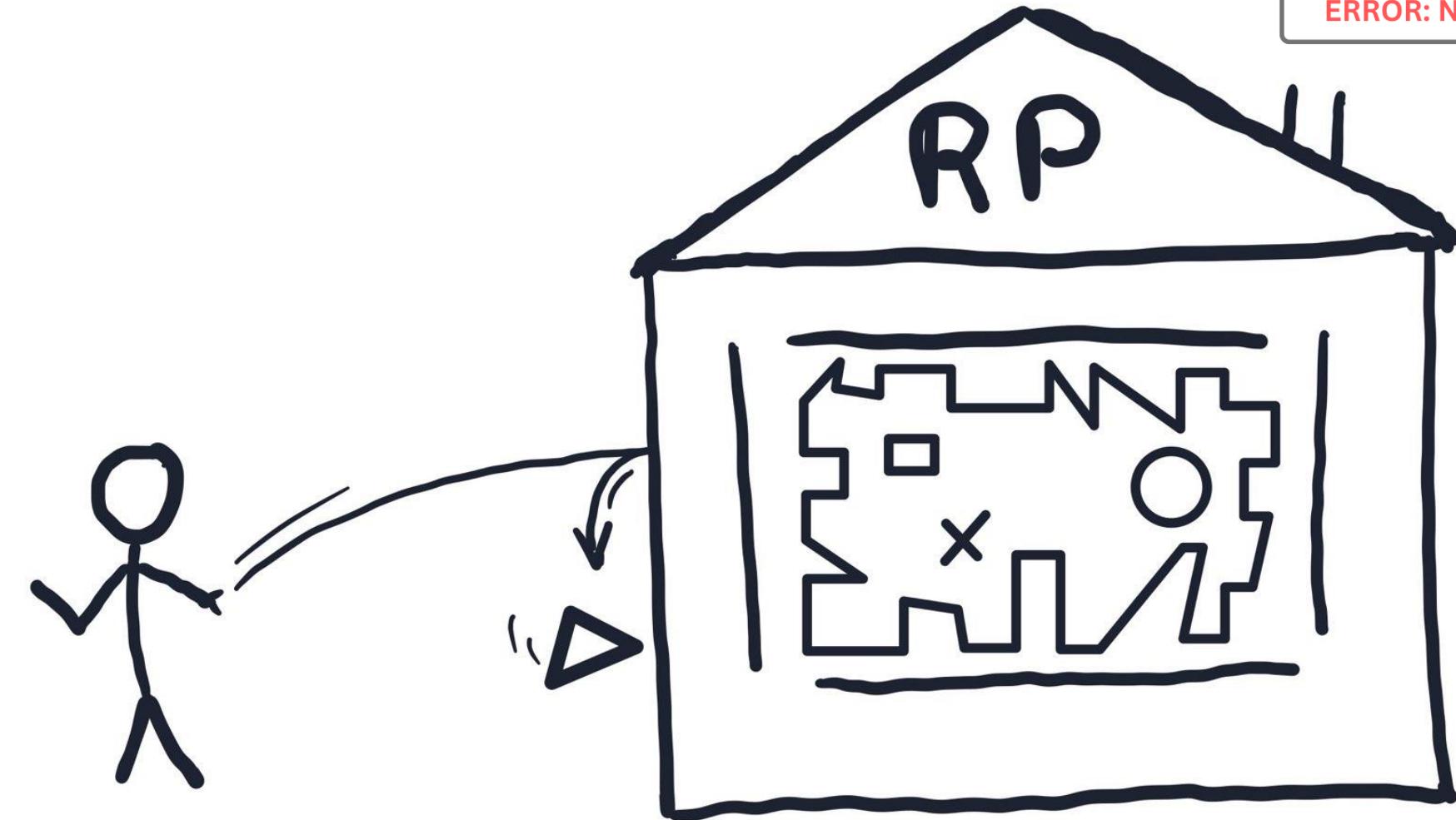


The complex Reality

- 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

```

RPKI-ROA { iso(1) member-body(2) us(840) rsadsi(113549)
  pkcs(1) pkcs9(9) smime(16) mod(0) 61 }

DEFINITIONS EXPLICIT TAGS ::= BEGIN

RouteOriginAttestation ::= SEQUENCE {
  version [0] INTEGER DEFAULT 0,
  asID ASID,
  ipAddrBlocks SEQUENCE (SIZE(1..MAX)) OF IPAddrBlock }

ASID ::= INTEGER

ROAIPAddressFamily ::= SEQUENCE {
  addressFamily OCTET STRING (SIZE (2..)),
  addresses SEQUENCE (SIZE (1..MAX)) OF ROAIPAddress }

ROAIPAddress ::= SEQUENCE {
  address IPAddress,
  maxLength INTEGER OPTIONAL }

IPAddress ::= BIT STRING

END

TBSCertList ::= SEQUENCE {
  version          Version OPTIONAL,
  -- if present
  signature        AlgorithmIdentifier,
  issuer           Name,
  thisUpdate       Time,
  nextUpdate       Time OPTIONAL,
  revokedCertificates SEQUENCE OF SEQUENCE {
    userCertificate CertificateSubjectName,
    revocationDate  Time,
    crlEntryExtensions Extensions OPTIONAL,
    -- if present
    } OPTIONAL,
  [0] Extensions OPTIONAL,
  -- if present
  -- contains the DER encoding of an ASN.1 value
  -- corresponding to the extension type identified
  -- by extnID
  }

Manifest ::= SEQUENCE {
  version      [0] INTEGER DEFAULT 0,
  manifestNumber INTEGER (0..MAX),
  thisTBScertificate ::= SEQUENCE {
    nextVersion [0] Version DEFAULT v1,
    serialNumber CertificateSerialNumber,
    signature   AlgorithmIdentifier,
    issuer      Name,
    validity    Validity,
    subject     Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version MUST be v2 or v3
    subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version MUST be v2 or v3
    extensions   [3] Extensions OPTIONAL
    -- If present, version MUST be v3 --
  }
  FileAndHash ::= SEQUENCE {
    file        file,
    hash        hash
  }
  Version ::= INTEGER { v1(0), v2(1), v3(2) }
  CertificateSerialNumber ::= INTEGER
  Validity ::= SEQUENCE {
    notBefore   Time,
    notAfter    Time }
  Time ::= CHOICE {
    utcTime    UTCTime,
    generalTime GeneralizedTime }
  UniqueIdentifier ::= BIT STRING
  SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm   AlgorithmIdentifier,
    subjectPublicKey BIT STRING }
  Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
  Extension ::= SEQUENCE {
    extnID     OBJECT IDENTIFIER,
    critical   BOOLEAN DEFAULT FALSE,
    extnValue  OCTET STRING
    -- contains the DER encoding of an ASN.1 value
    -- corresponding to the extension type identified
    -- by extnID
  }
}

```

Why is this so difficult

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

```
RPKI-ROA { iso(1) member-body(2) us(840) rsadsi(113549)
  pkcs(1) pkcs9(9) smime(16) mod(0) 61 }

DEFINITIONS EXPLICIT TAGS ::= BEGIN

RouteOriginAttestation ::= SEQUENCE {
  version [0] INTEGER DEFAULT 0,
  asID ASID,
  ipAddrBlocks SEQUENCE (SIZE(1..MAX)) }

ASID ::= INTEGER

ROAIPAddressFamily ::= SEQUENCE {
  addressFamily OCTET STRING (SIZE (2..)),
  addresses SEQUENCE (SIZE (1..MAX)) OF ROAIPAddress }

ROAIPAddress ::= SEQUENCE {
  address IPAddress,
  maxLength INTEGER OPTIONAL }

IPAddress ::= BIT STRING

END

TBSCertList ::= SEQUENCE {
  version Version OPTIONAL,
    -- if present
  signature AlgorithmIdentifier,
  issuer Name,
  thisUpdate Time,
  nextUpdate Time OPTIONAL,
  revokedCertificates SEQUENCE OF SEQUENCE {
    userCertificate CertificateSerialNumber,
    revocationDate Time,
    crlEntryExtensions Extensions OPTIONAL,
      -- if present
    } OPTIONAL,
    [0] Extensions OPTIONAL,
      -- if present
  }

crlExtensions
```

```
Manifest ::= SEQUENCE {
  version [0] INTEGER DEFAULT 0,
  manifestNumber INTEGER (0..MAX),
  thisTBScertificate ::= SEQUENCE {
    nextVersion [0] Version DEFAULT v1,
    serialNumber CertificateSerialNumber,
    signature AlgorithmIdentifier,
    issuer Name,
    validity Validity,
    subject Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,
      -- If present, version MUST be v2 or v3
    subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,
      -- If present, version MUST be v2 or v3
    extensions [3] Extensions OPTIONAL
      -- If present, version MUST be v3 -- }
  FileAndHash ::= SEQUENCE {
    file Hash,
    hash Hash }
  }

Version ::= INTEGER { v1(0), v2(1), v3(2) }

CertificateSerialNumber ::= INTEGER

Validity ::= SEQUENCE {
  notBefore Time,
  notAfter Time }

Time ::= CHOICE {
  utcTime UTCTime,
  generalTime GeneralizedTime }

UniqueIdentifier ::= BIT STRING

SubjectPublicKeyInfo ::= SEQUENCE {
  algorithm AlgorithmIdentifier,
  subjectPublicKey BIT STRING }

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension

Extension ::= SEQUENCE {
  extnID OBJECT IDENTIFIER,
  critical BOOLEAN DEFAULT FALSE,
  extnValue OCTET STRING
    -- contains the DER encoding of an ASN.1 value
    -- corresponding to the extension type identified
    -- by extnID }
```



```
30 82 04 8D 30 82 03 75 A0 03 02 01 02 02 02 04
EC 30 0D 06 09 2A 86 48 86 F7 0D 01 01 0B 05 00
30 33 31 31 30 2F 06 03 55 04 03 13 28 37 36 33
33 64 30 39 39 32 38 61 65 38 31 34 39 35 31 34
36 35 35 63 34 66 66 32 61 61 61 61 32 32 65 61
34 66 36 31 36 30 1E 17 0D 32 34 30 37 30 38 31
35 31 36 35 39 5A 17 0D 32 34 30 37 31 32 31 35
31 36 35 39 5A 30 33 31 31 30 2F 06 03 55 04 03
13 28 37 36 33 33 64 30 39 39 32 38 61 65 38 31
34 39 35 31 34 36 35 35 63 34 66 66 32 61 61 61
61 32 32 65 61 34 66 36 31 36 30 82 01 19 30 0D
06 09 2A 86 48 86 F7 0D 01 01 01 05 00 03 82 01
06 00 AE 6C AE 78 75 0F 1E 11 B0 96 30 0E 53 B5
E6 F8 DE 11 44 CA 61 2E 66 93 85 8C 02 EE 47 7F
79 E8 0E F2 28 D5 19 28 24 1A 21 16 FC 53 E0 78
72 41 FD CC 80 6F A1 79 3B C8 0E 46 13 2C 61 44
FE 87 4F 6A 12 E3 4B DD 80 40 8C B8 56 02 7E 85
... skipping 160 bytes ...
F3 48 6C C9 7F 1F 09 ED 2D 45 AA 38 AF C7 4E 8C
35 1B 02 03 01 00 01 A3 82 01 B2 30 82 01 AE 30
0F 06 03 55 1D 13 01 01 FF 04 05 30 03 01 01 FF
30 1D 06 03 55 1D 0E 04 16 04 14 76 33 D0 99 28
AE 81 49 51 46 55 C4 FF 2A AA A2 2E A4 F6 16 30
0E 06 03 55 1D 0F 01 01 FF 04 04 03 02 01 06 30
82 01 06 06 08 2B 06 01 05 05 07 01 0B 04 81 F9
30 81 F6 30 3D 06 08 2B 06 01 05 05 07 30 05 86
31 72 73 79 6E 63 3A 2F 2F 6D 79 2E 73 65 72 76
65 72 2E 63 6F 6D 2F 64 61 74 61 2F 6D 79 2E 73
65 72 76 65 72 2E 63 6F 6D 2F 72 65 70 6F 2F 74
61 2F 30 69 06 08 2B 06 01 05 05 07 30 0A 86 5D
72 73 79 6E 63 3A 2F 2F 6D 79 2E 73 65 72 76 65
```

CA Certificate
SignerName
SignerID
Validity
SubjectName
SubjectKey
SubjectID
IssuerRsync
Digest
CertSignature
DigestSignature
.....



It gets worse...

- RPKI uses...

It gets worse...

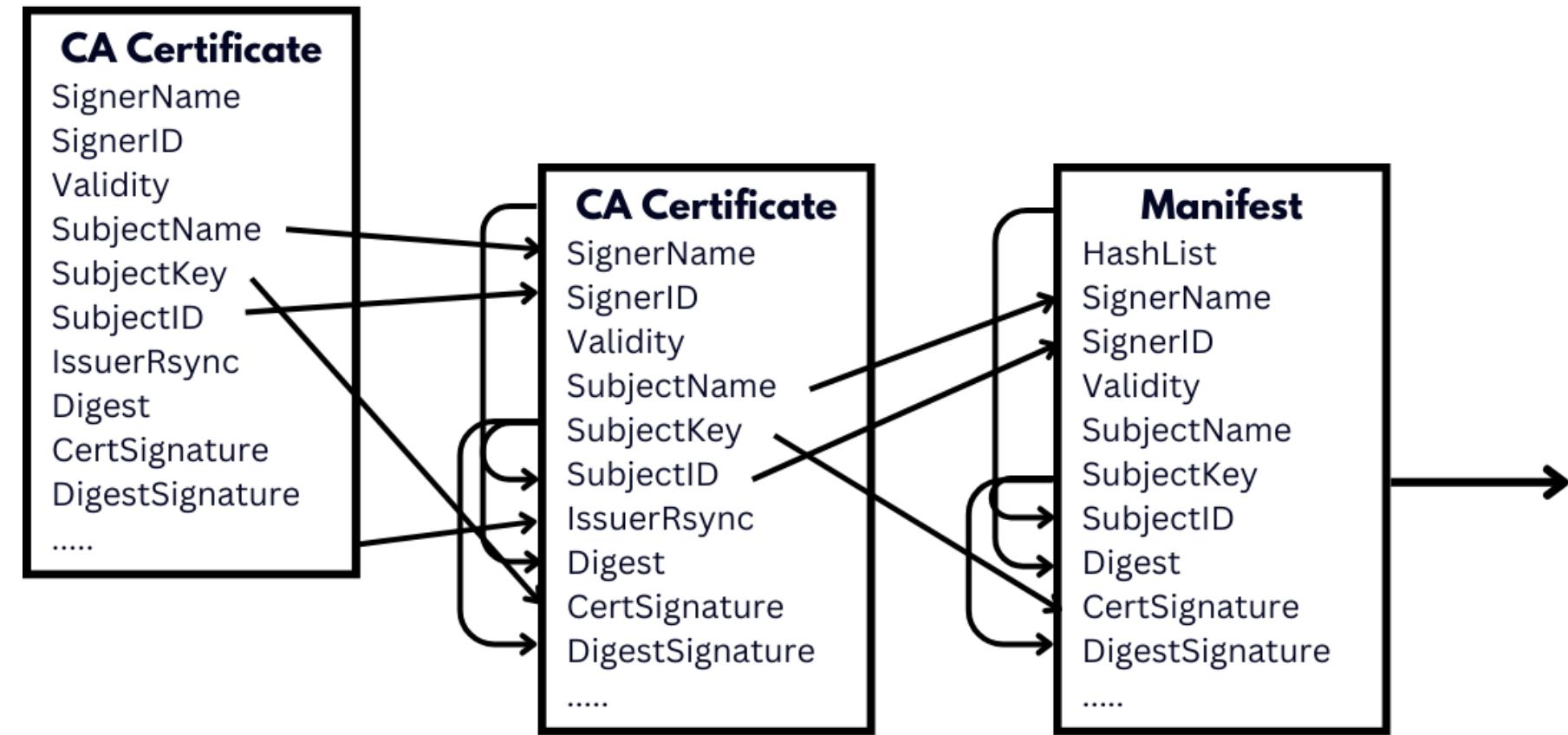
- RPKI uses...

CRYPTOGRAPHY



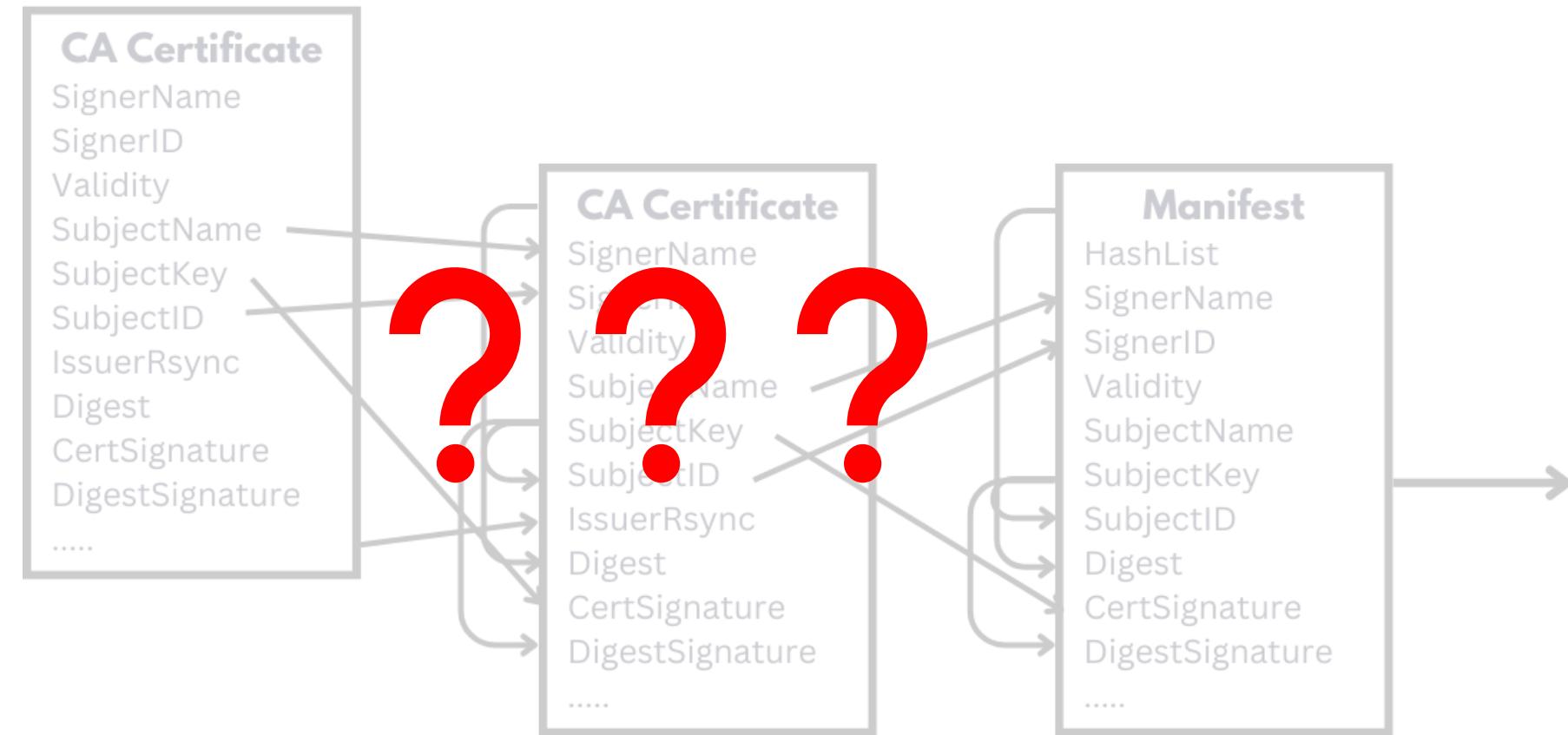
It gets worse...

- RPKI uses cryptography
- Fuzzers struggle with cryptography



It gets worse...

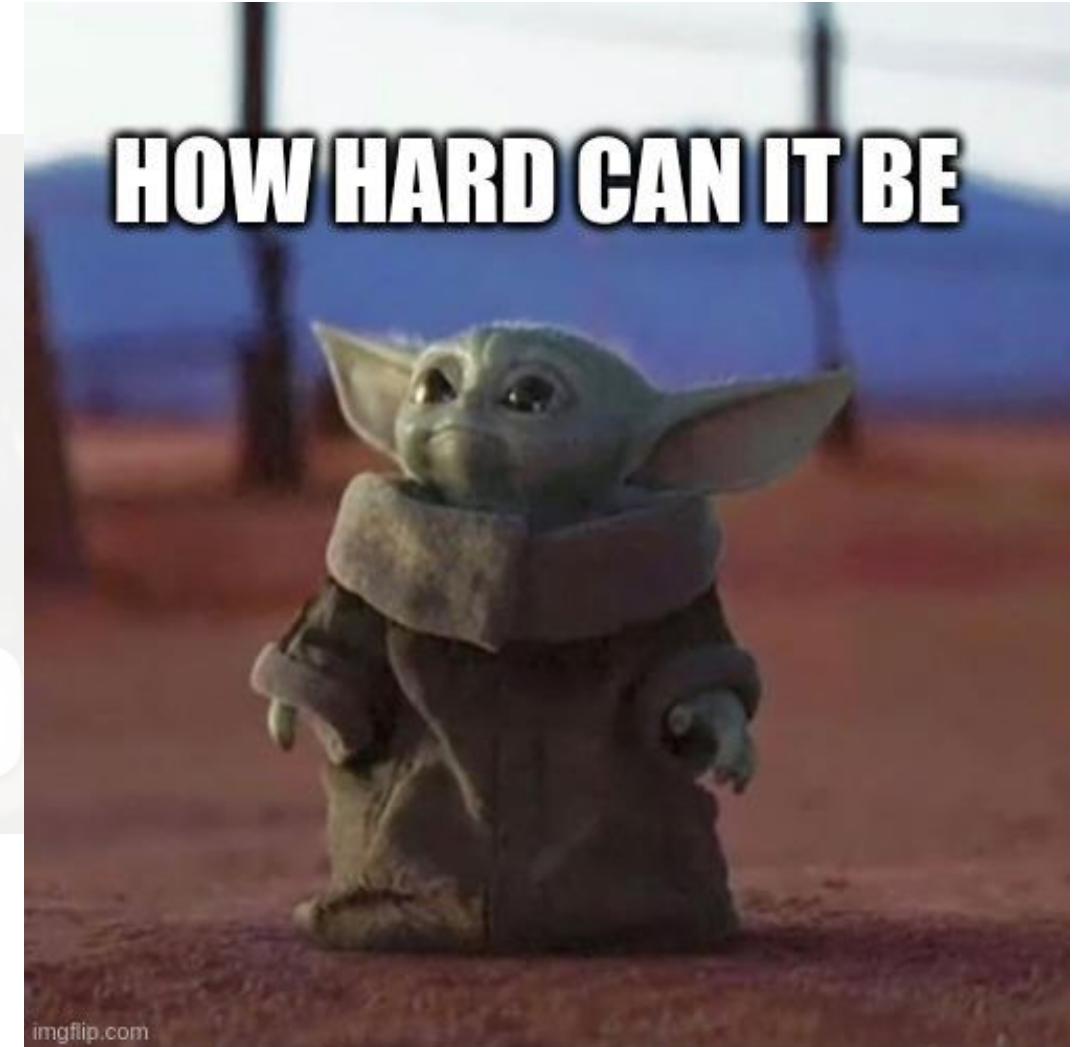
- RPKI uses cryptography
- Fuzzers struggle with cryptography



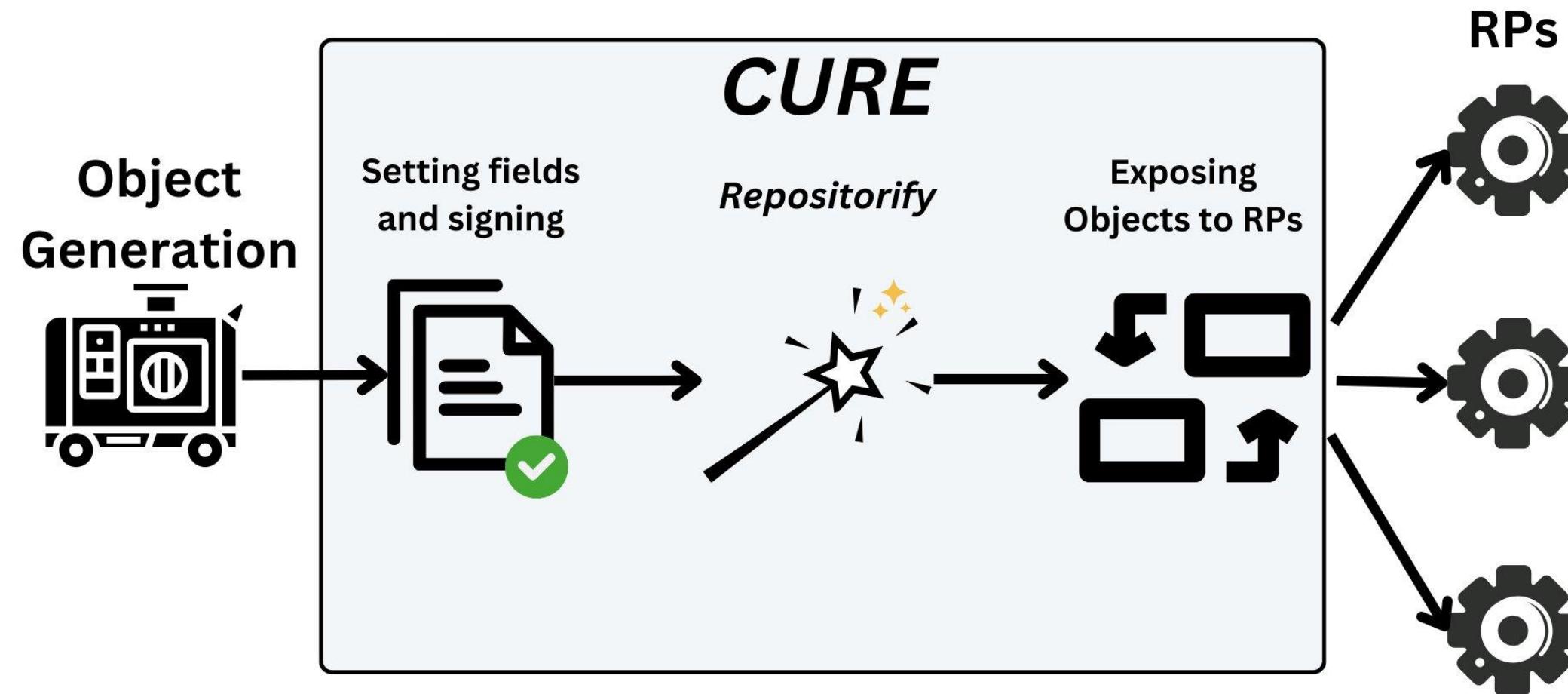
Only one solution...



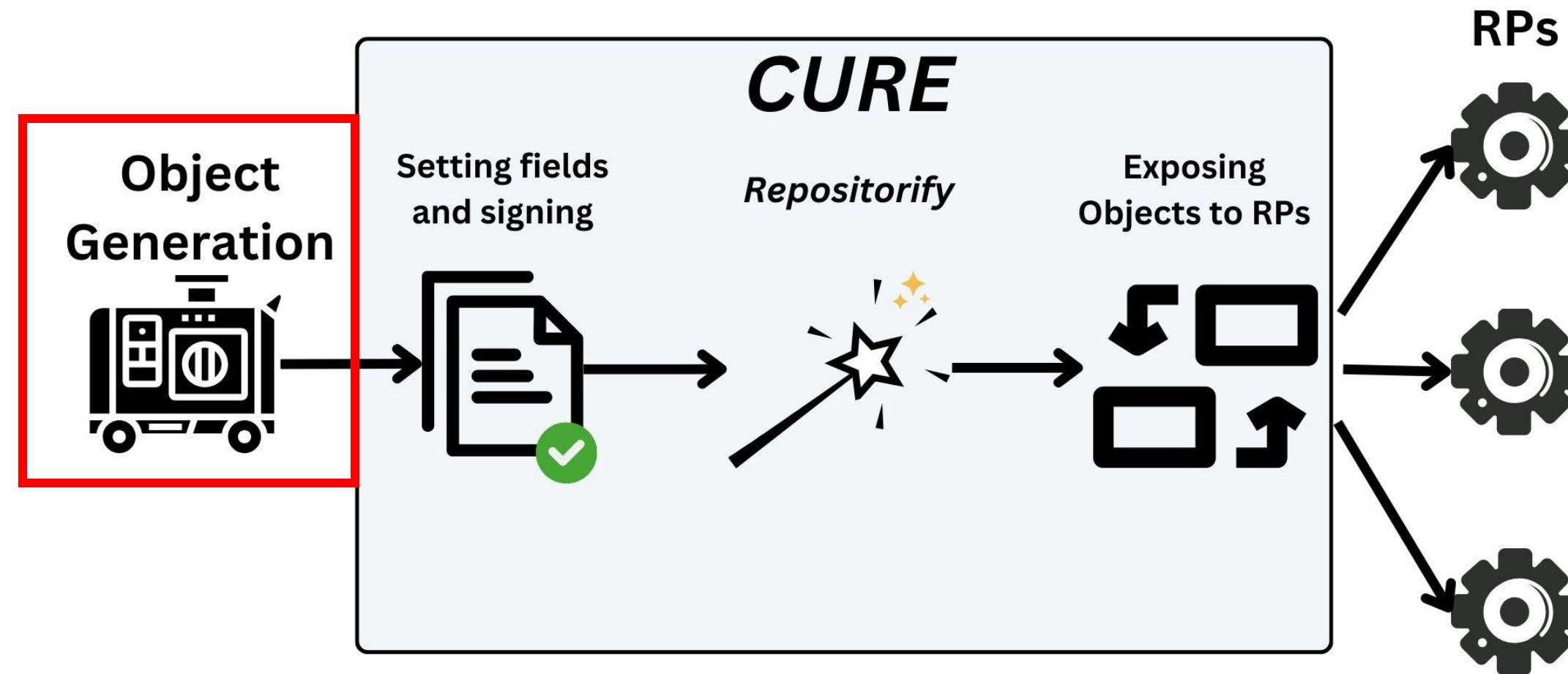
Only one solution...



Building yet another Fuzzer

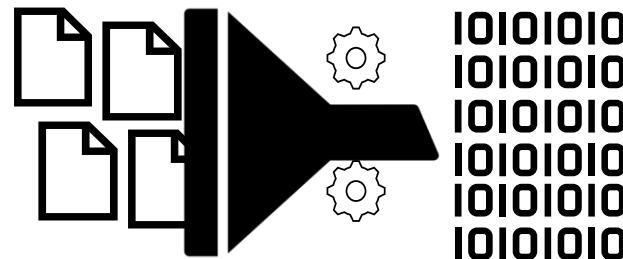


Building yet another Fuzzer



Object Generation

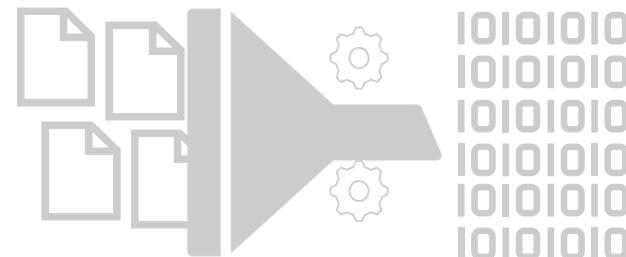
1. Random Byte Mutation



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

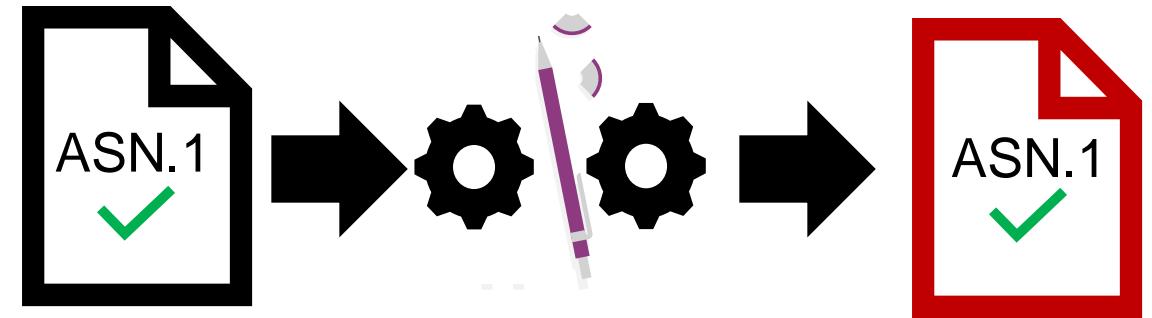
Object Generation

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

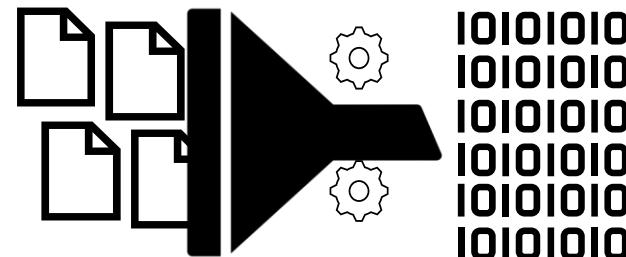
2. Structure Aware Mutation



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

Object Generation

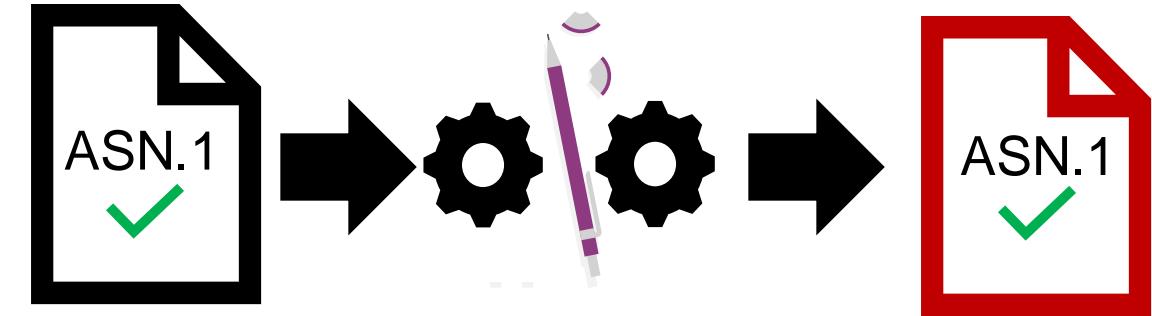
1. Random Byte Mutation



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

Found Bugs: 7

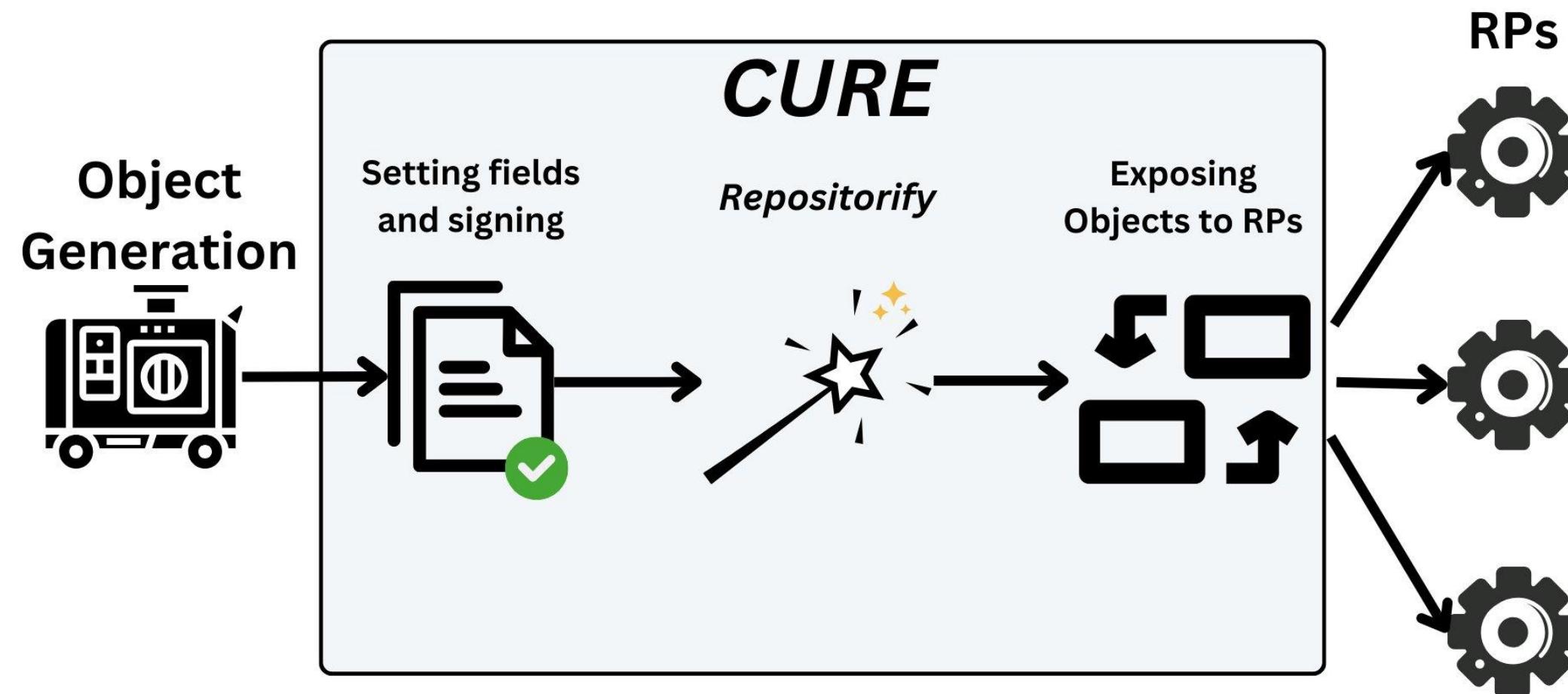
2. Structure Aware Mutation



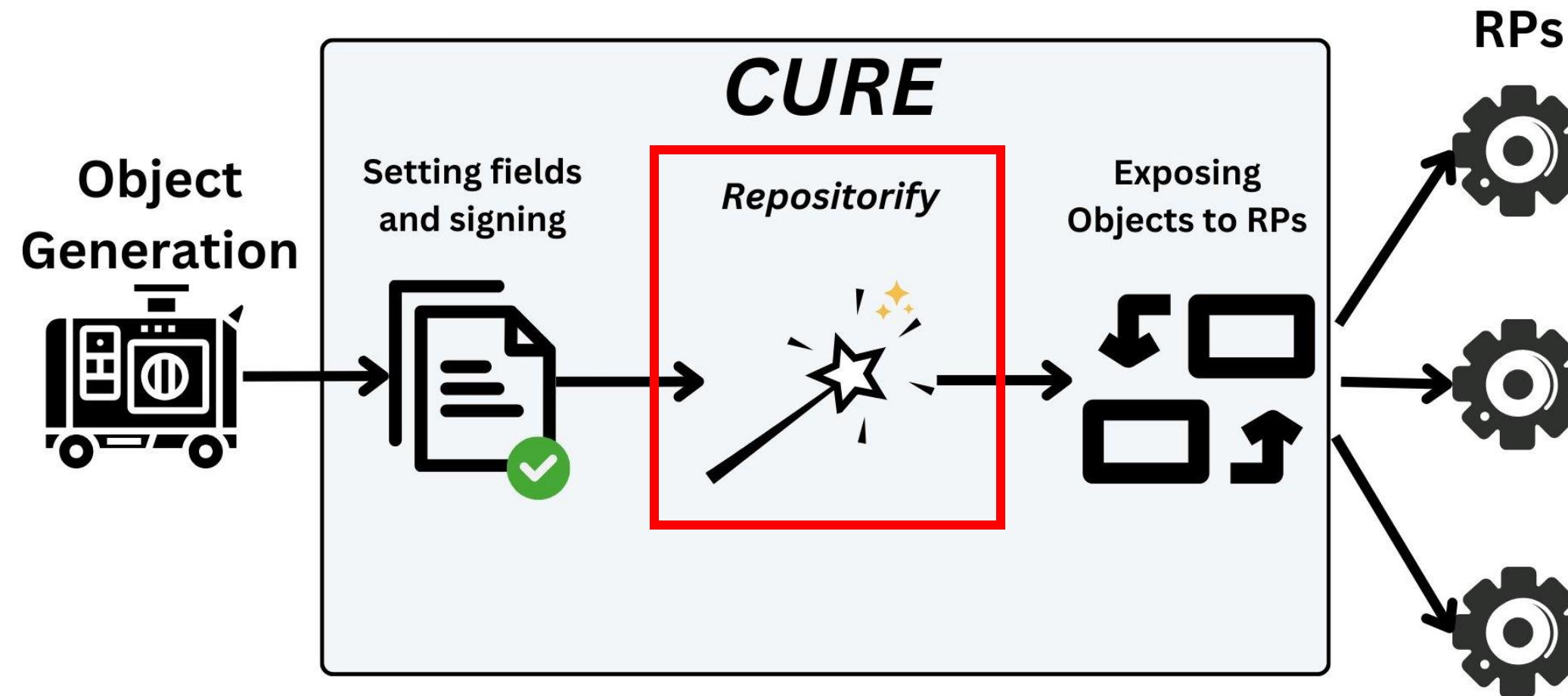
- i. schema-abiding, correctly encoded objects
- ii. manipulate content of fields
- iii. targets processing and validation logic

Found Bugs: 11

Repositorify Module

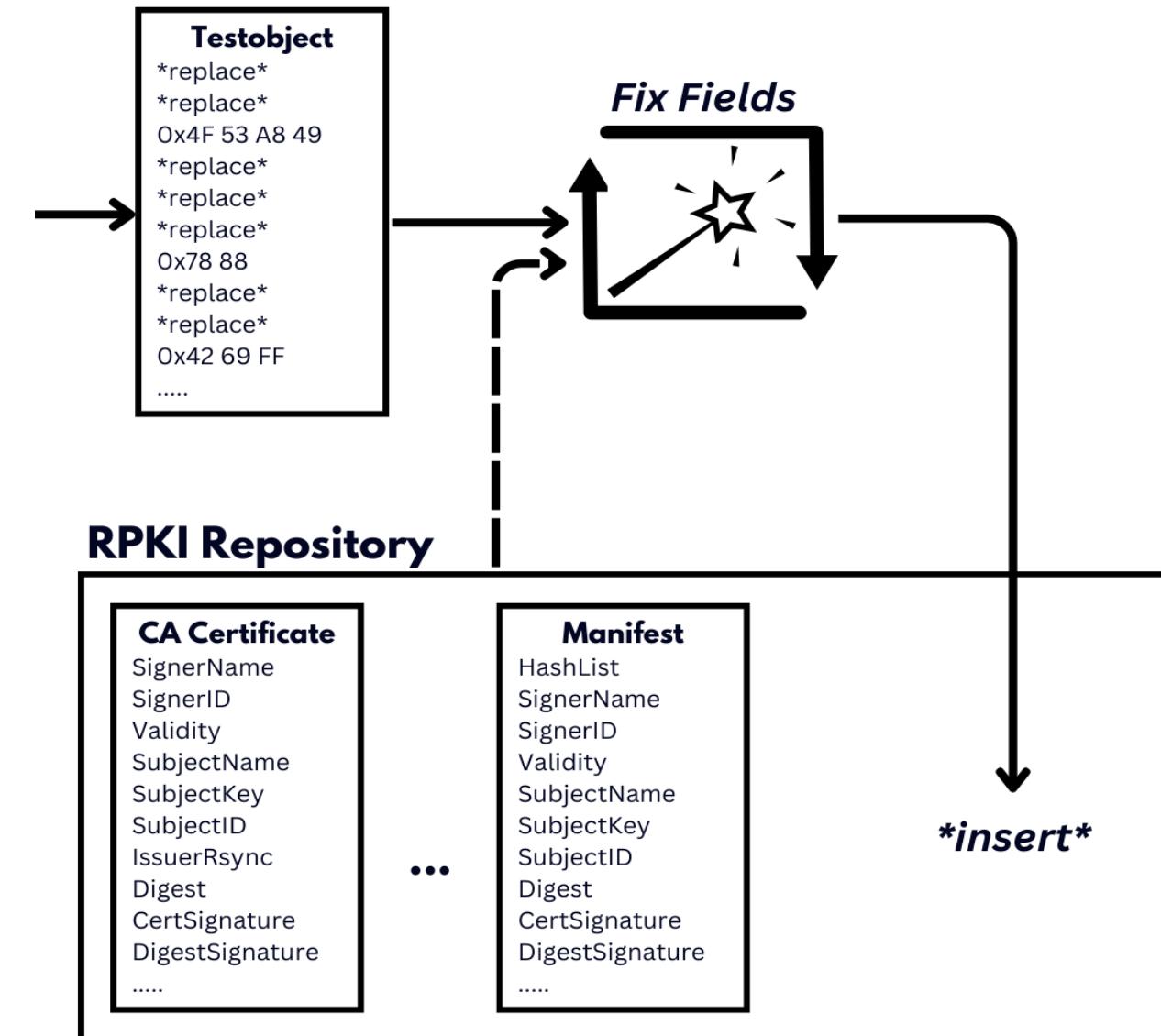


Repositorify Module



Repositorify Module

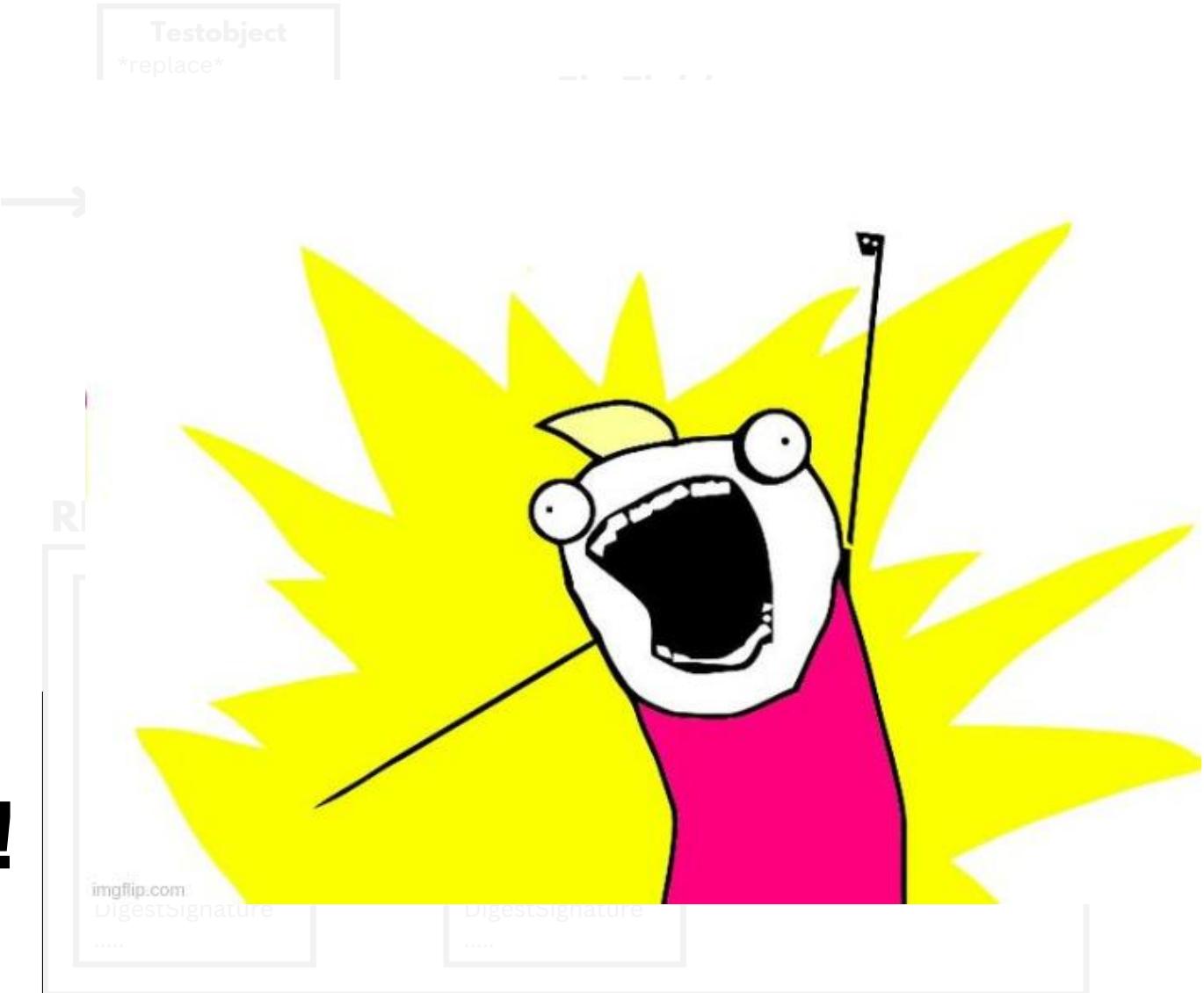
- Create valid RPKI repository
- Replace fields in objects
 - E.g. compute signatures
- Insert Test-Objects into repository



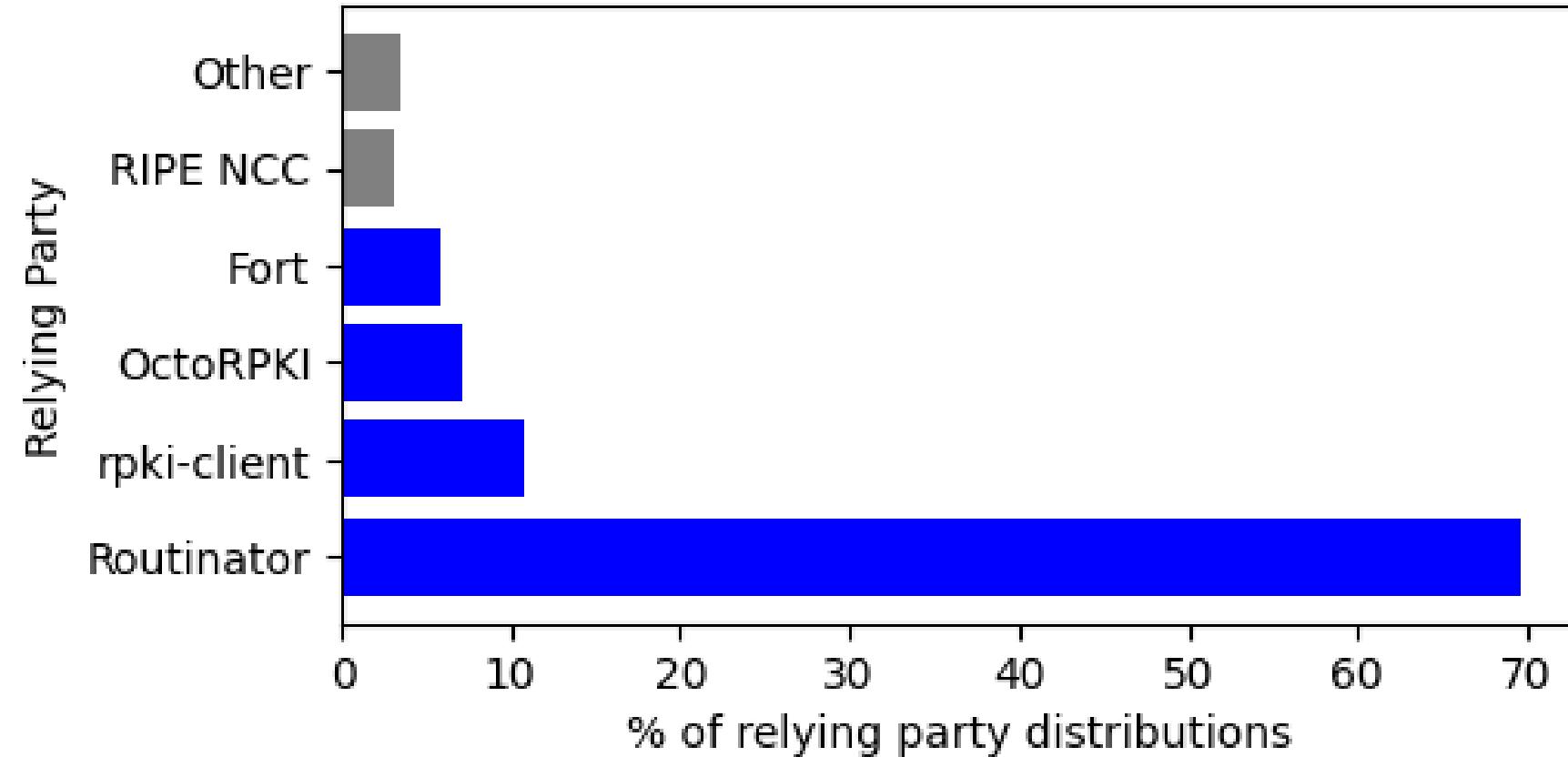
Repositorify Module

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

Let's find vulnerabilities!!

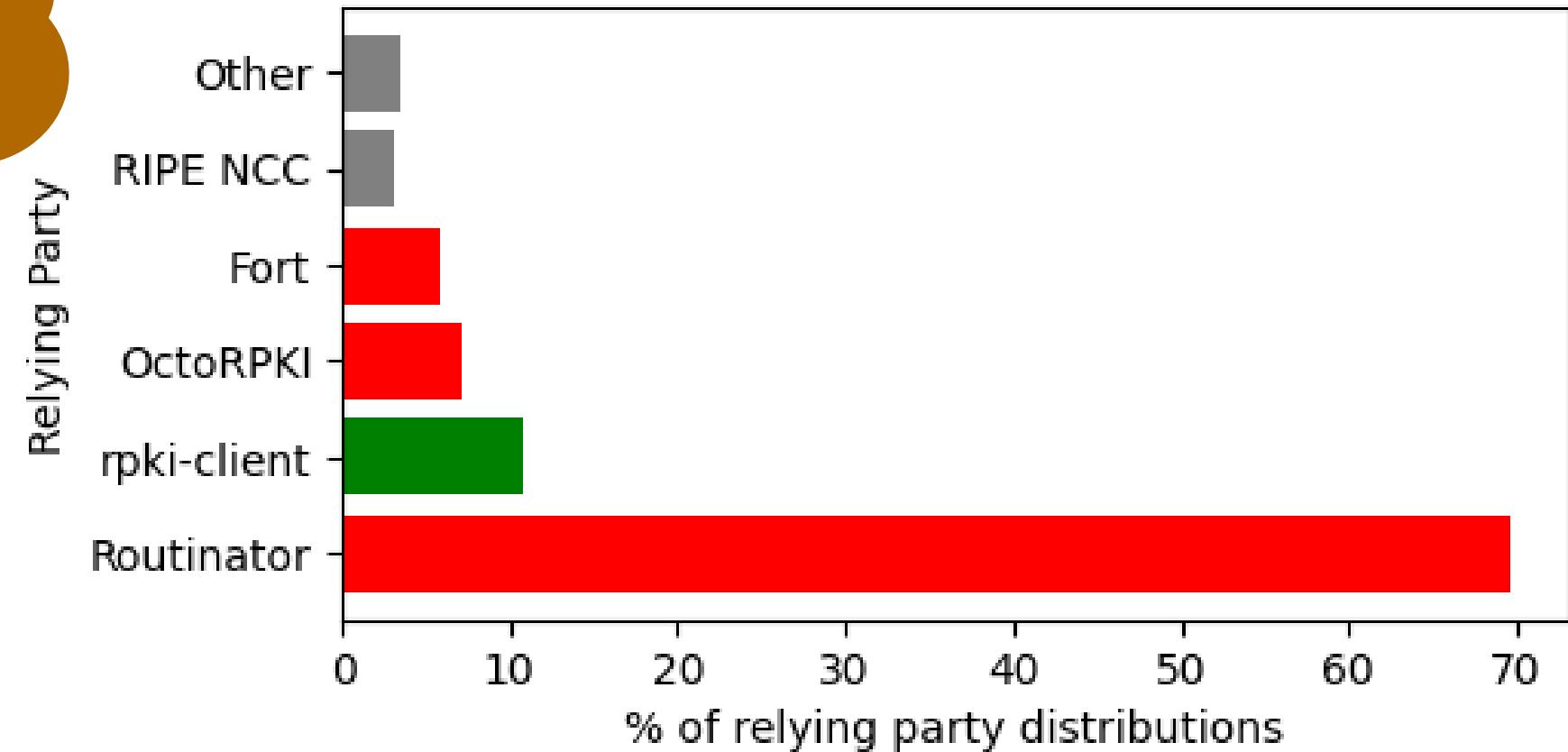


Relying Party Distributions



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
    url="https://server.com/data/../../fake.TAL"
    hash="33f969c5b6fd9ab501f9def2d47f7576ba80
          Oa91d09d34a080ed2cf90a86d1ec"
/>
</notification>
```

Vulnerability Type: DoS

- Adversary can create objects of any format

Vulnerability Type: DoS

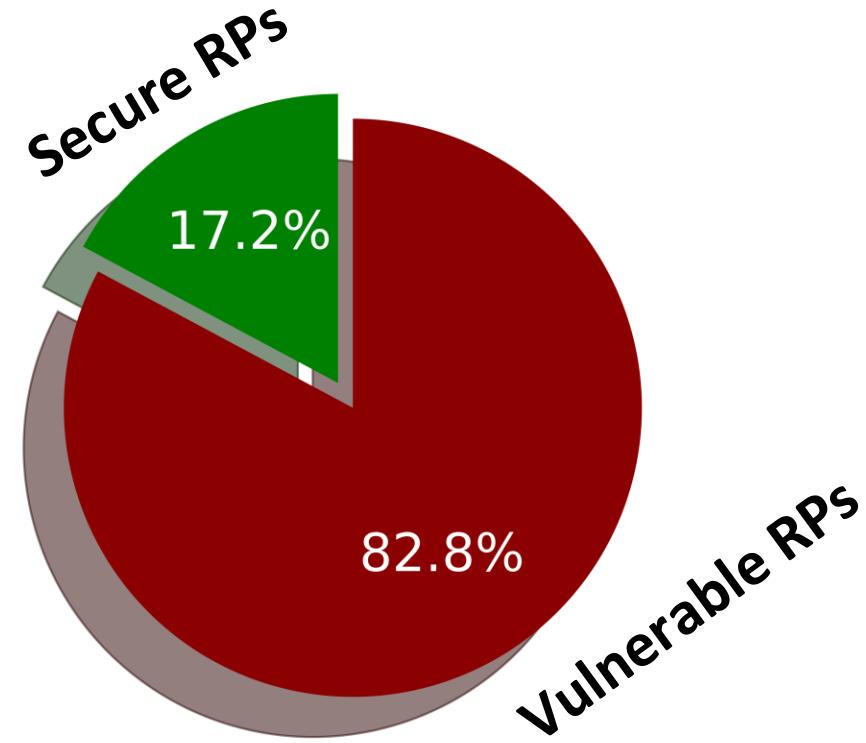
- Adversary can create objects of any format
- Vulnerable Software:
 - Routinator: Parsing of ASN.1 Data
 - OctoRPKI: Processing of Object Fields
 - Fort: Processing of RTR Requests
- Exploit:
Adversary forces RPs in perpetual fail-and-restart mode

Routinator.log

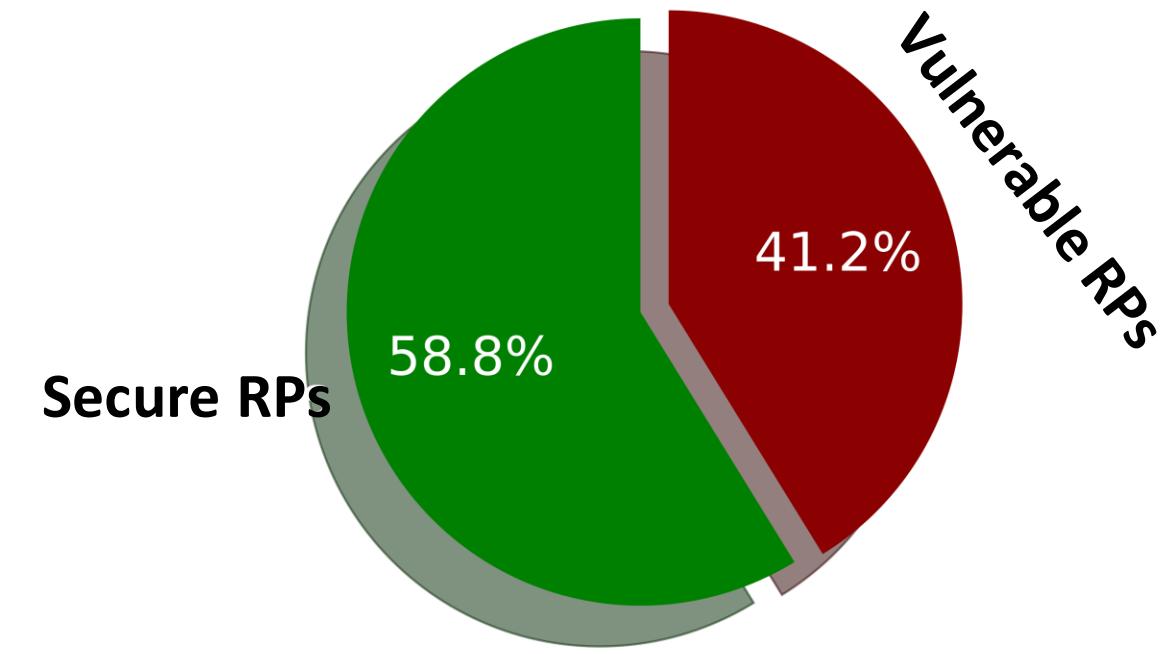
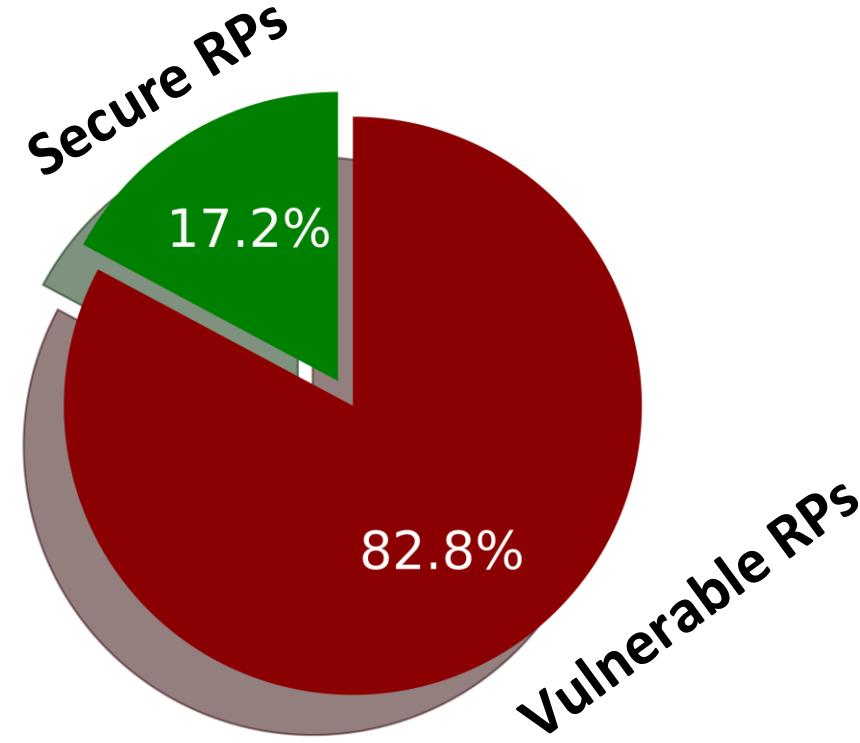
```
thread '<unnamed>' panicked at 'index out of bounds:
the len is 2 but the index is 2',
bcder/src/tag.rs:line:column
note: run with `RUST_BACKTRACE=1` environment
variable to display a backtrace
Aborted
```

Internet Evaluations

Internet Evaluations (Then)

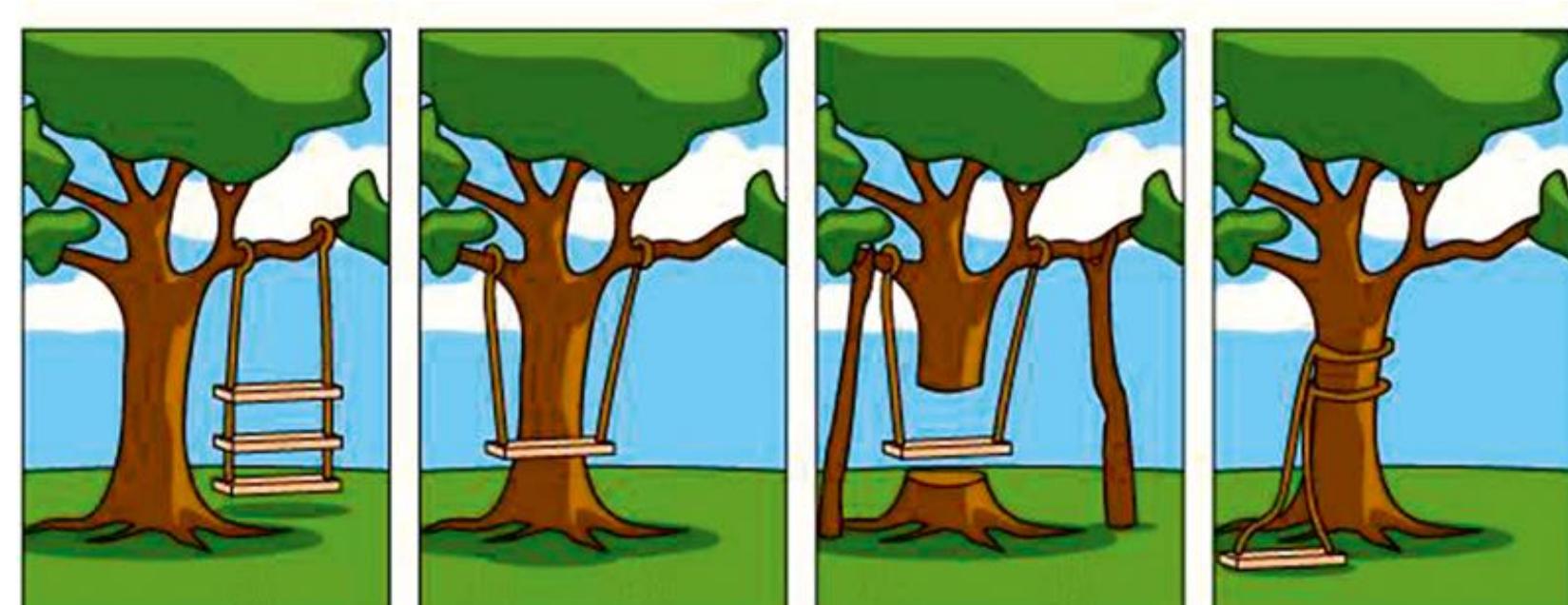


Internet Evaluations (Now)



Results: Global Inconsistencies

Results: Global Inconsistencies



HOW THE RFC
EXPLAINED IT

HOW ROUTINATOR
UNDERSTOOD IT

HOW OCTORPKI
UNDERSTOOD IT

HOW FORT
UNDERSTOOD IT

Results: Global Inconsistencies

➤ Post-processing ROA Payload:

Routinator: 441,770

OctoRPKI: 434,074

Fort: 435,002

rpkiclient: 441,777

Results: Global Inconsistencies

- Post-processing ROA Payload:

Routinator: 441,770

Fort: 435,002

OctoRPKI: 434,074

rpkiclient: 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

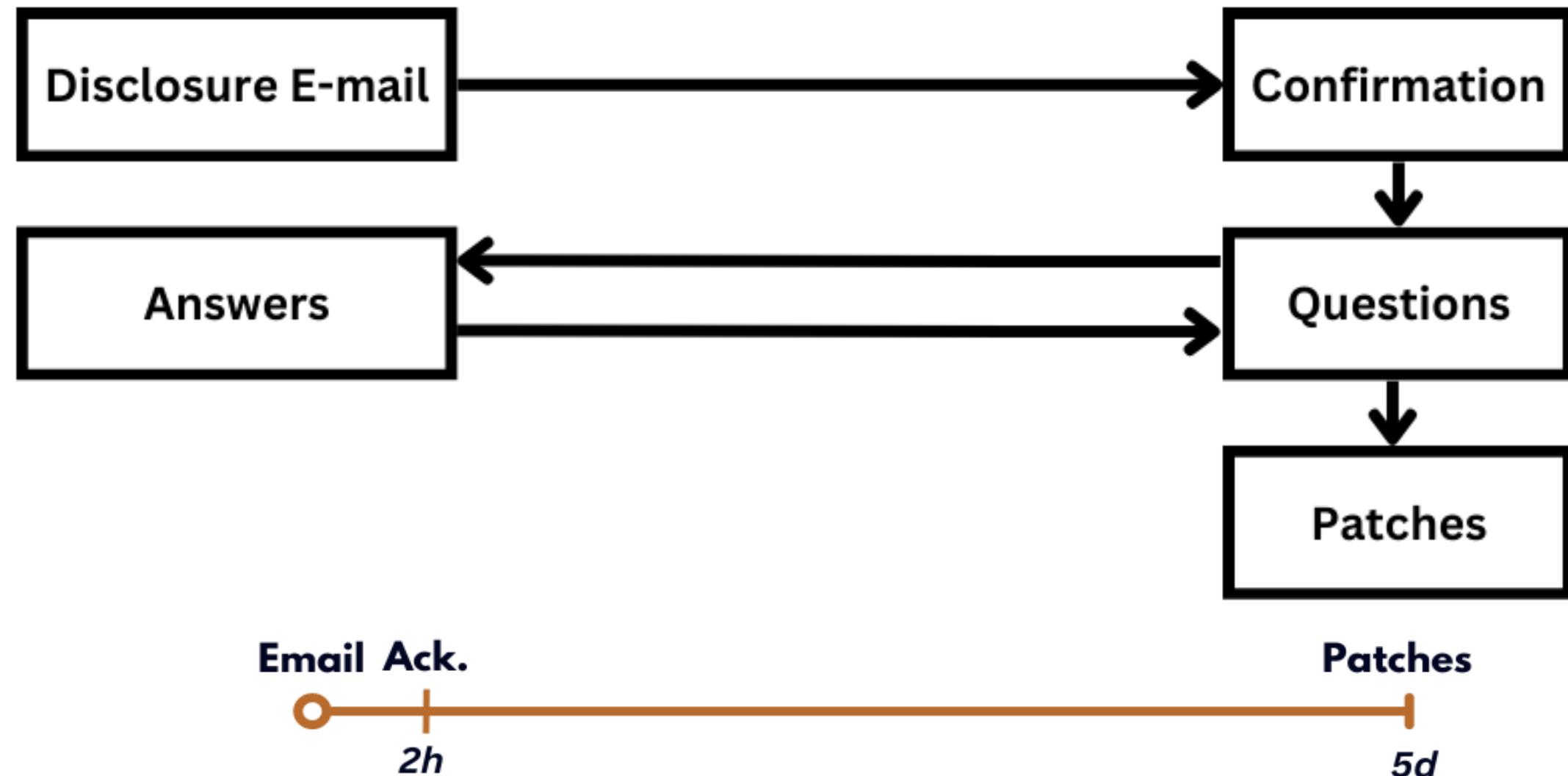


Vulnerabilities in Fort

Sent: Jul 20th '23 - 11:56

The experience differed significantly between vendors...

Disclosure – Vendor 1



Disclosure – Vendor 1

Disclosure E-mail

That was nice!

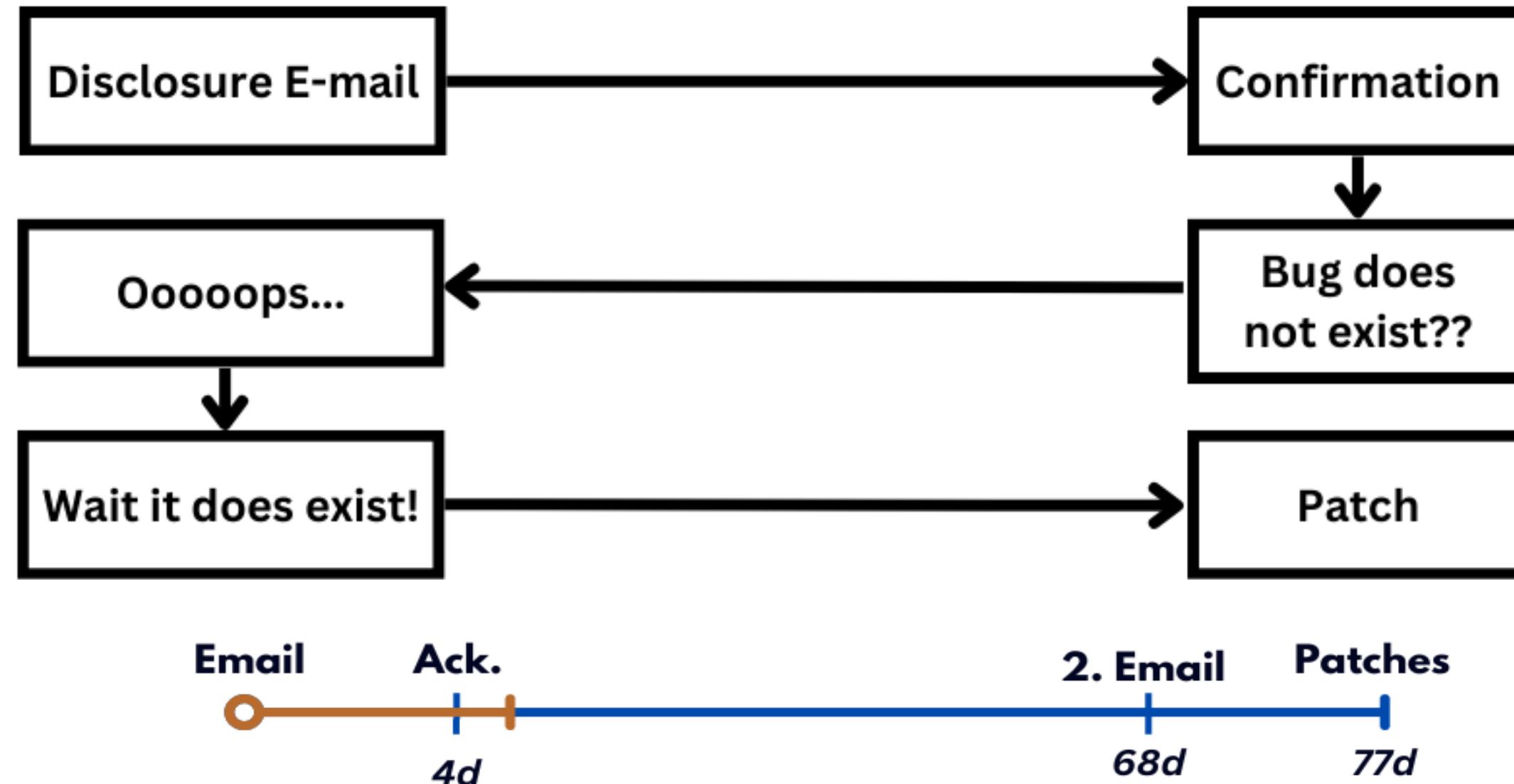
Email Ack.



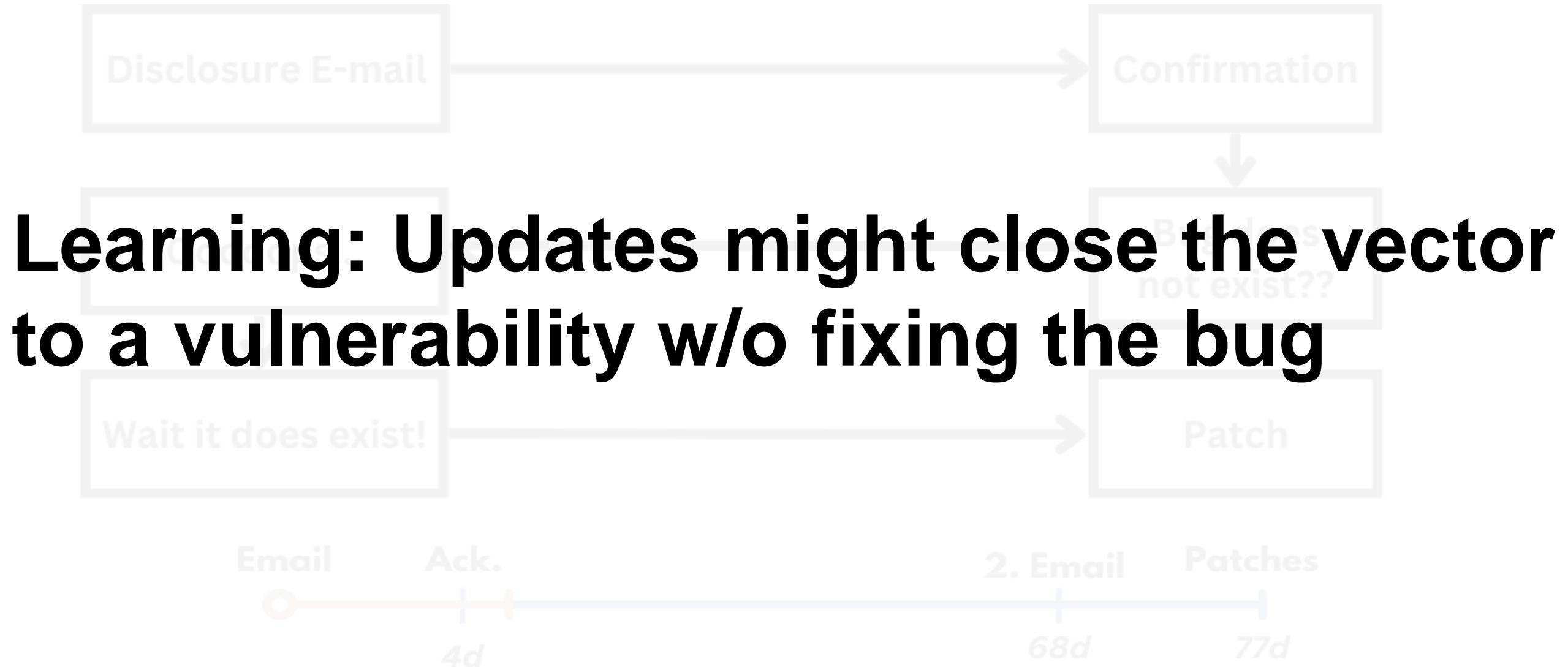
2h

5d

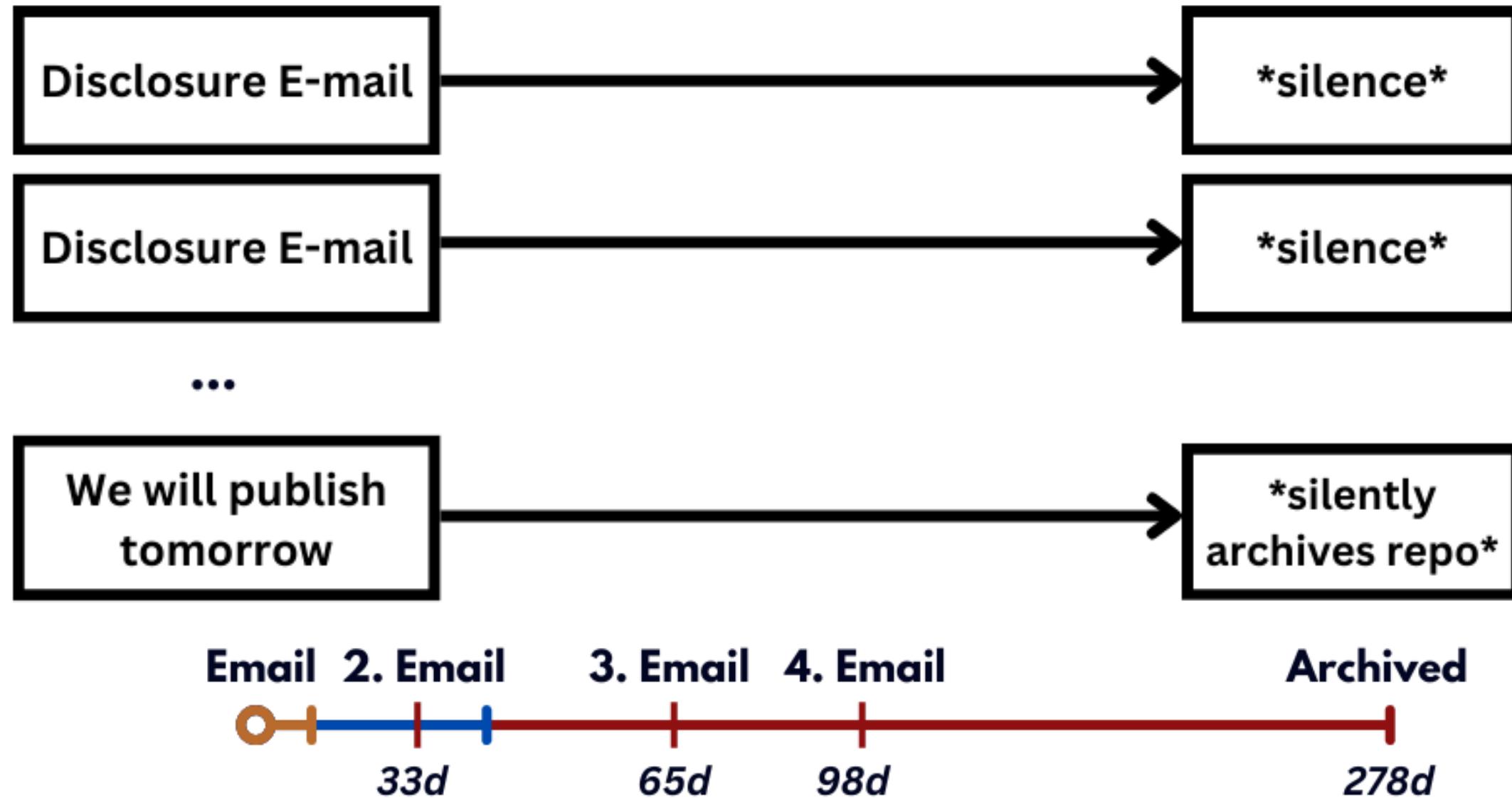
Disclosure – Vendor 2



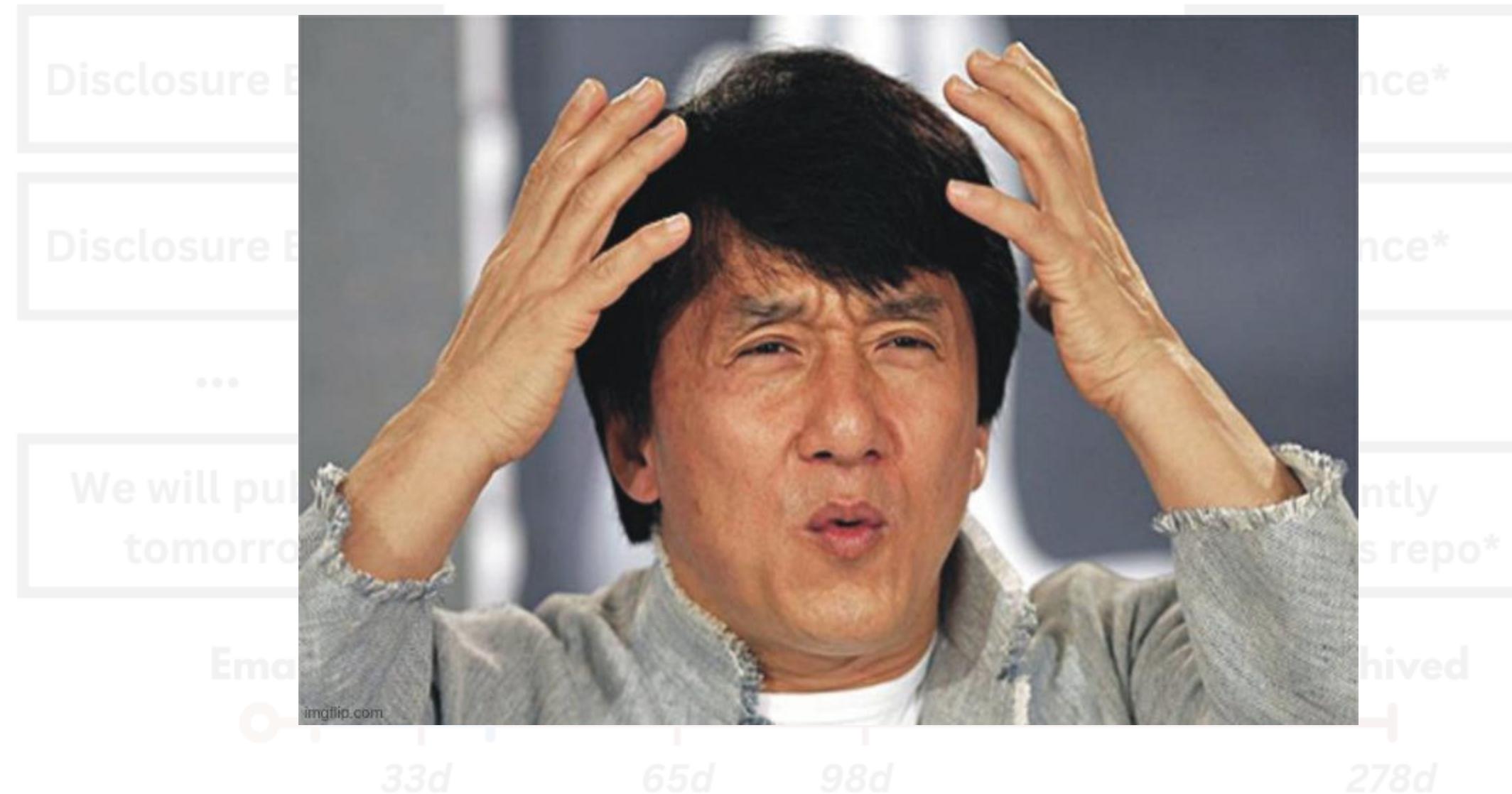
Disclosure – Vendor 2



Disclosure – Vendor 3



Disclosure – Vendor 3



Disclosure – Vendor 3

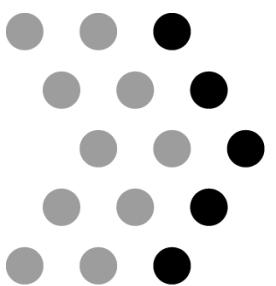


Lessons Learned

- **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