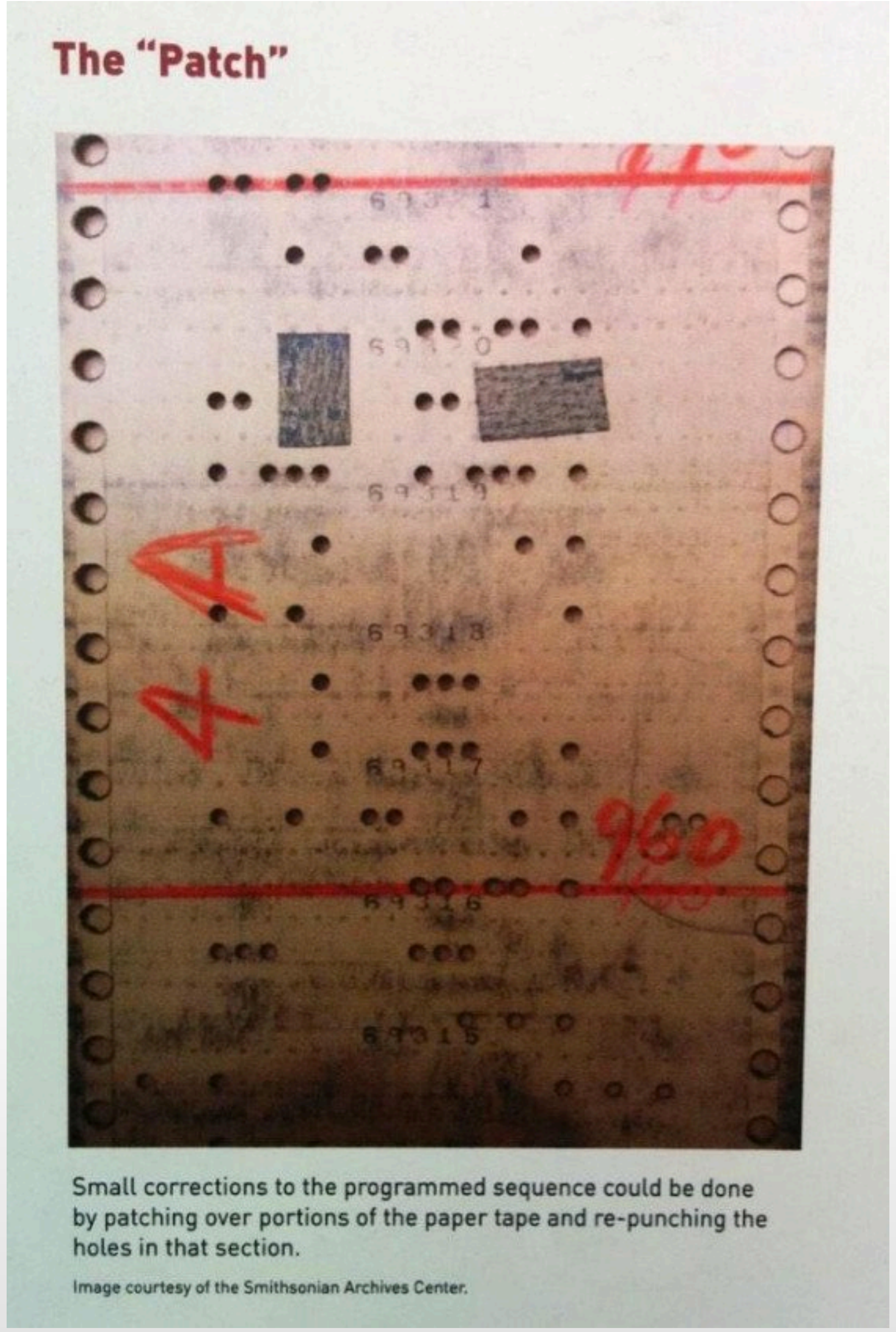
CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds

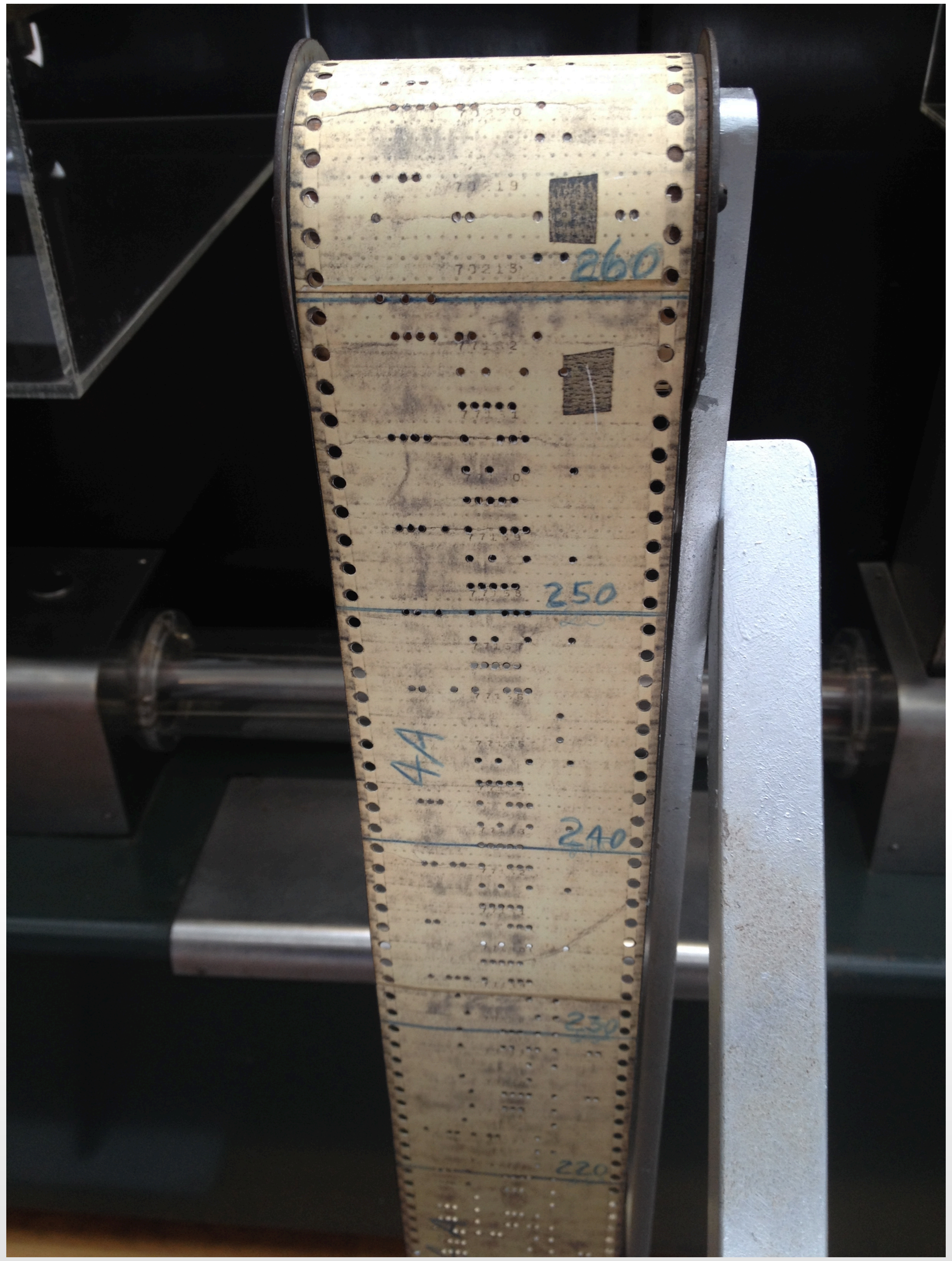
[1]**Kirill Nikitin**, [1]Eleftherios Kokoris-Kogias, [1]Philipp Jovanovic, [1]Linus Gasser, [1]Nicolas Gailly, [2]Ismail Khoffi, [3]Justin Cappos, [1]Bryan Ford

[1]École polytechnique fédérale de Lausanne (EPFL)
[2]University of Bonn
[3]New York University

Software Updates



Hilary Mason's Twitter



A program tape for the 1944 Harvard Mark I, one of the first digital computers. Wikipedia.

Software Updates

- Softwares updates are used to patch disclosed vulnerabilities, add new features, and improve security posture
- If you *do not* update your system, things can go bad…

MALICIOUS VIRUS What is Wannacry ransomware? Malware used to cripple NHS in 2017 cyber attack

More than 200,000 victims in around 150 countries have been infected by malicious software

By Gemma Mullin and Emma Lake
4th August 2017, 8:30 am | Updated: 4th August 2017, 11:25 am

The Sun

Forbes / Asia / #CyberSecurity

JUN 22, 2017 @ 05:00 AM 7,003 12 Stocks to Buy Now

Cyber Attack At Honda Stops Production After WannaCry Worm Strikes

Peter Lyon, CONTRIBUTOR
I write about automobiles and games.
FULL BIO

Forbes

US & WORLD TECH CYBERSECURITY

Australian police blame WannaCry for spoiling 8,000 traffic cam tickets

by Jacob Kastrenakes | Jun 27, 2017, 5:13pm EDT

The Verge

Software Updates

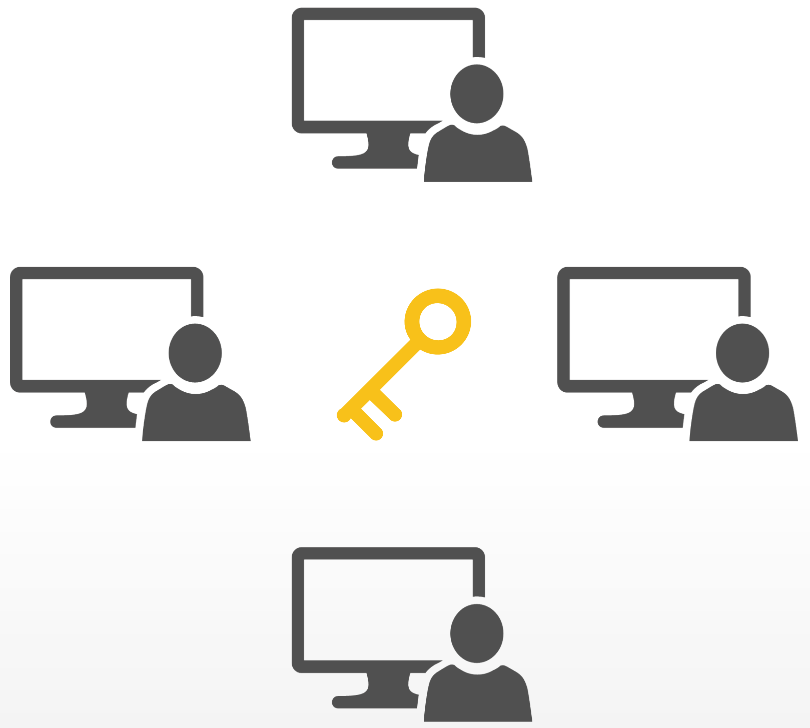
- But even if you *do* update your system regularly, things can go wrong too…
- Software-update systems are a lucrative attack target due to their centralized design and potential impact on users

How can we make software-update systems more secure and transparent?

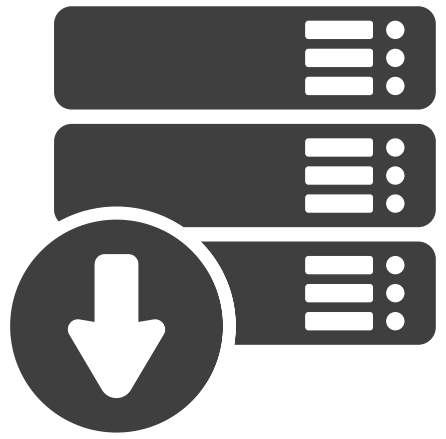
Software Release Pipeline

Development/Review – Building release binaries – Sign-off – Release distribution

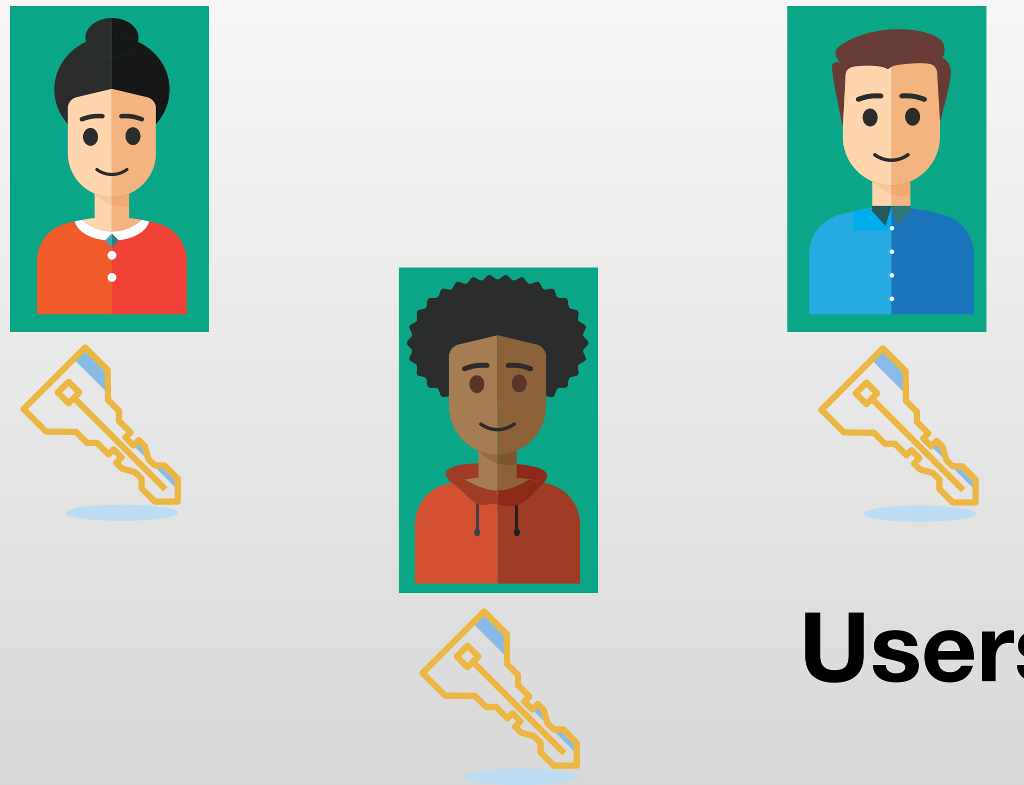
Developers



</CODE>



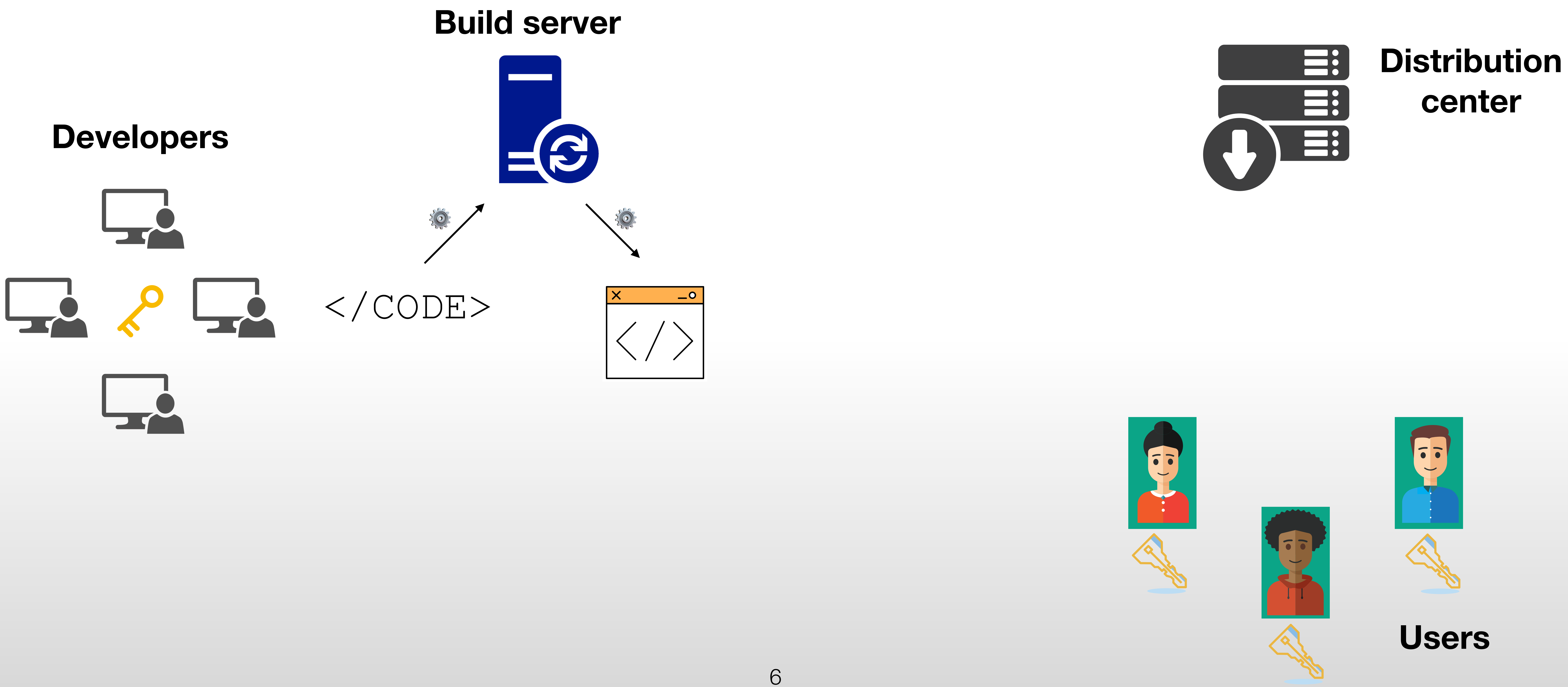
Distribution center



Users

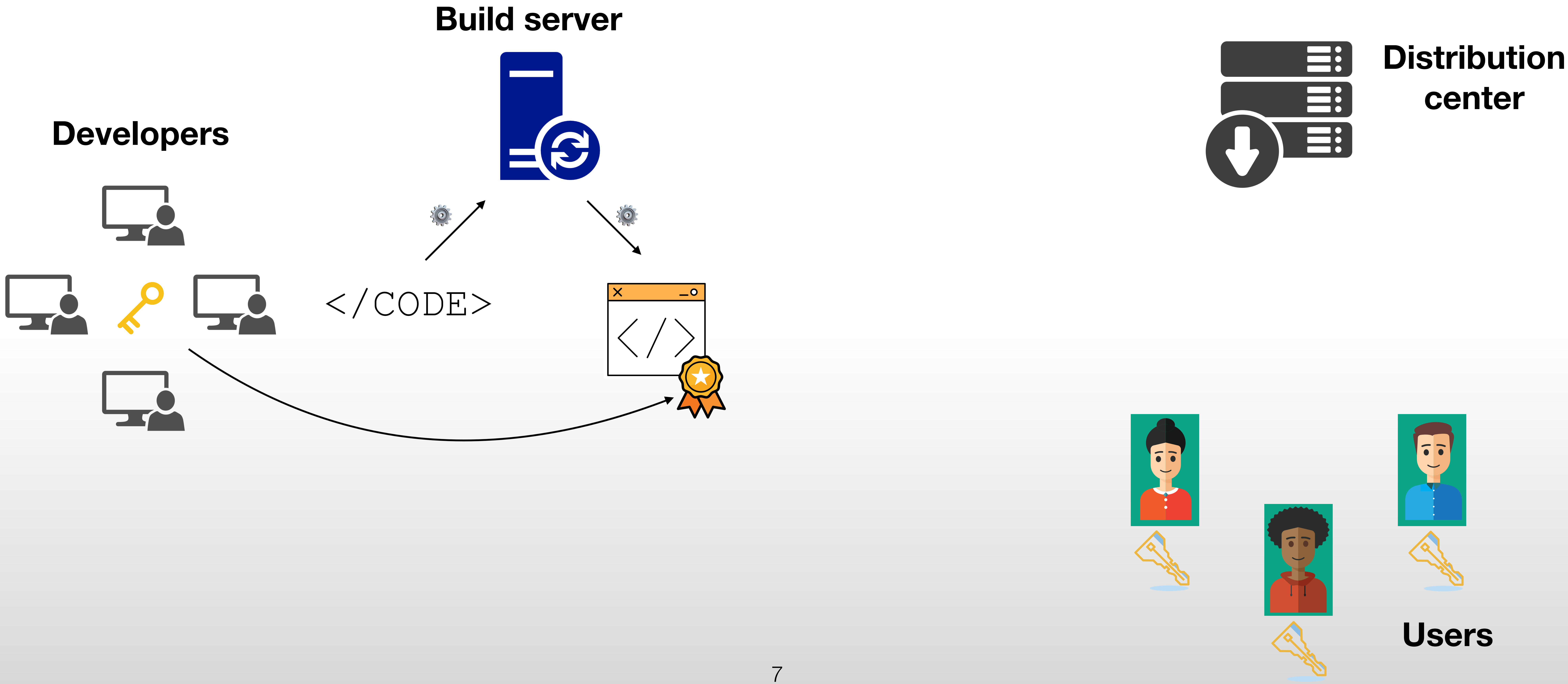
Software Release Pipeline

Development/Review – **Building release binaries** – Sign-off – Release distribution



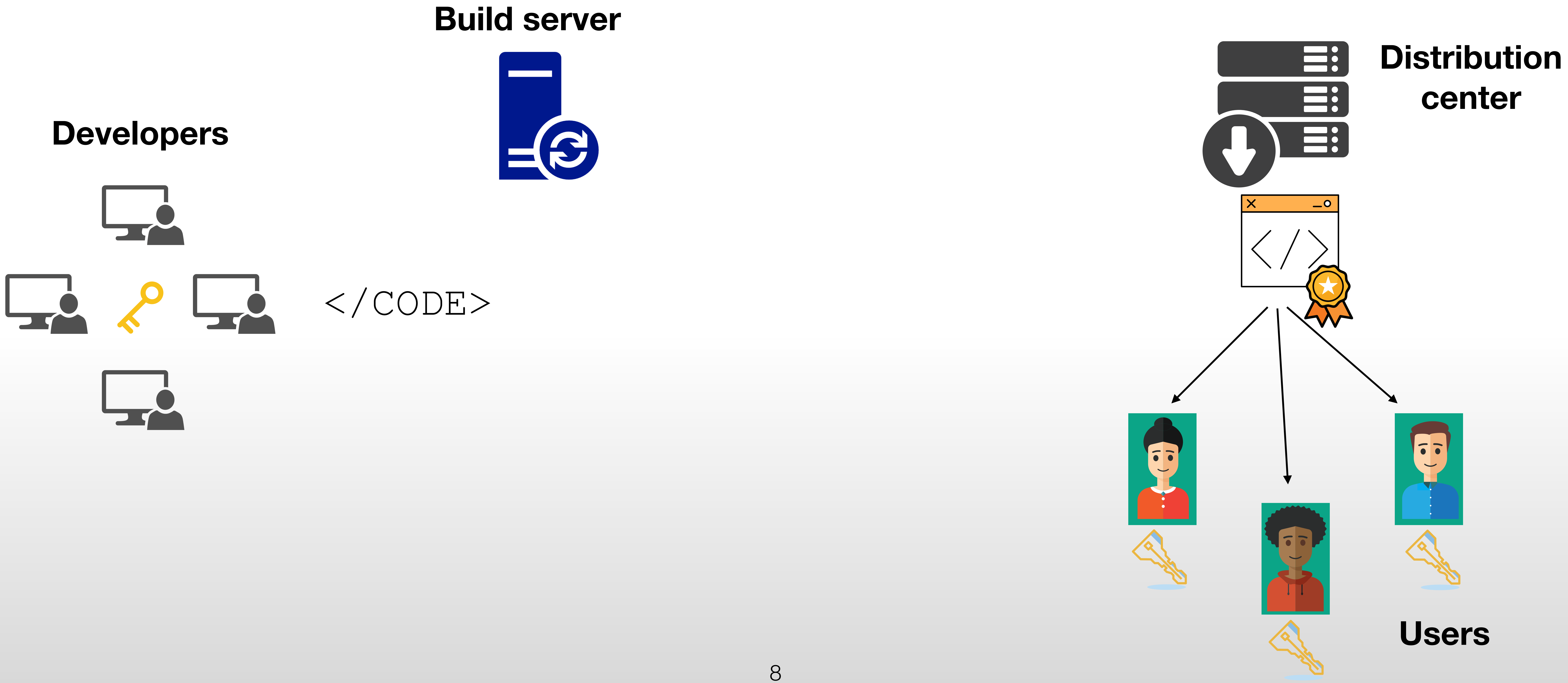
Software Release Pipeline

Development/Review – Building release binaries – **Sign-off** – Release distribution



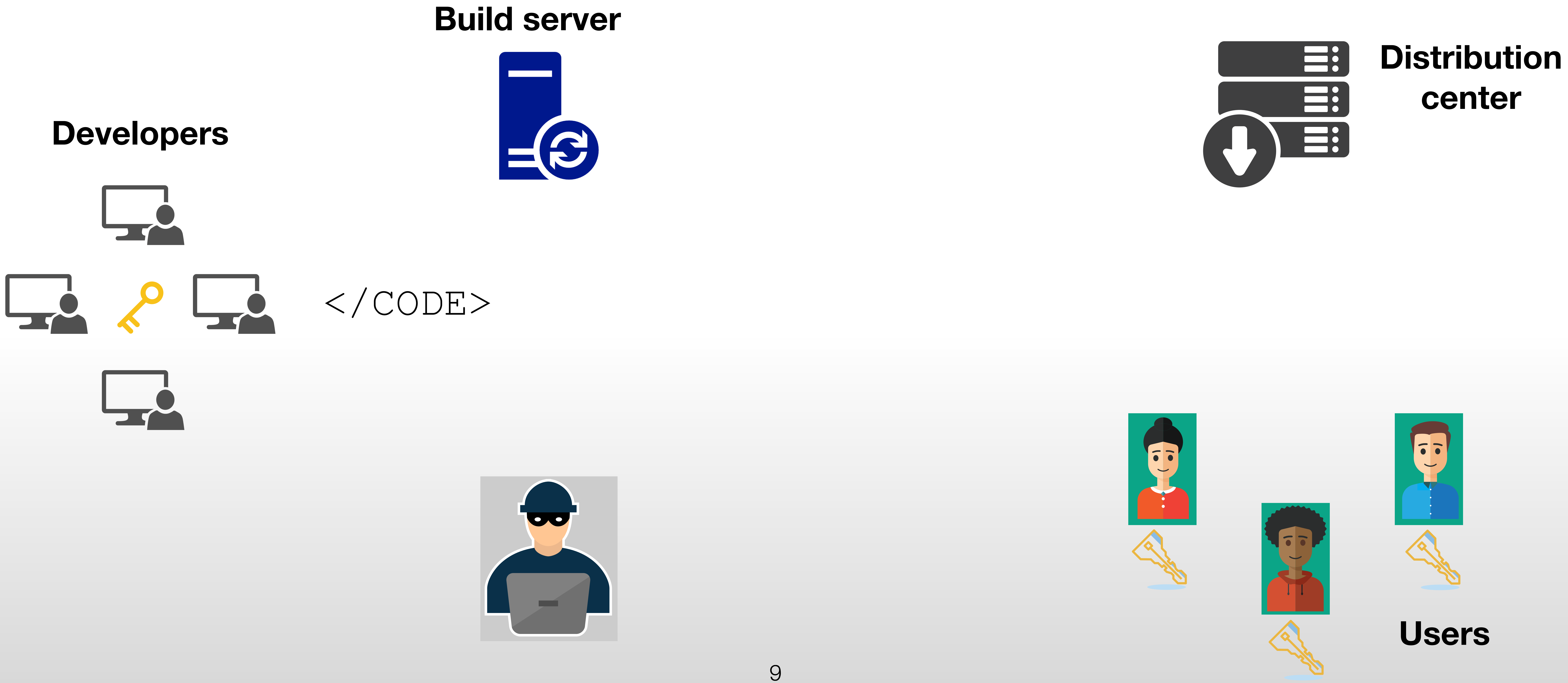
Software Release Pipeline

Development/Review – Building release binaries – Sign-off – **Release distribution**



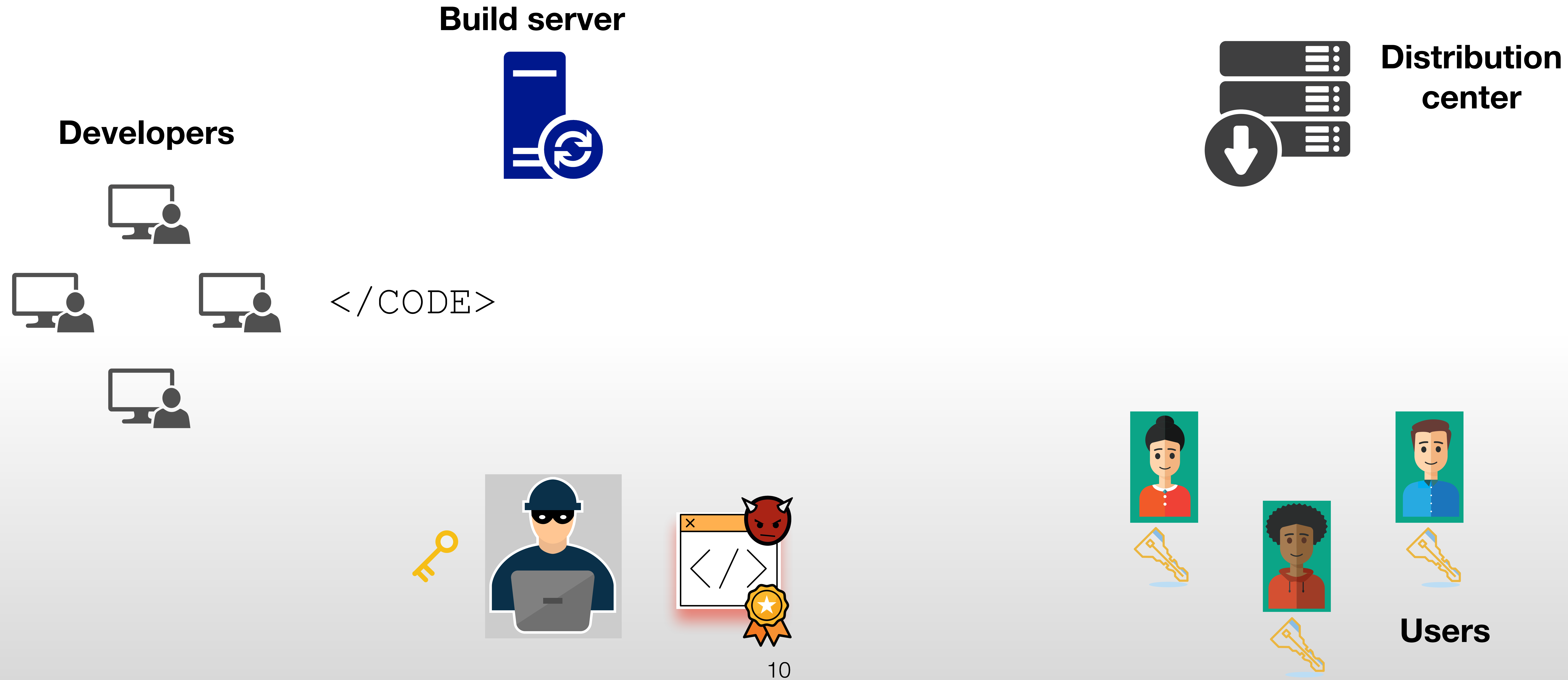
Challenges

1. Make software-update process resilient to partial key compromise



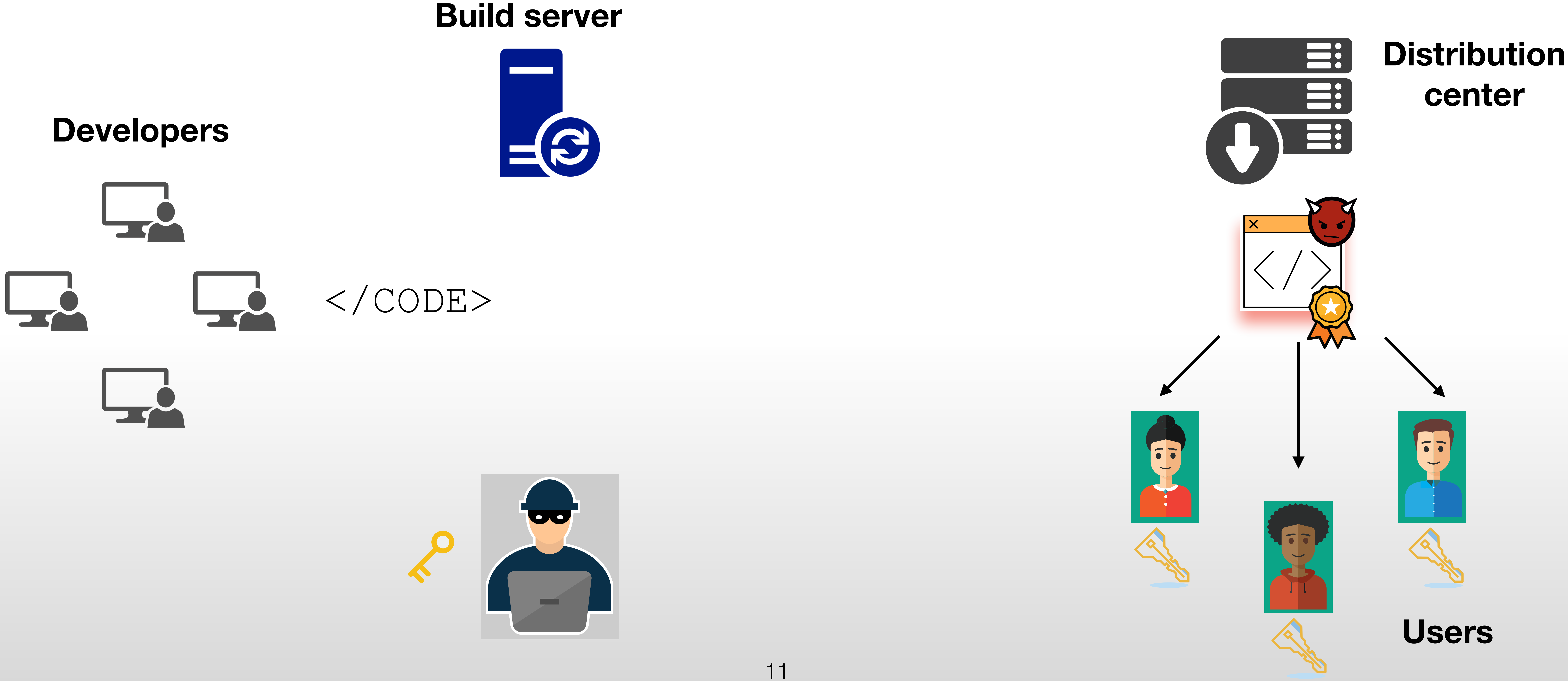
Challenges

1. Make software-update process resilient to partial key compromise



Challenges

1. Make software-update process resilient to partial key compromise

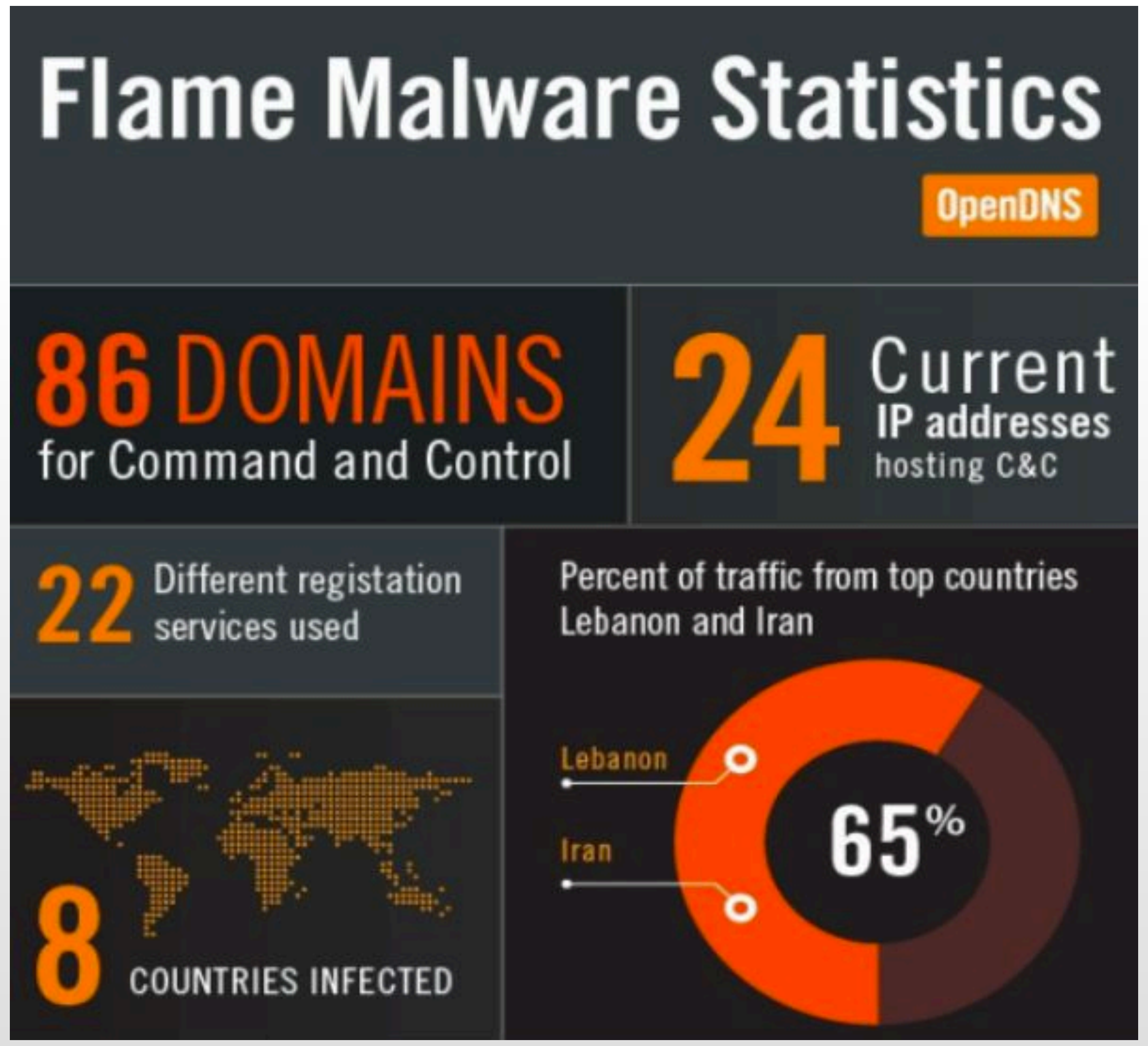


Challenges

1. Make software-update process resilient to partial key compromise

Talos report on
Petya/NotPetya attacks

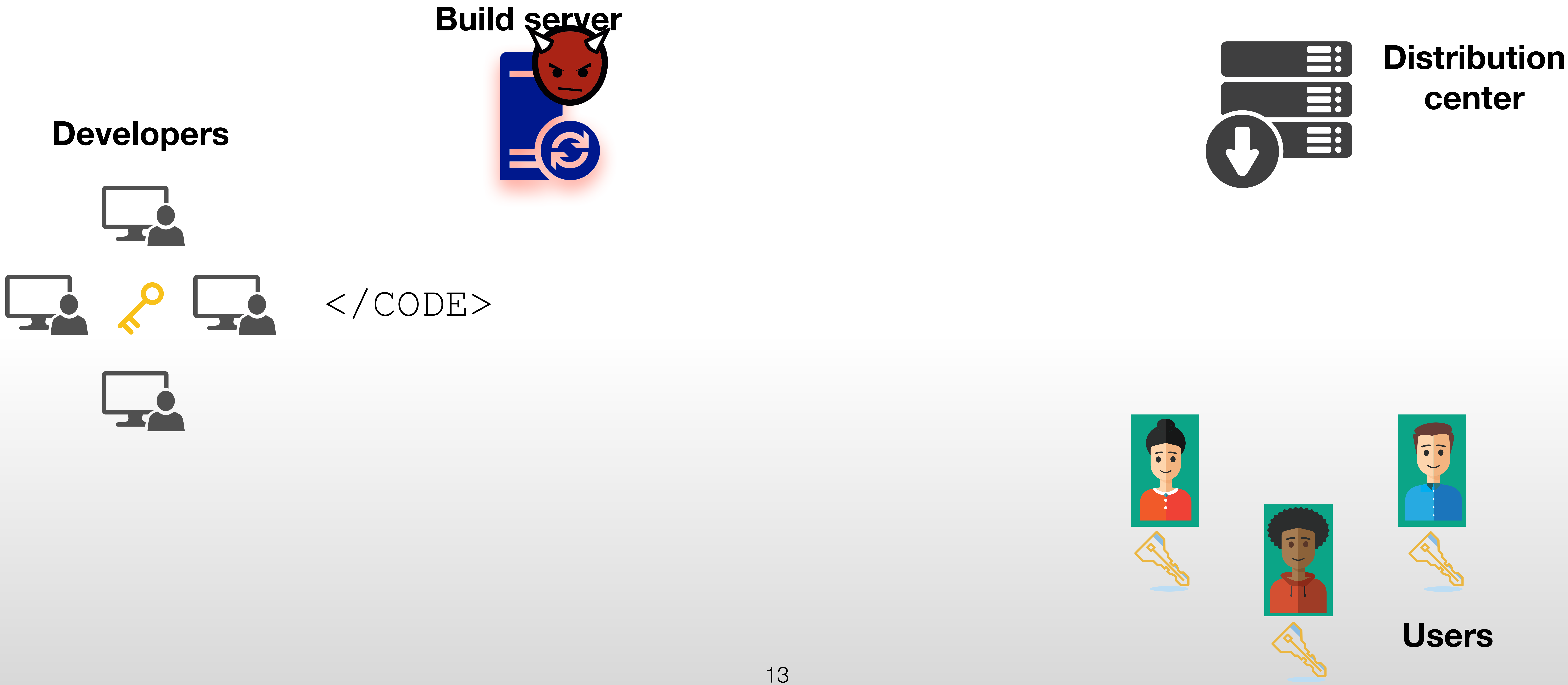
Kaspersky Securelist



Mashable

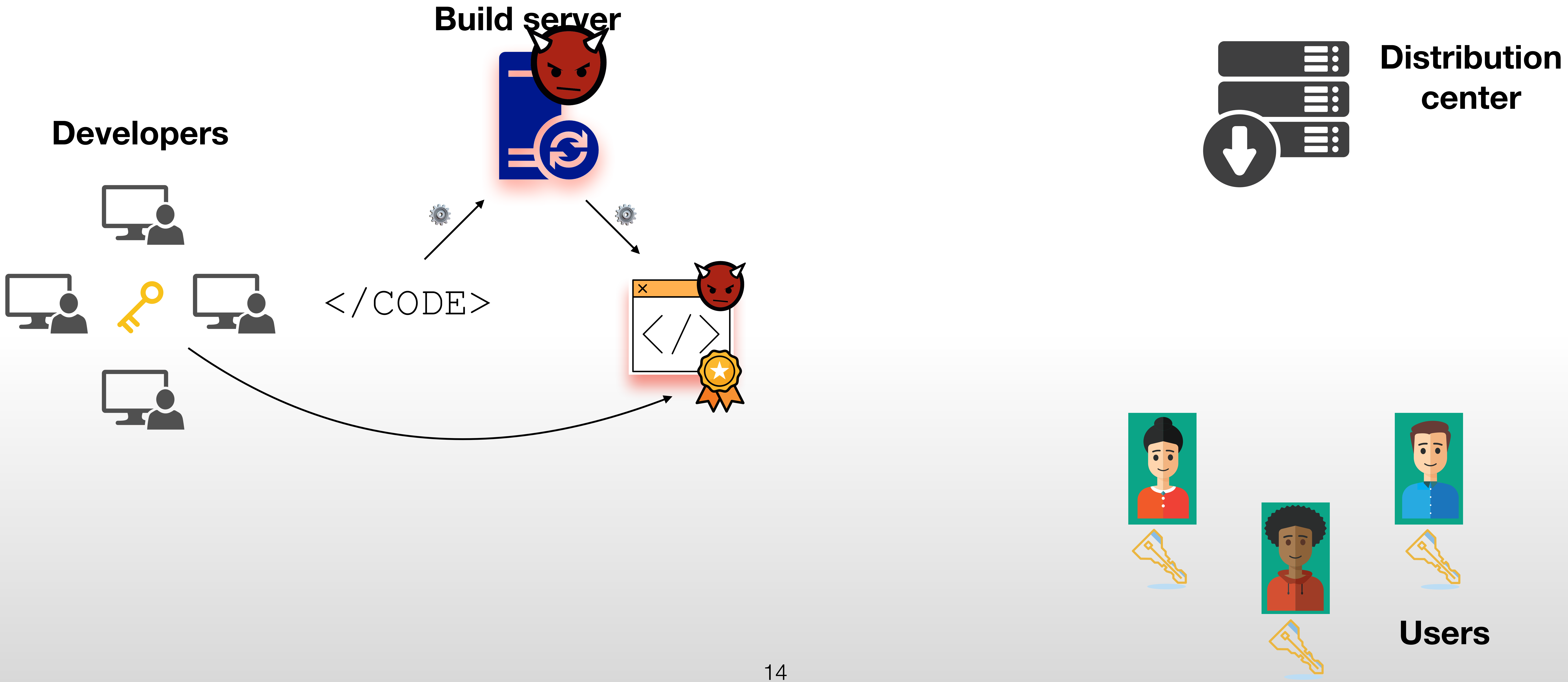
Challenges

2. Prevent malicious substitution of a release binary during building process



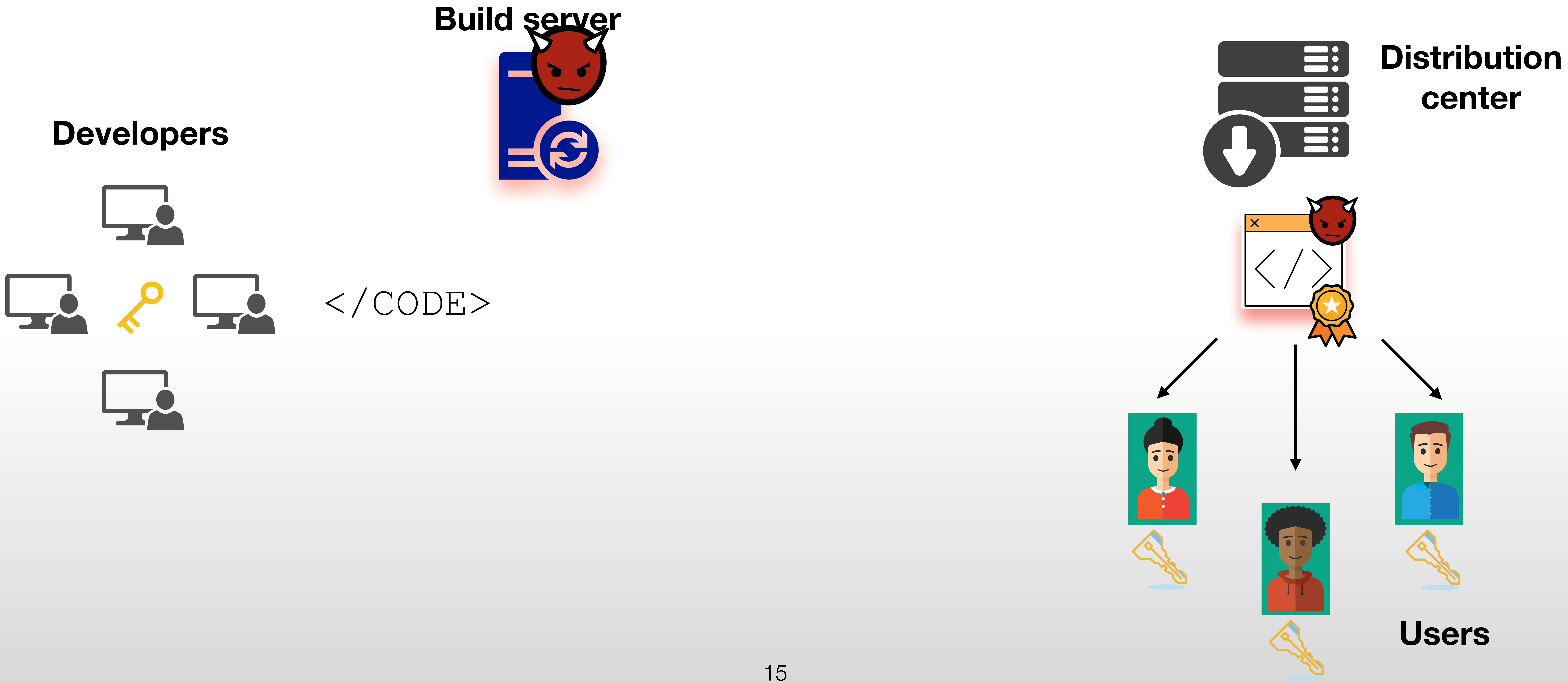
Challenges

2. Prevent malicious substitution of a release binary during a build process



Challenges

2. Prevent malicious substitution of a release binary during a build process



Challenges

2. Prevent malicious substitution of a release binary during a build process

reproducible-builds.org

Provide a verifiable path from source code to binary.

debian GNU/Linux

Over 90% of the source packages included in Debian 9 will build bit-for-bit identical binary packages

Challenges

How many of you have reproducibly built software binaries for personal use?

Challenges

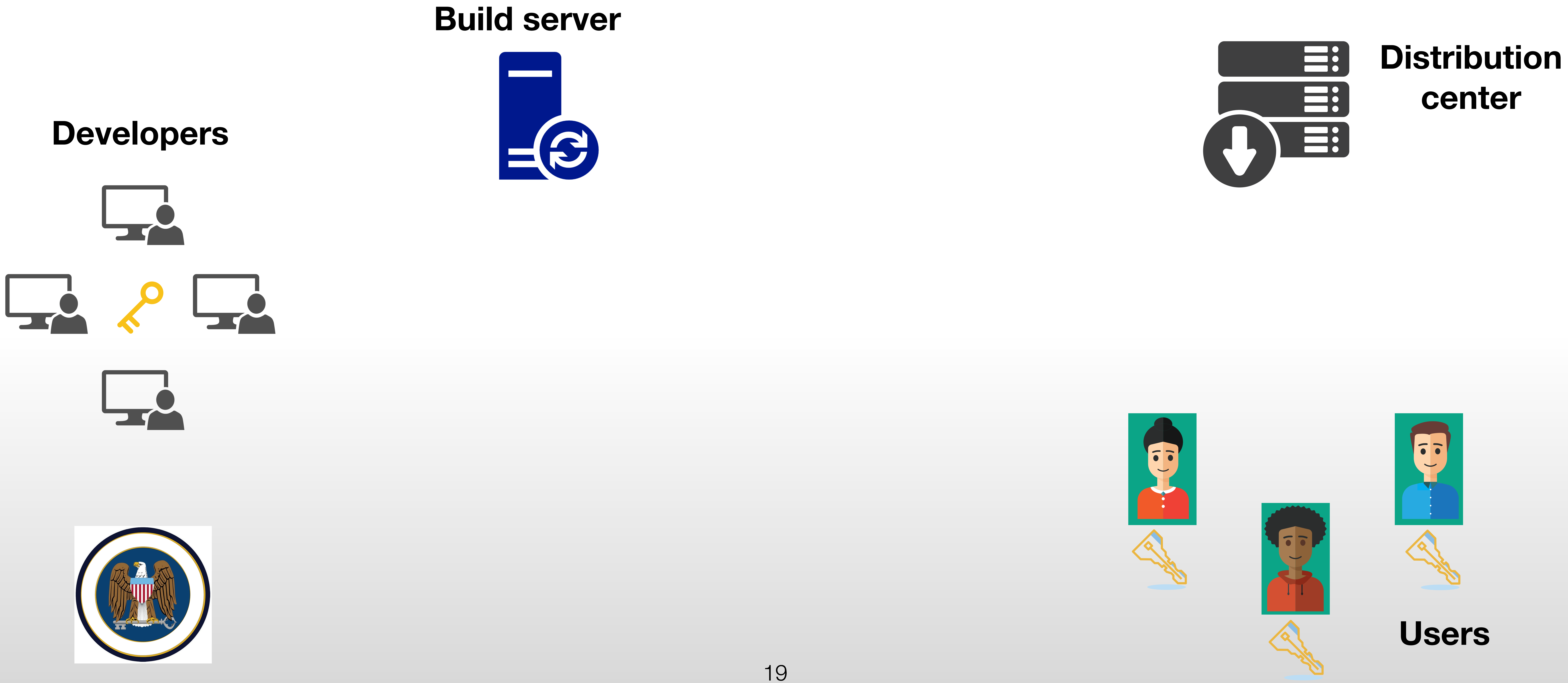
2. Prevent malicious substitution of a release binary during a build process

Building the Tor Browser bundle
<u>takes 32 hours</u> on a modern laptop

Closed-source software?

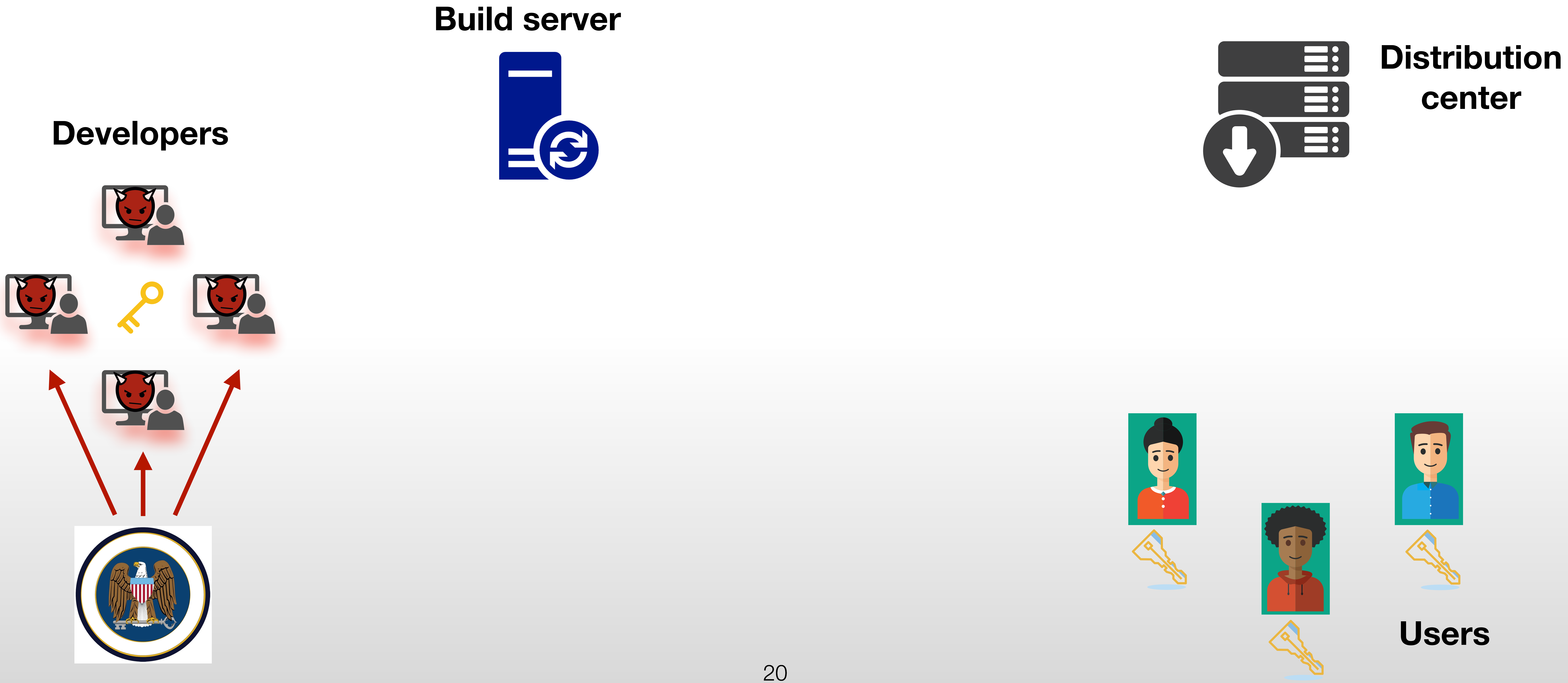
Challenges

3. Protect users from targeted attacks by coerced or bribed developers



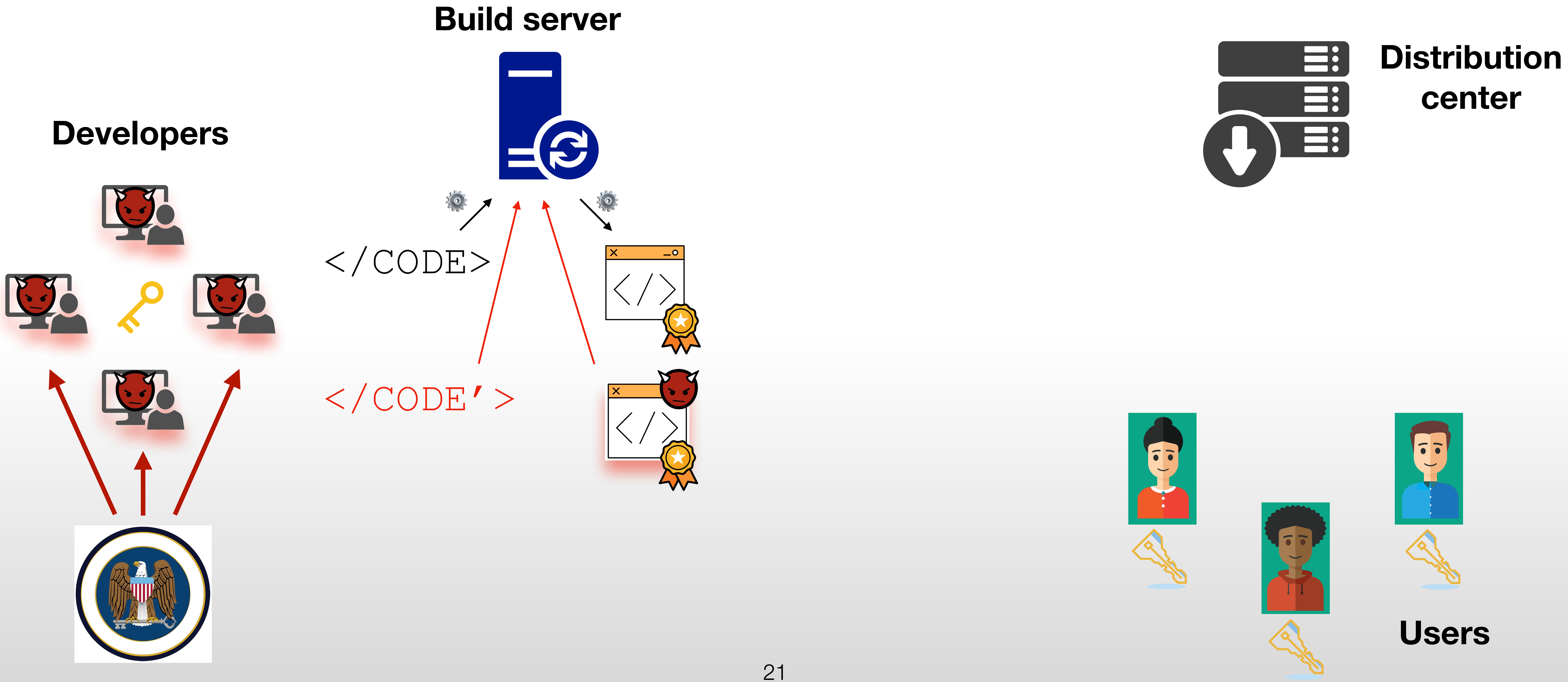
Challenges

3. Protect users from targeted attacks by coerced or bribed developers



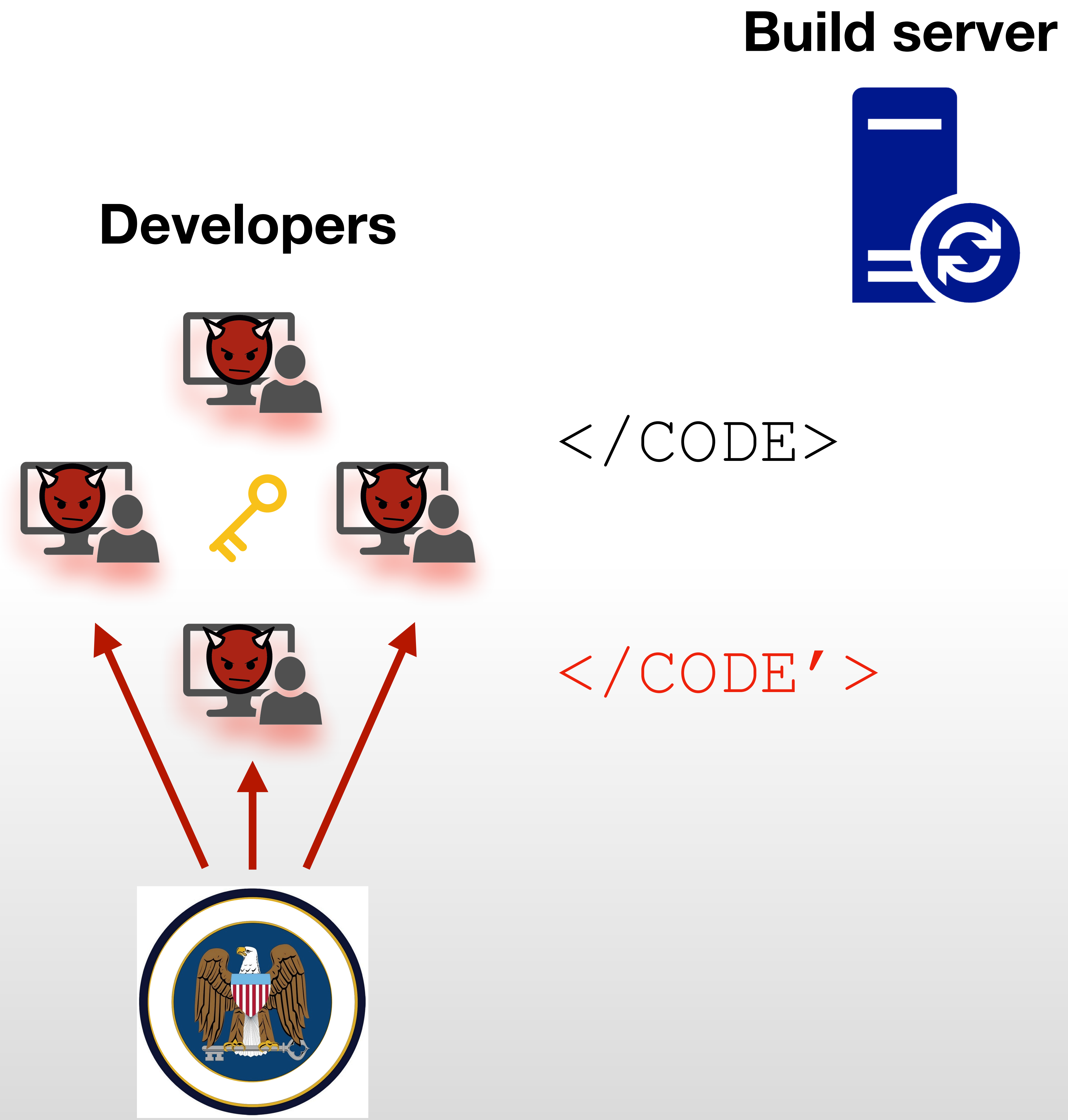
Challenges

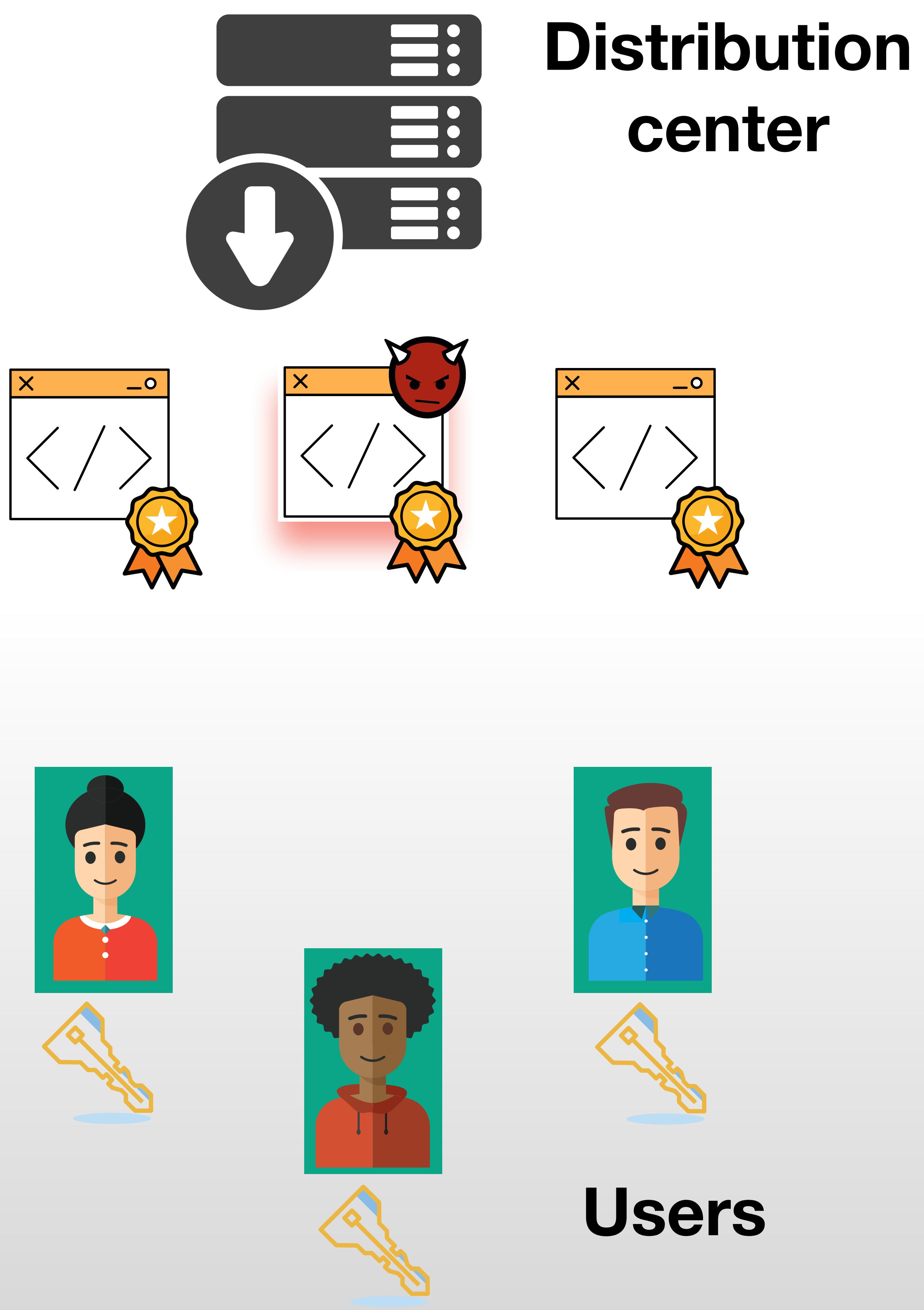
3. Protect users from targeted attacks by coerced or bribed developers



Challenges

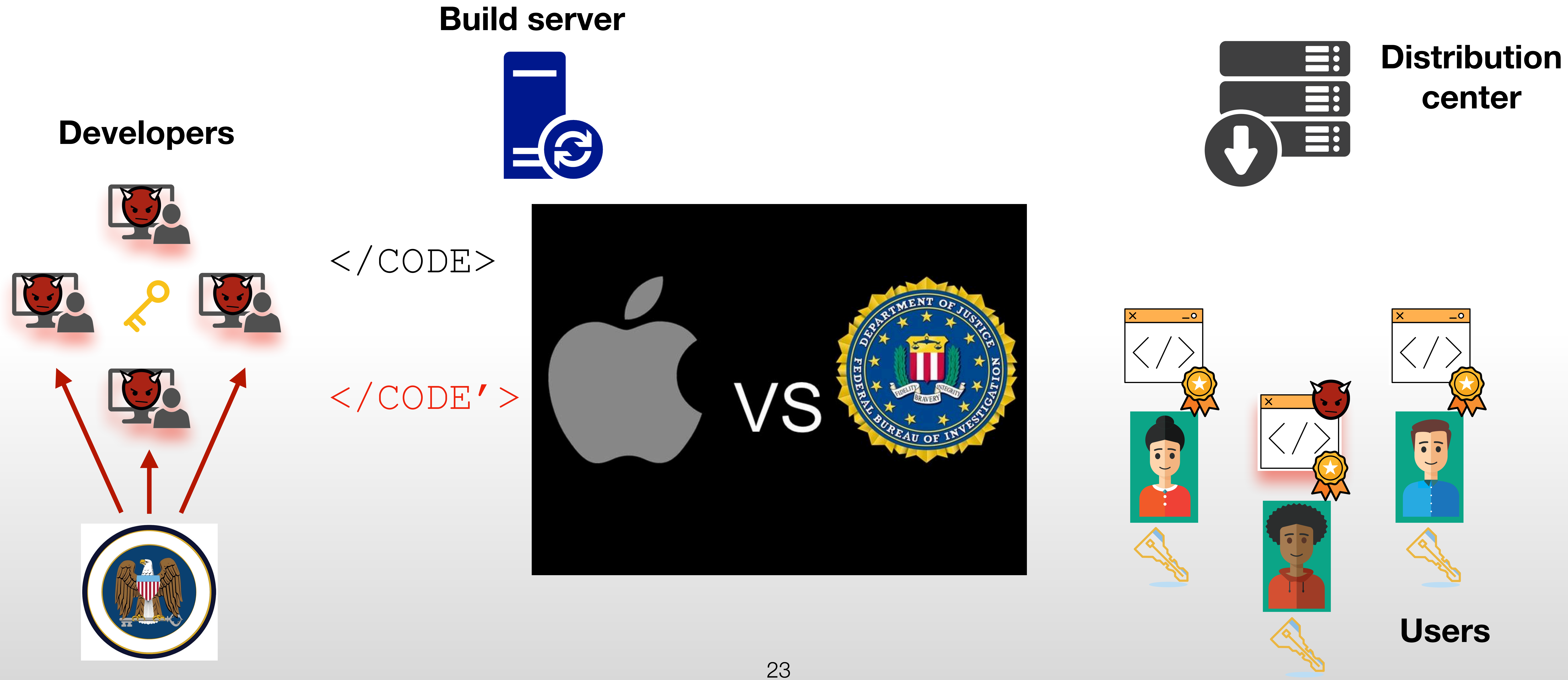
3. Protect users from targeted attacks by coerced or bribed developers





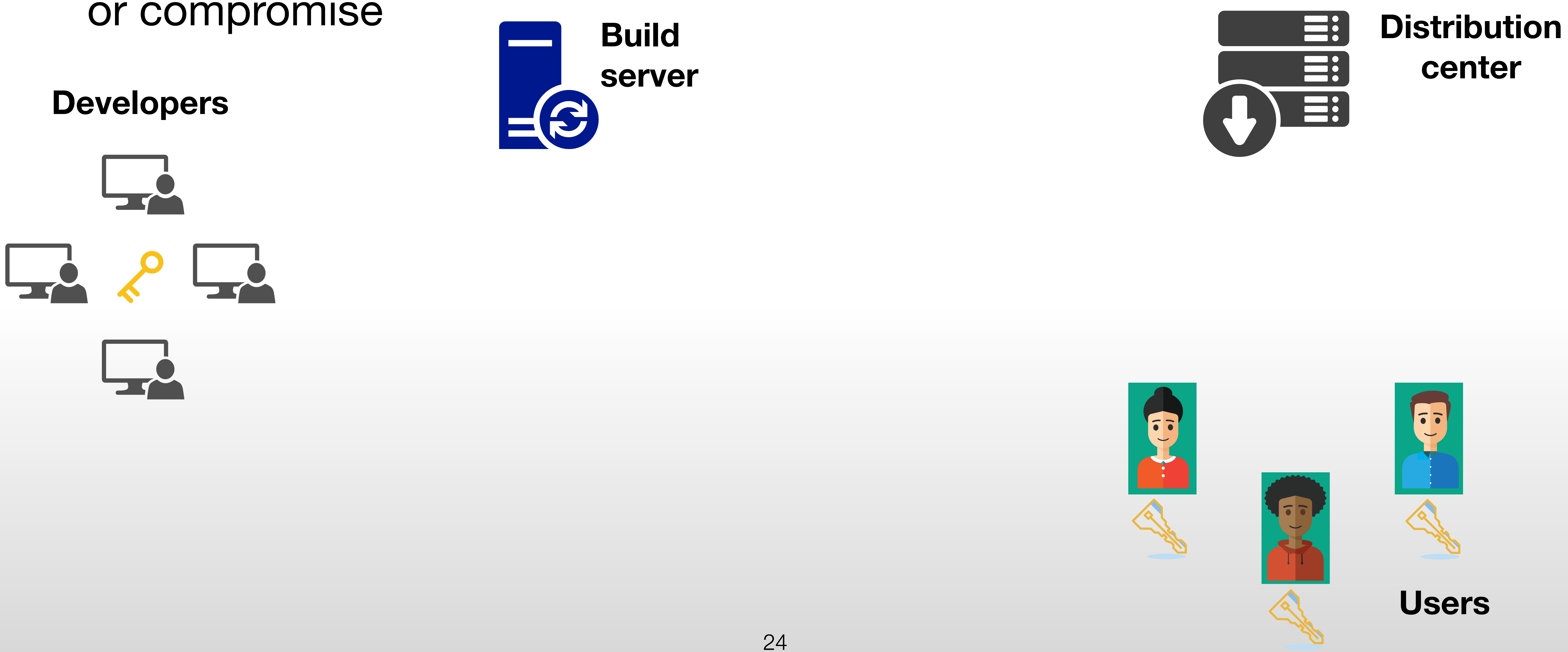
Challenges

3. Protect users from targeted attacks by coerced or bribed developers



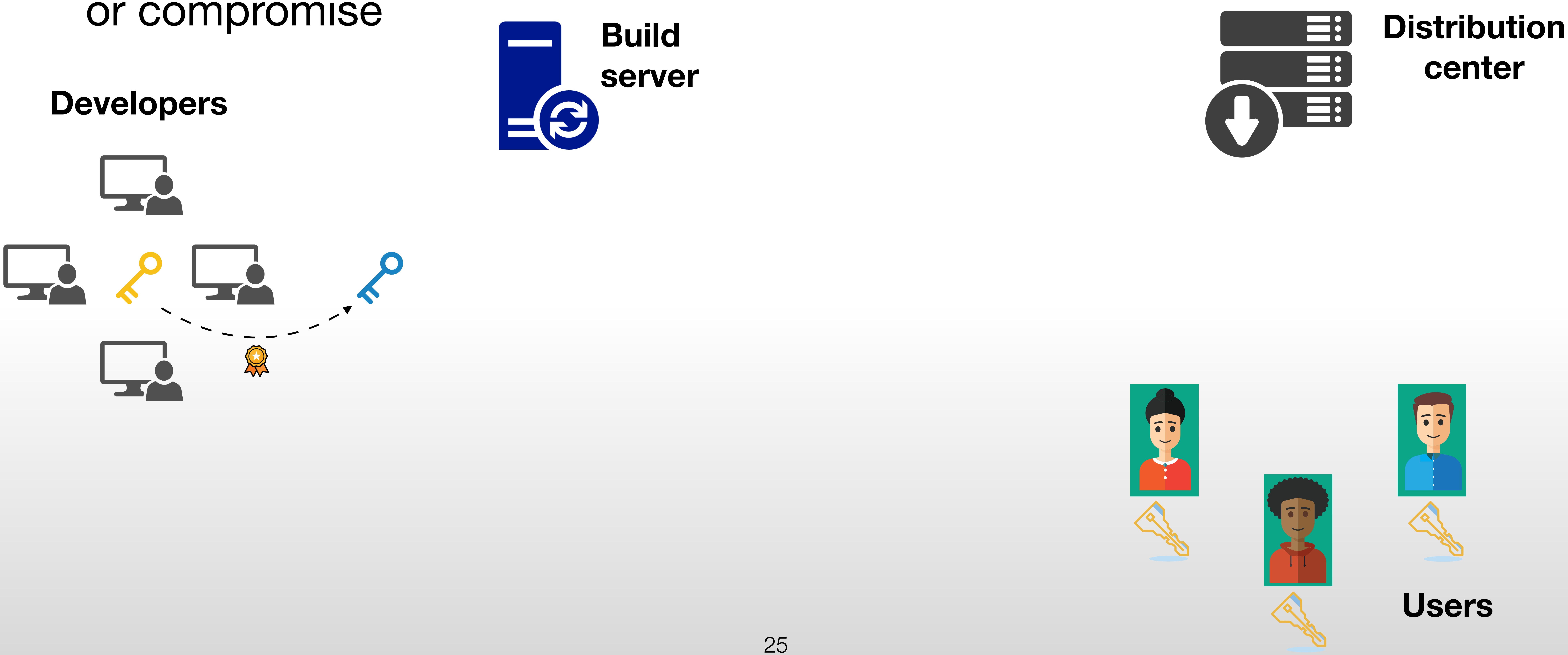
Challenges

4. Enable developers to securely rotate their signing keys in case of renewal or compromise



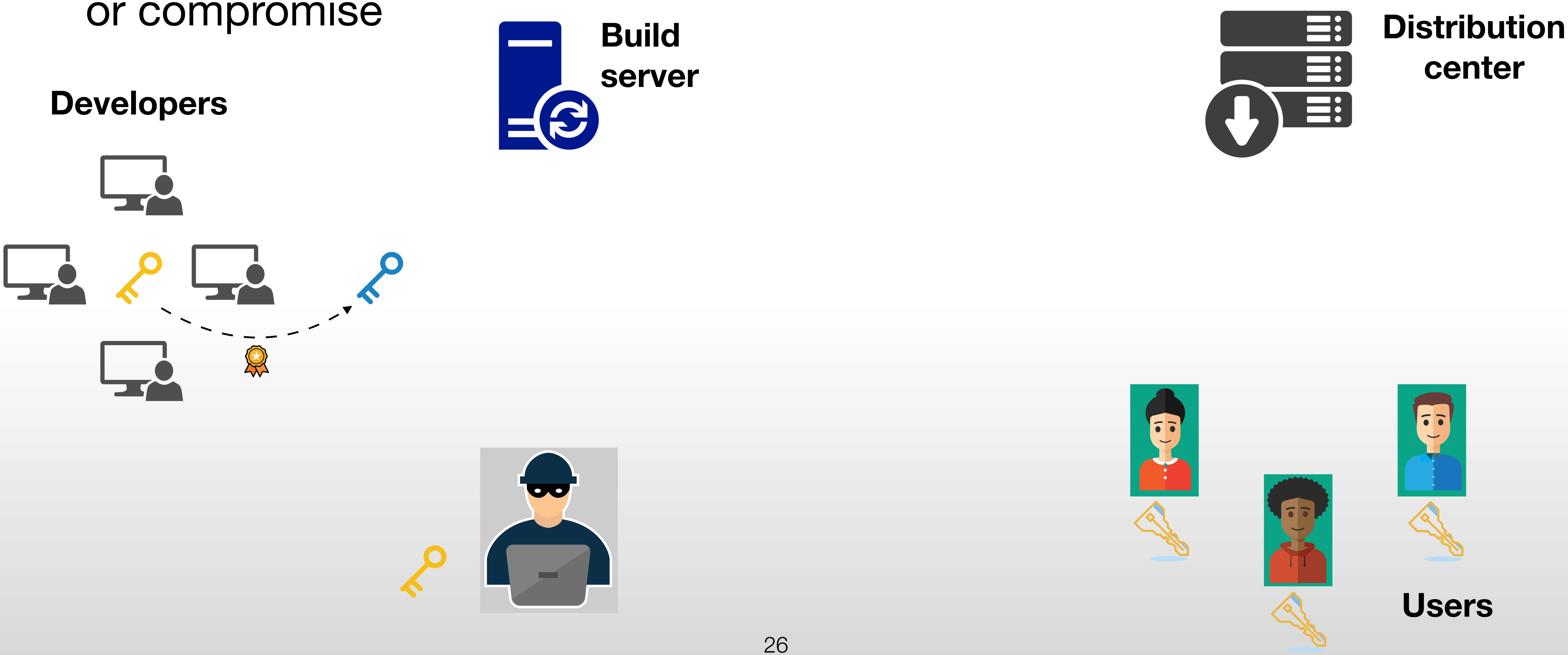
Challenges

4. Enable developers to securely rotate their signing keys in case of renewal or compromise



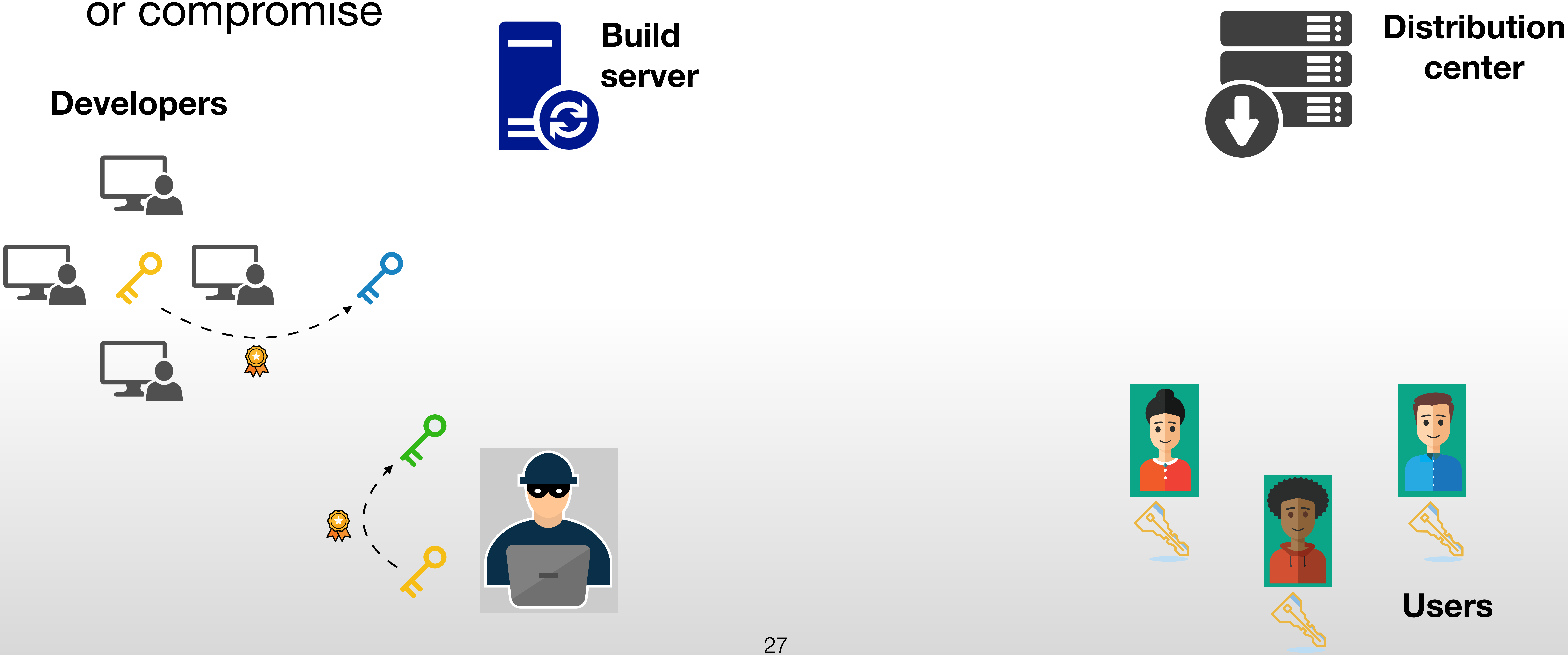
Challenges

4. Enable developers to securely rotate their signing keys in case of renewal or compromise



Challenges

4. Enable developers to securely rotate their signing keys in case of renewal or compromise



Design of CHAINIAC

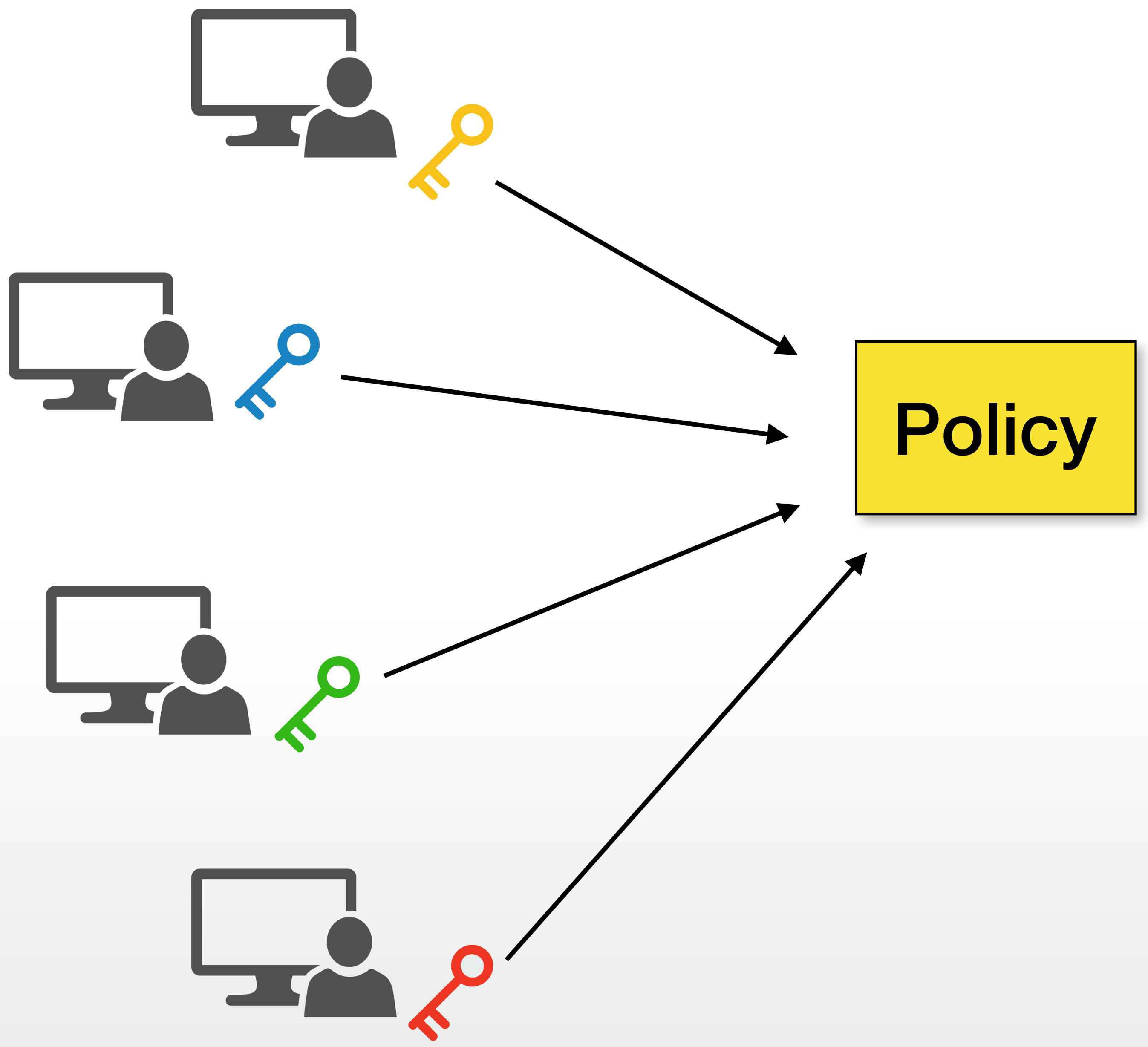
Roadmap to CHAINIAC

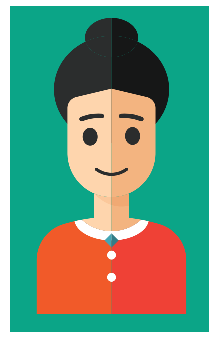
- Decentralized Release Approval
- Verified Builds
- Anti-equivocation
- Key Evolution

Decentralized Release-Approval

1. Make software-update process resilient to partial key compromise

Developers



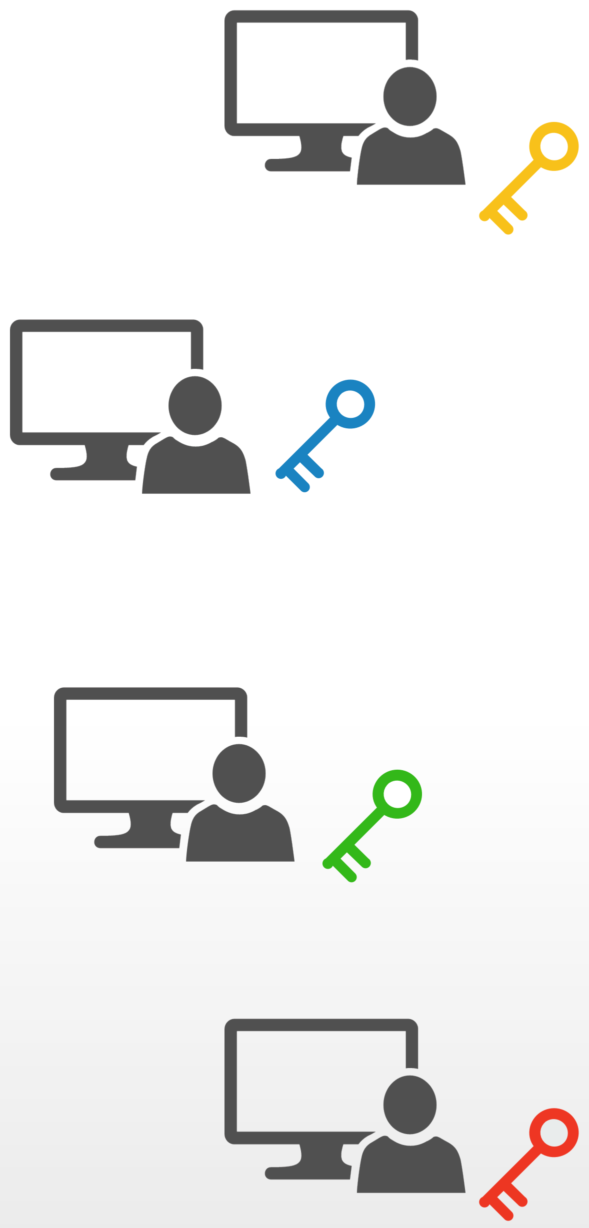


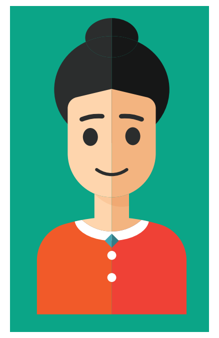
User

Decentralized Release-Approval

1. Make software-update process resilient to partial key compromise

Developers



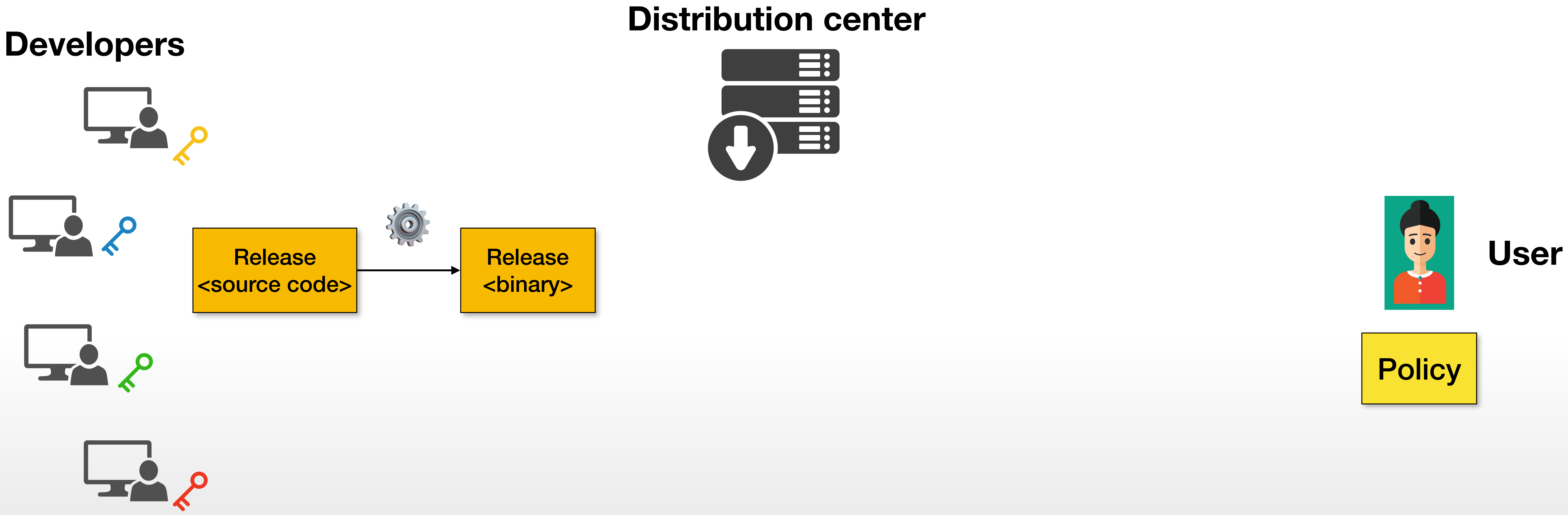


User

Policy

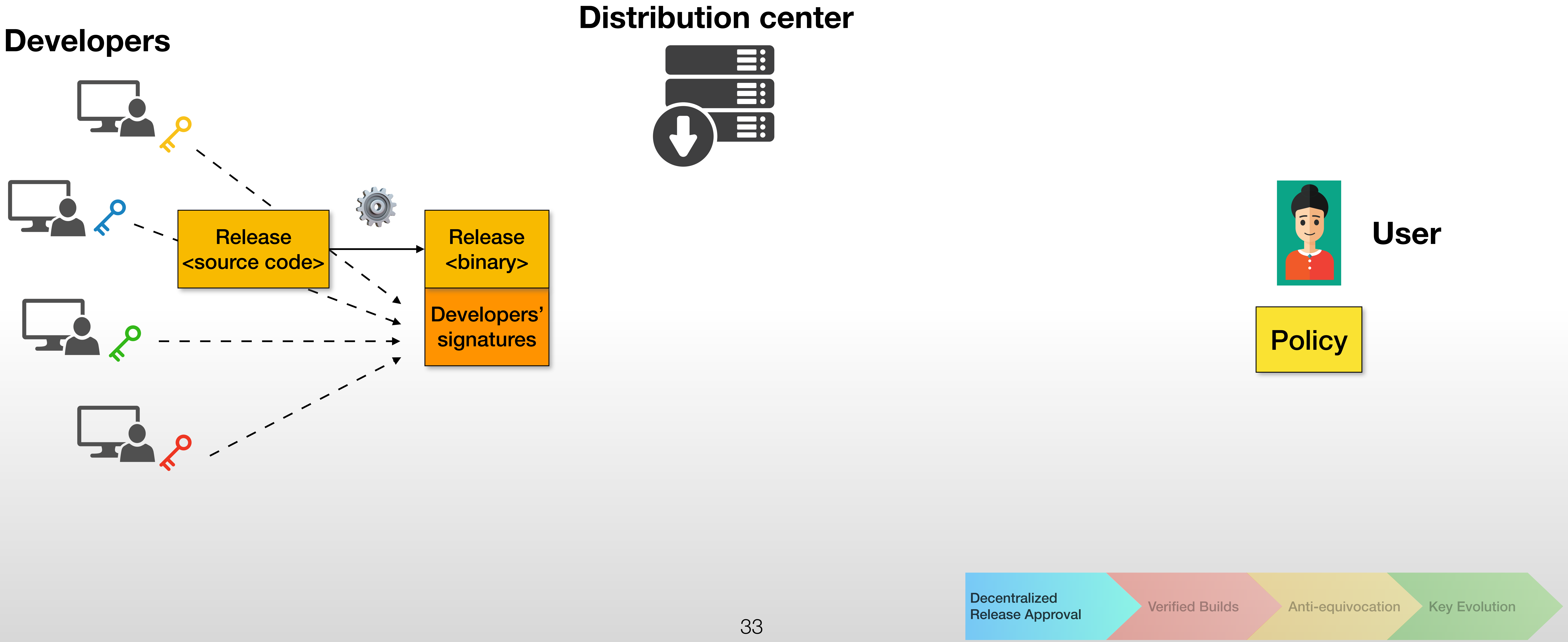
Decentralized Release-Approval

1. Make software-update process resilient to partial key compromise



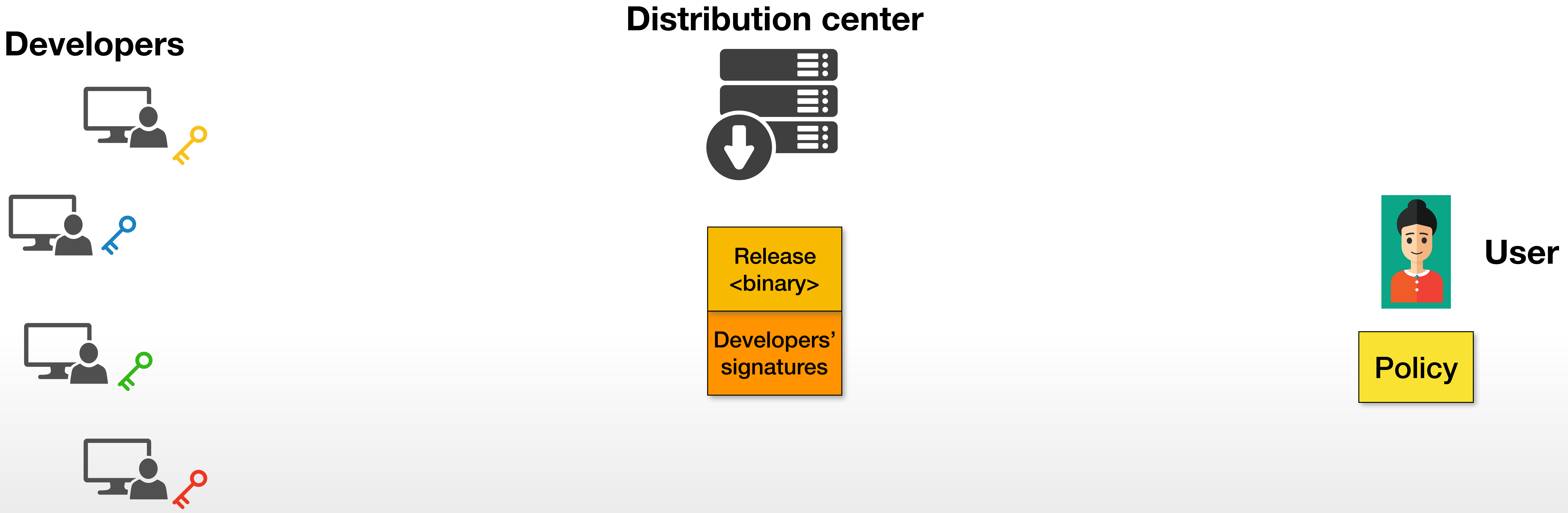
Decentralized Release-Approval

1. Make software-update process resilient to partial key compromise



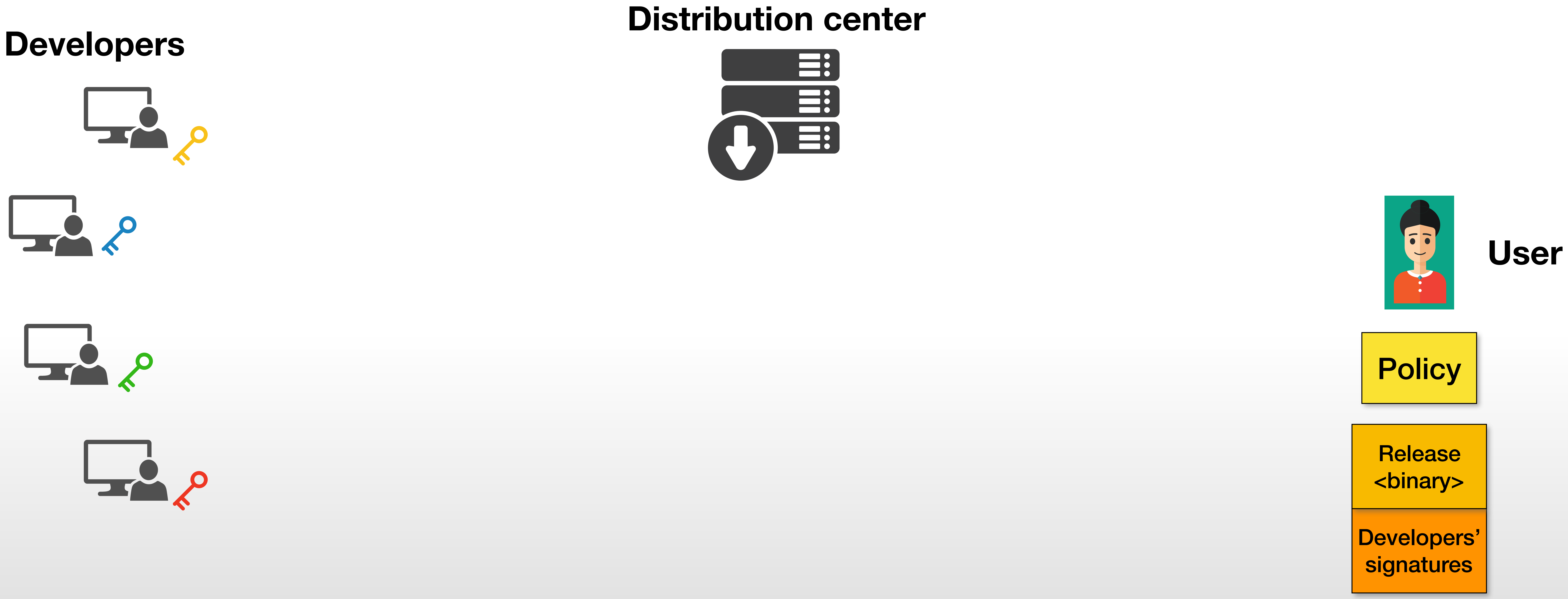
Decentralized Release-Approval

1. Make software-update process resilient to partial key compromise



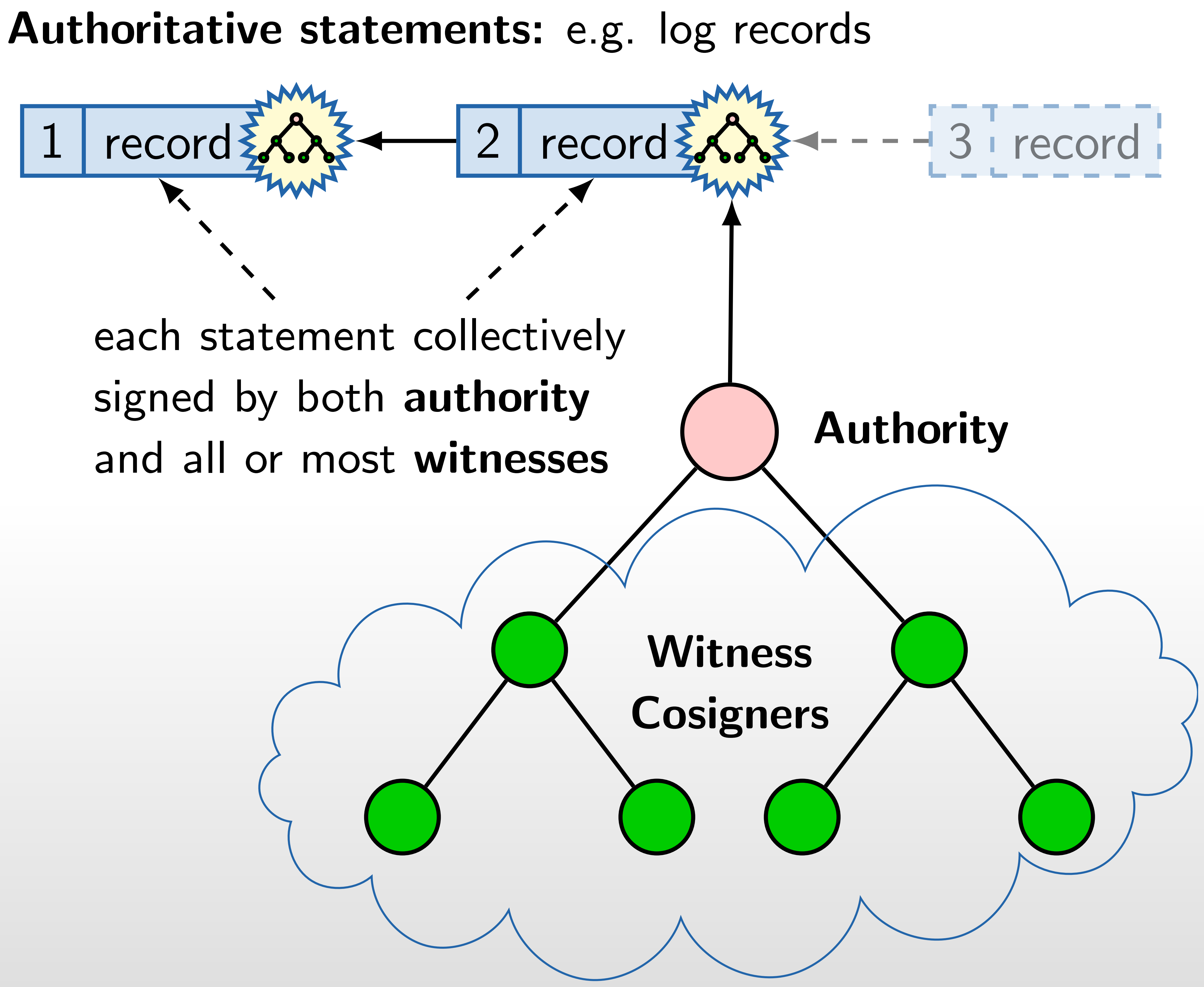
Decentralized Release-Approval

1. Make software-update process resilient to partial key compromise



Background

- Collective Authority (Cothority), Collective Signing (CoSi), and BFT-CoSi

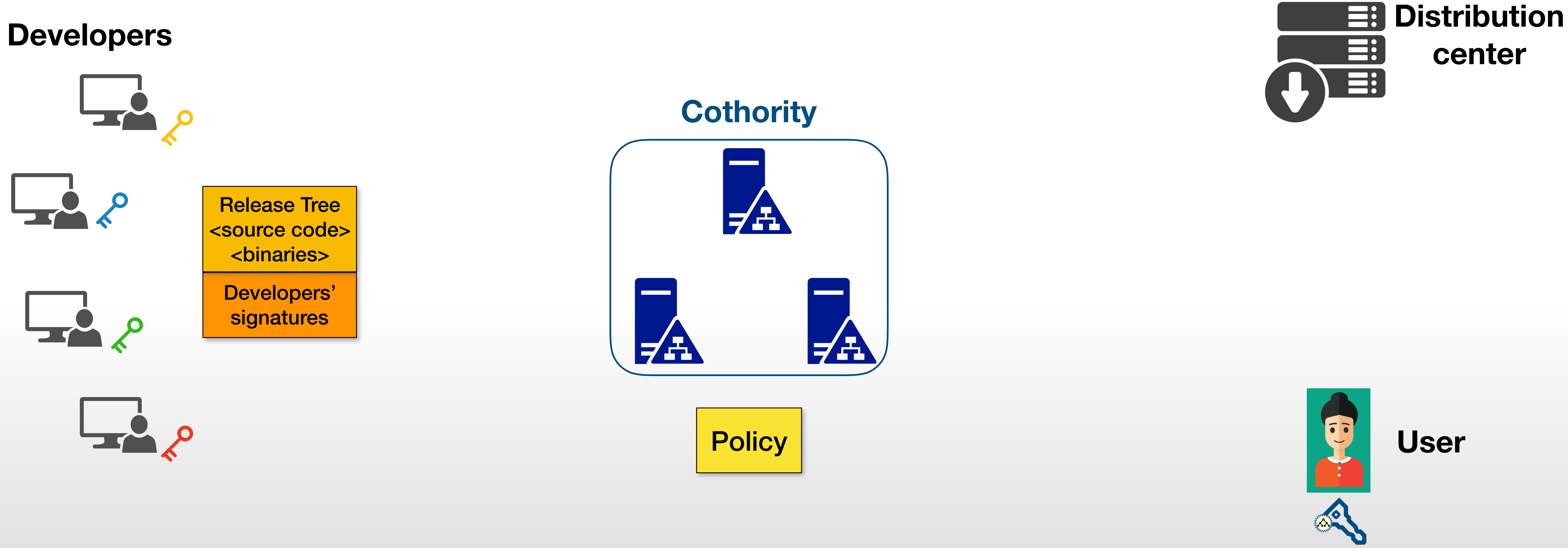


References

- Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning. In *37th IEEE Symposium on Security and Privacy*, May 2016.
- Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing. In *Proceedings of the 25th USENIX Conference on Security Symposium*, 2016.

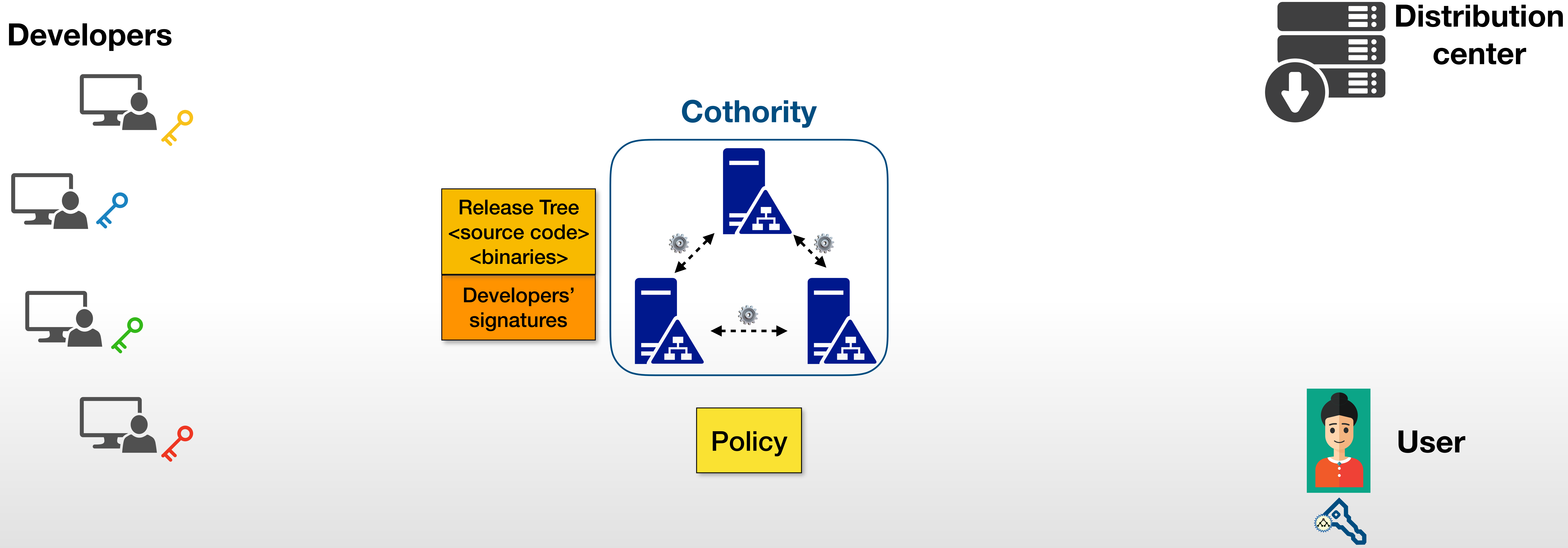
Verified Builds

2. Prevent malicious substitution of a release binary during building process



Verified Builds

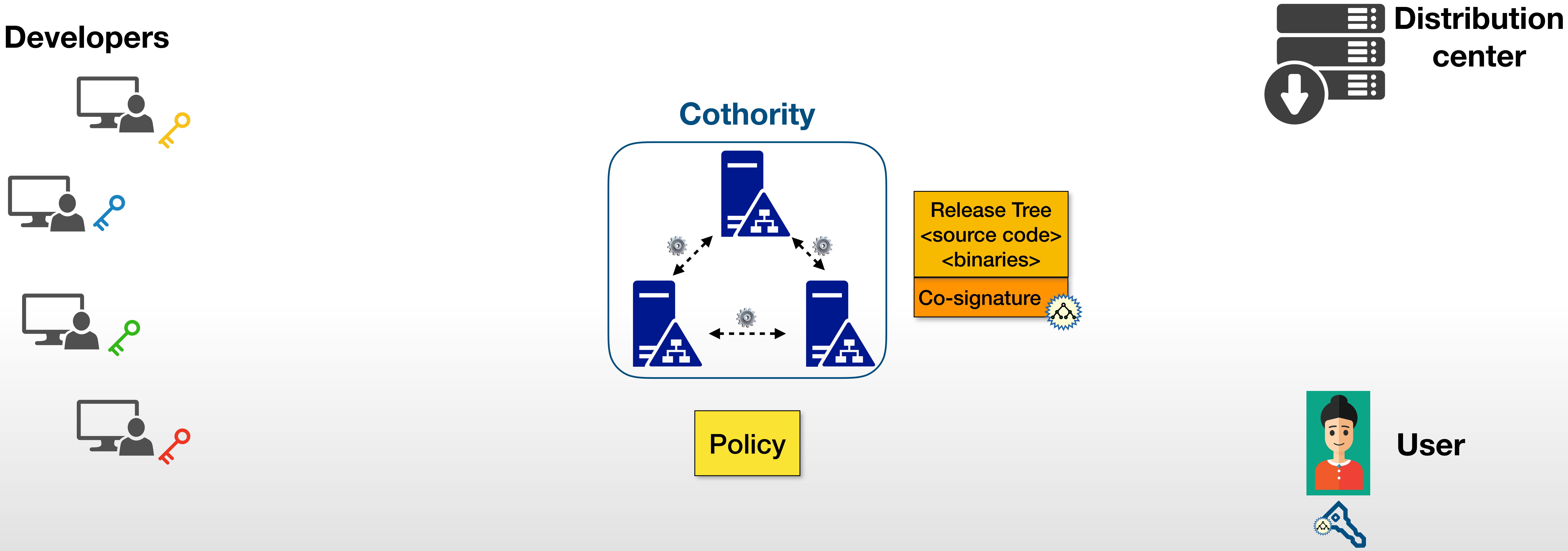
2. Prevent malicious substitution of a release binary during building process



Decentralized Release Approval | Verified Builds | Anti-equivocation | Key Evolution

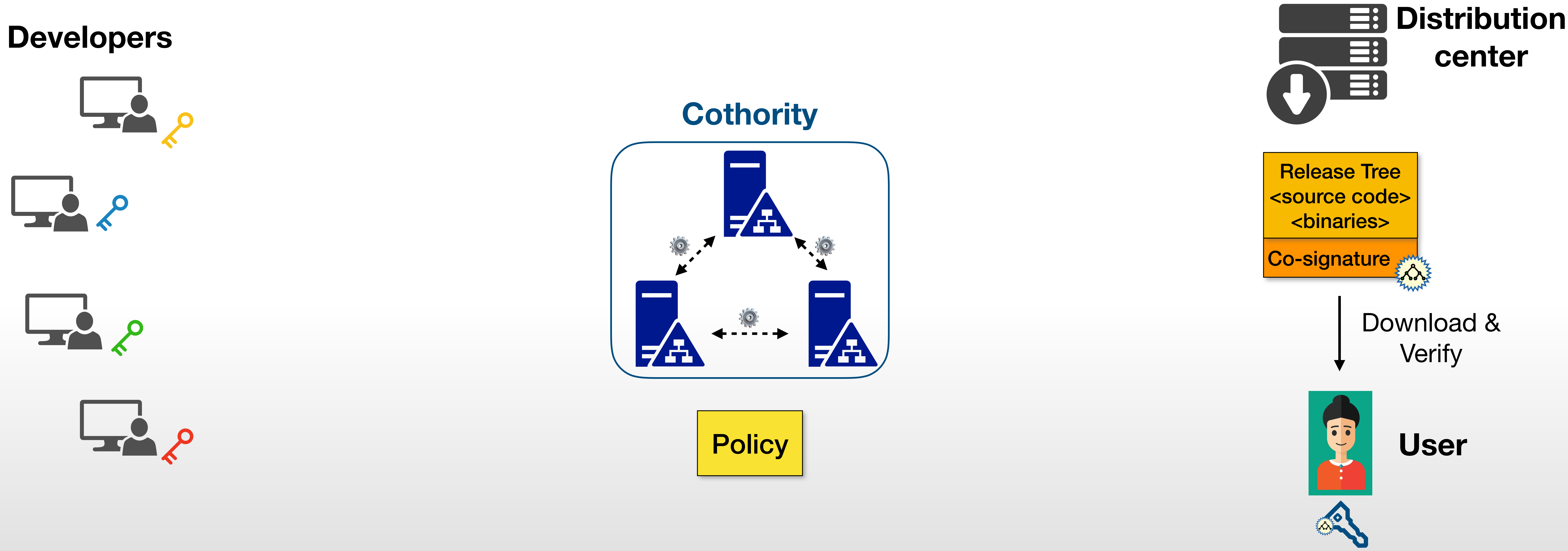
Verified Builds

2. Prevent malicious substitution of a release binary during building process



Verified Builds

2. Prevent malicious substitution of a release binary during building process



Decentralized Release Approval | Verified Builds | Anti-equivocation | Key Evolution

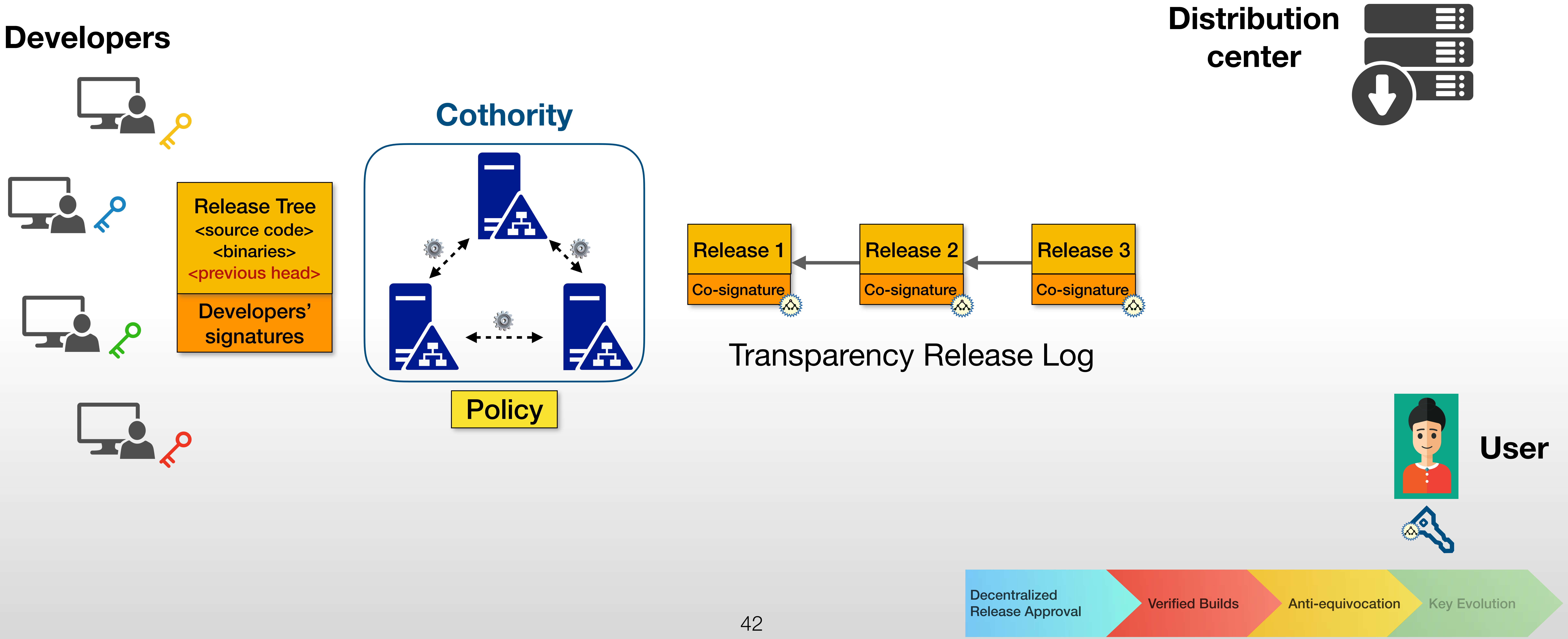
Verified Builds

Release Policy File

- List of individual developer public keys
- Signing threshold
- Cothority public key
- Supported platforms for verified builds
- …

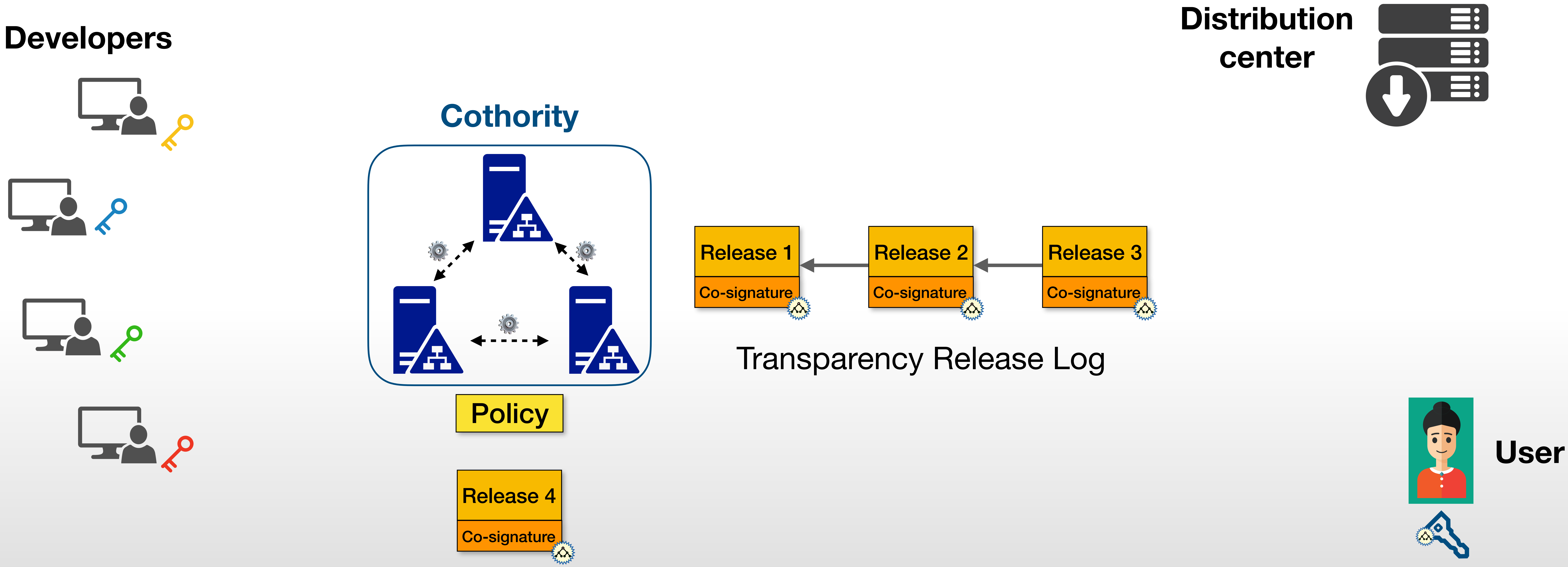
Anti-equivocation Measures

3. Protect users from targeted attacks by coerced or bribed developers



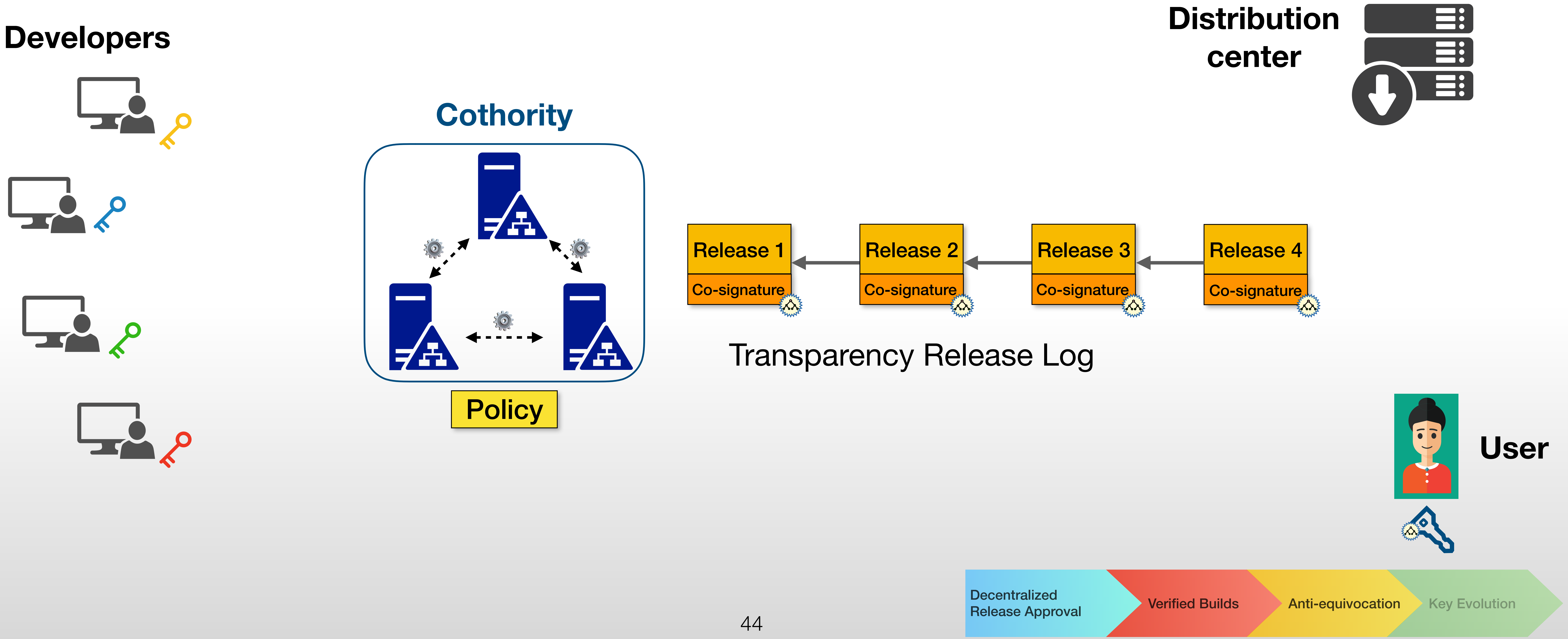
Anti-equivocation Measures

3. Protect users from targeted attacks by coerced or bribed developers



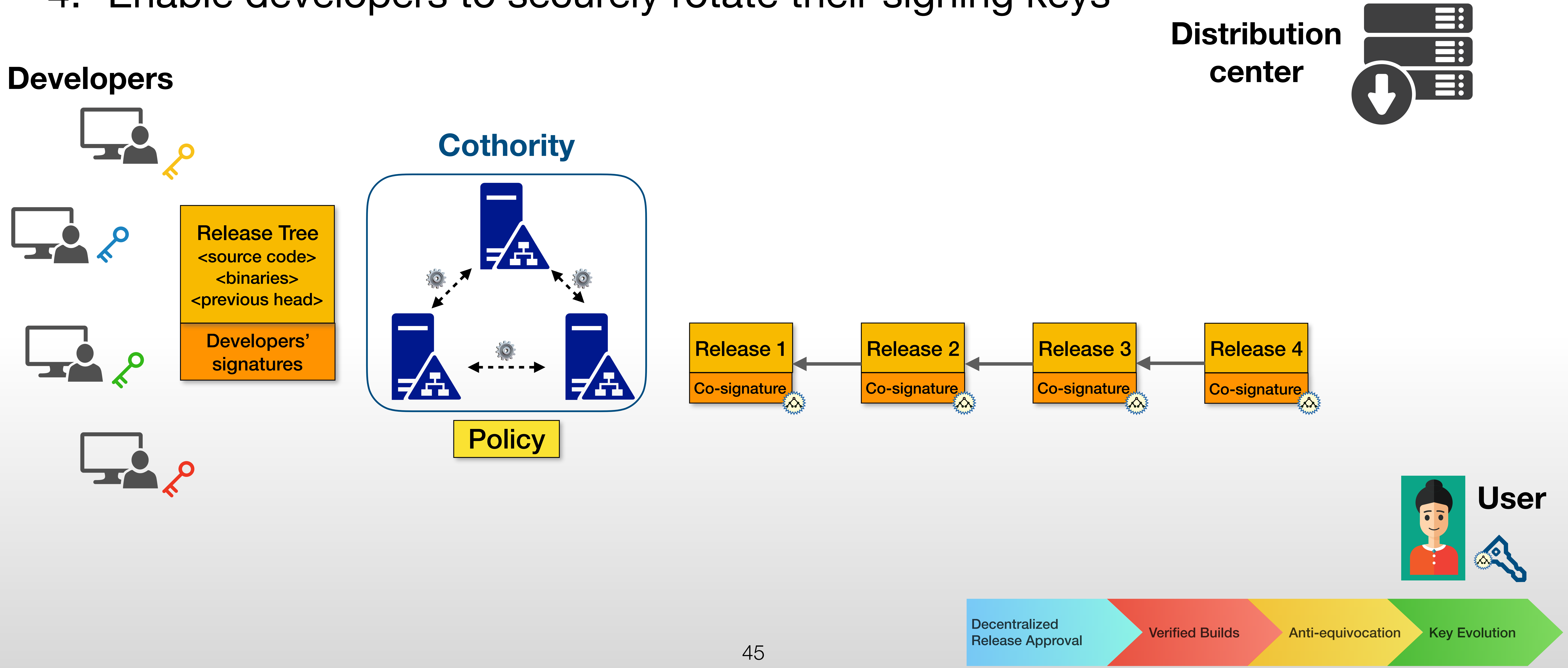
Anti-equivocation Measures

3. Protect users from targeted attacks by coerced or bribed developers



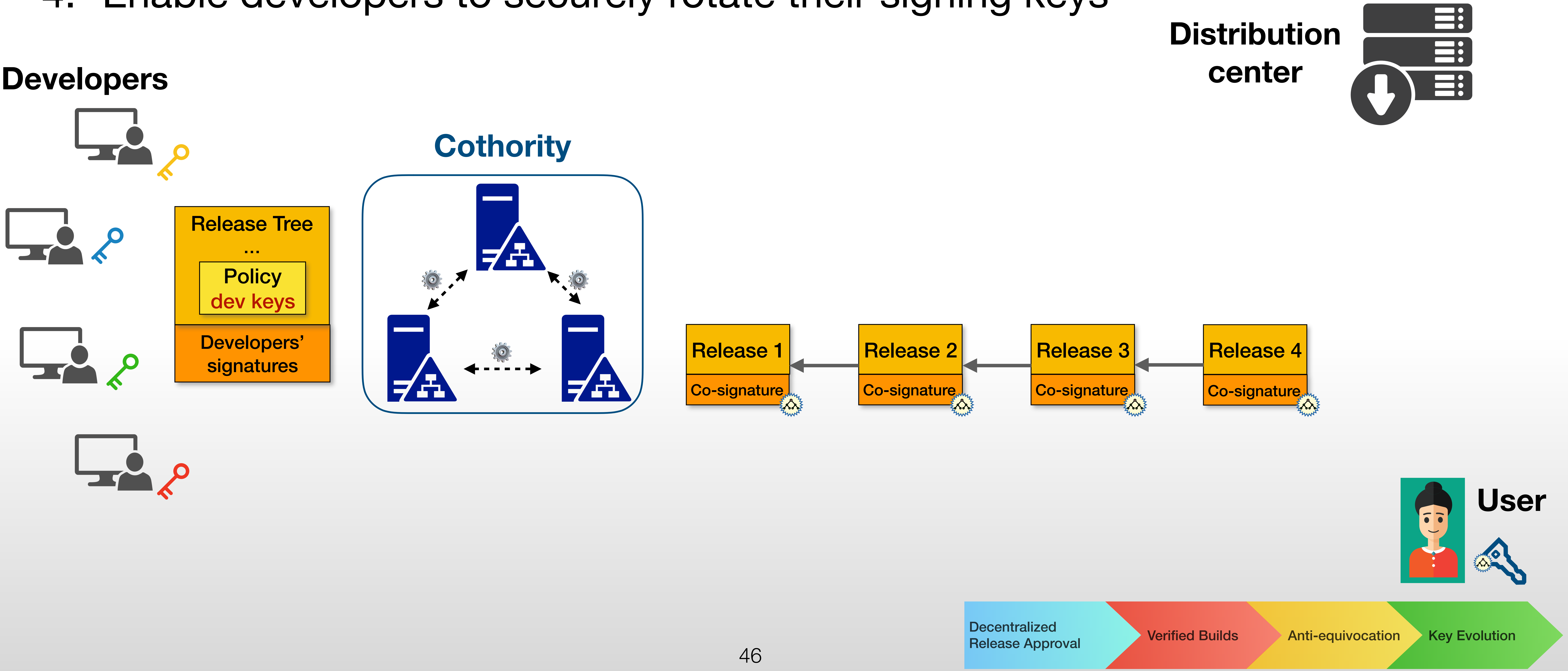
Evolution of Developer Keys

4. Enable developers to securely rotate their signing keys



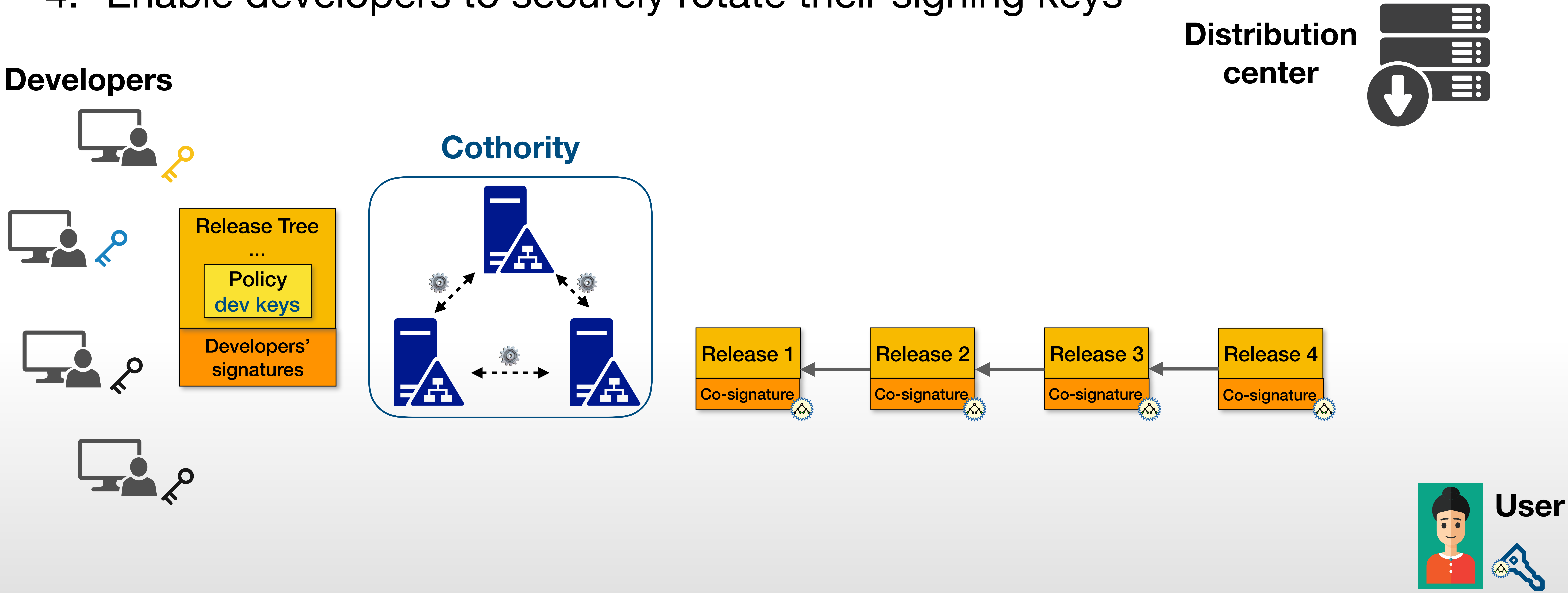
Evolution of Developer Keys

4. Enable developers to securely rotate their signing keys



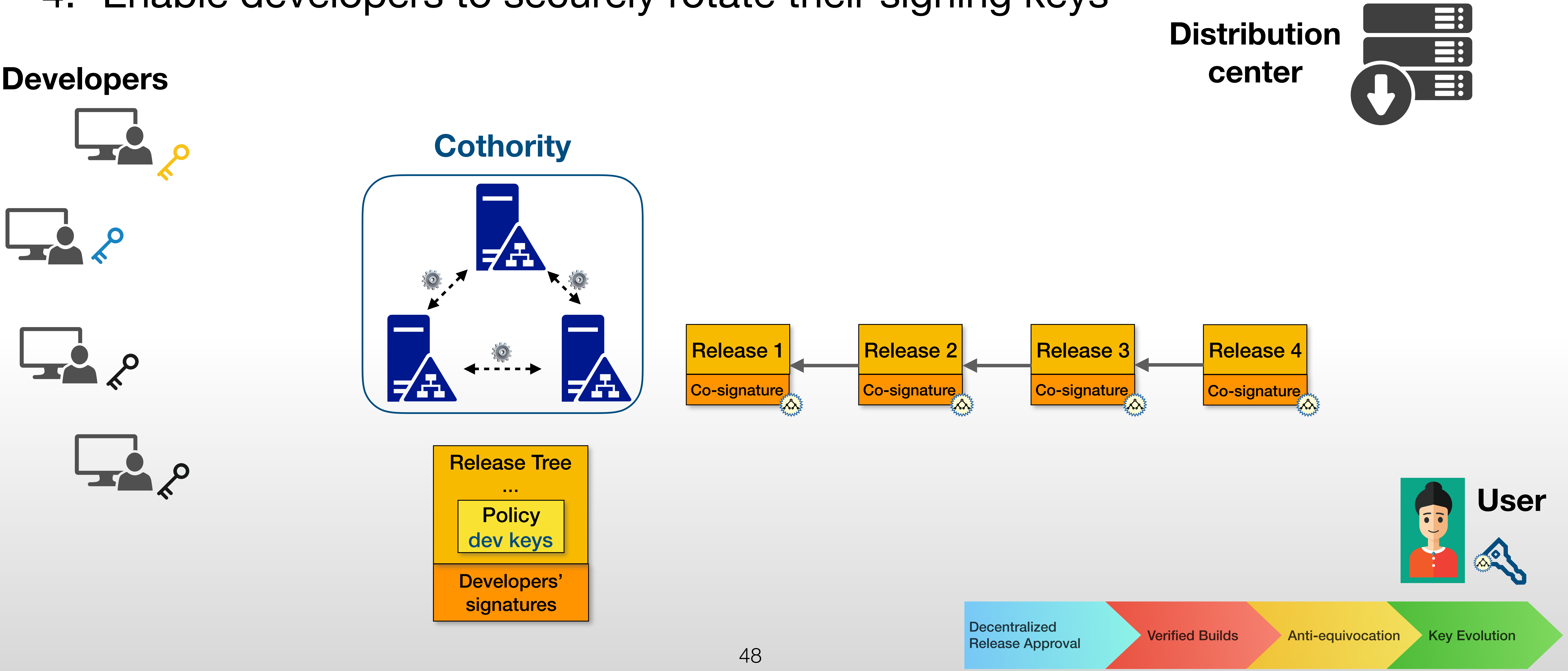
Evolution of Developer Keys

4. Enable developers to securely rotate their signing keys



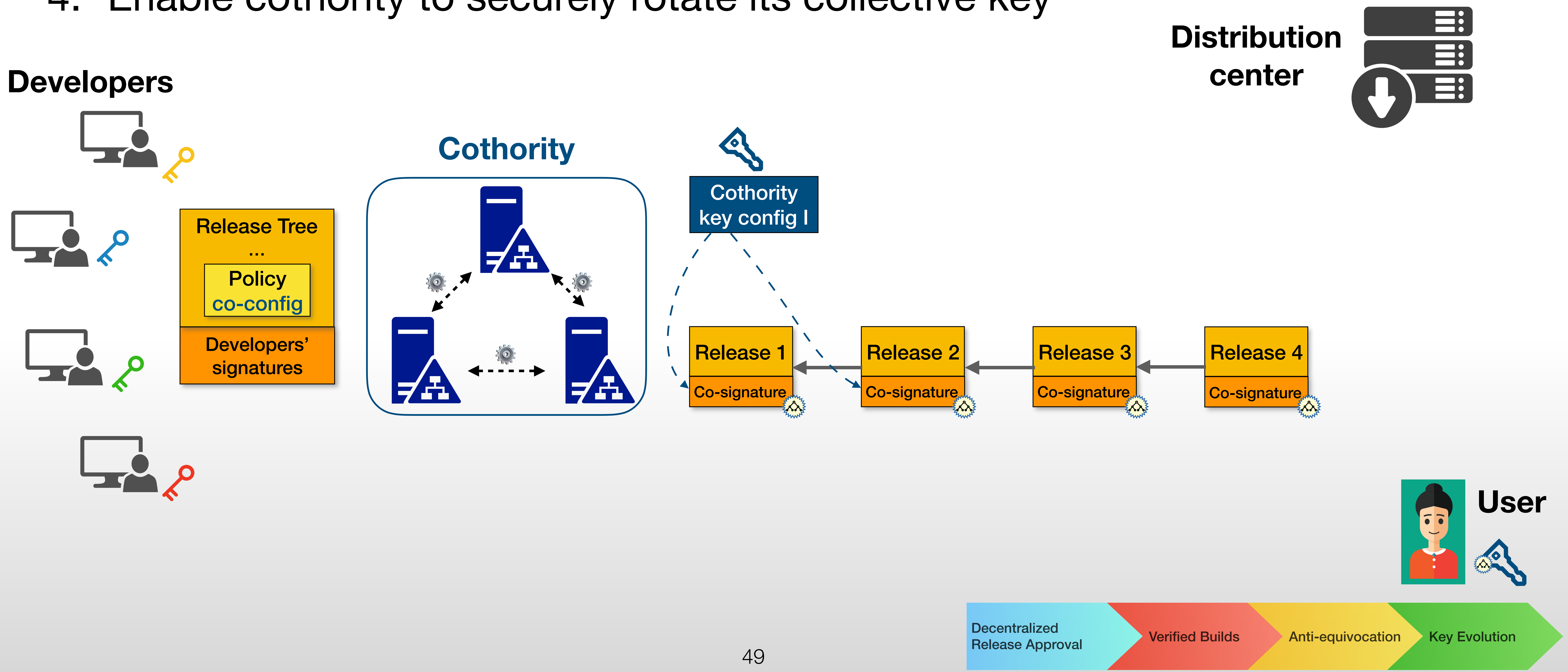
Evolution of Developer Keys

4. Enable developers to securely rotate their signing keys



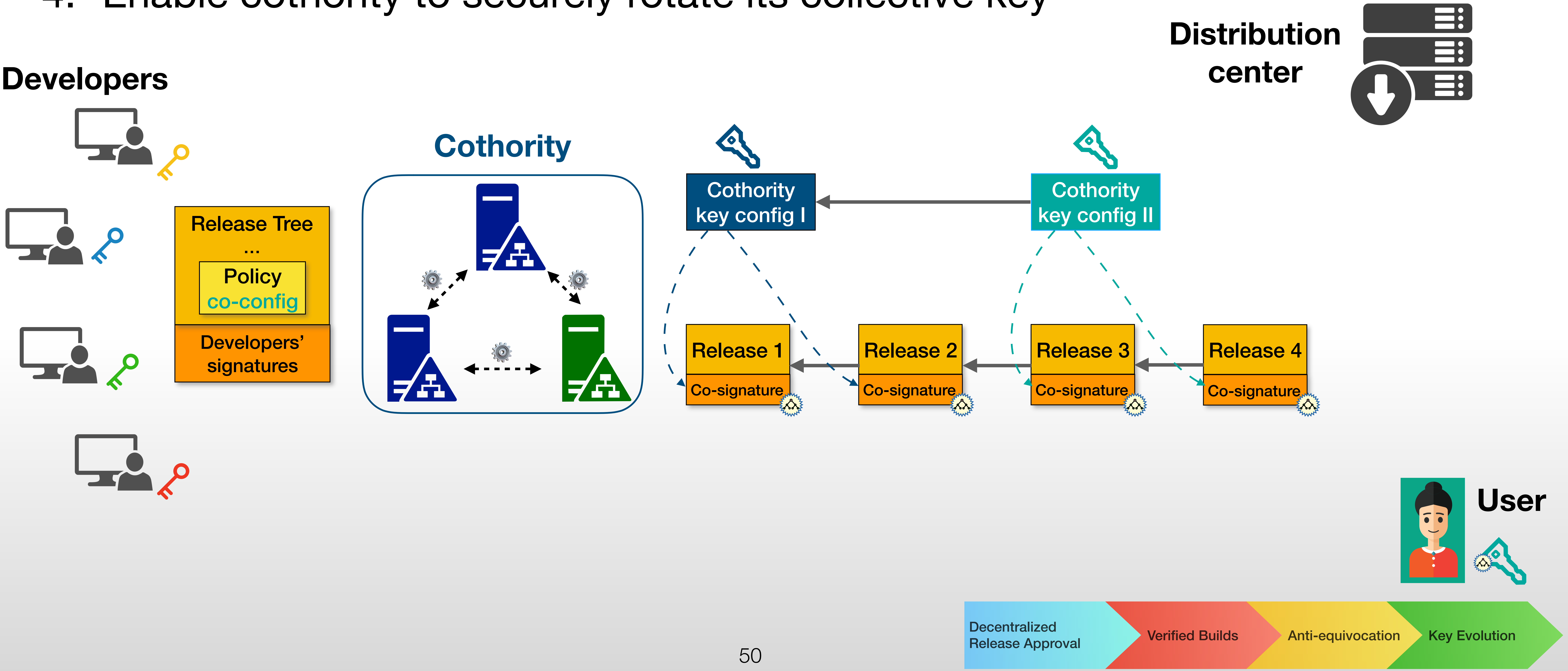
Evolution of Cothority Configuration

4. Enable cothority to securely rotate its collective key



Evolution of Cothority Configuration

4. Enable cothority to securely rotate its collective key

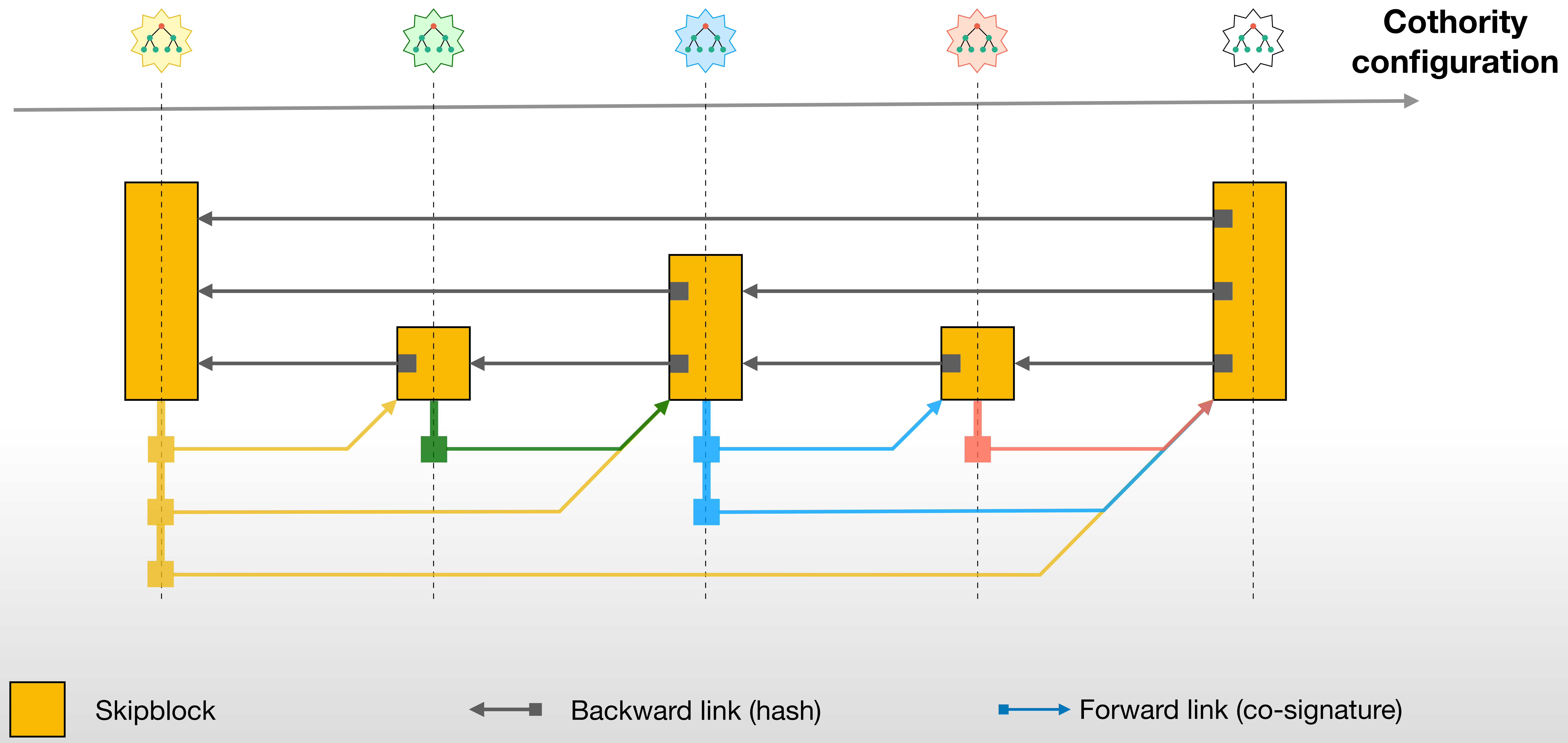


Skipchains

Skipchains

- Novel data structure: blockchain + skip lists
- Blocks have multi-hop two-way links:
  - *Backward links* - hashes of past blocks
  - *Forward links* - (collective) signatures
- Secure and efficient traversal of arbitrary long timelines

Skipchains



Cothority configuration
Skipblock
Backward link (hash)
Forward link (co-signature)

Implementation and Evaluation

Implementation

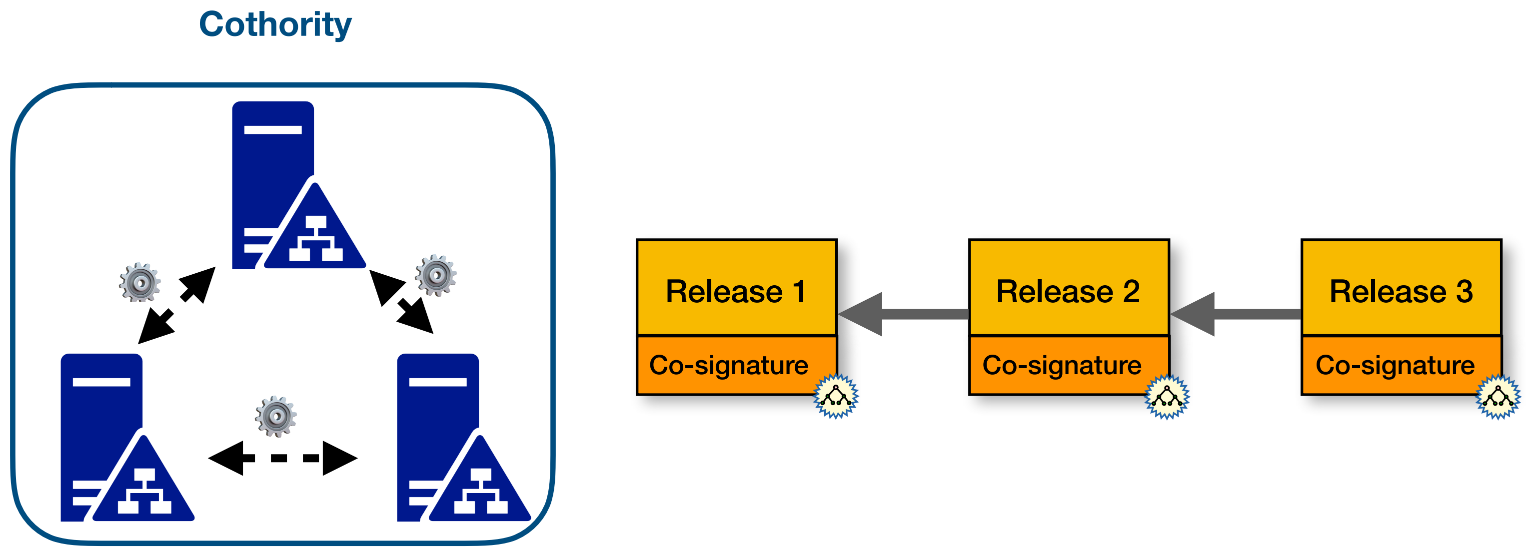
- CHAINIAC is implemented in Go
  - Using the DEDIS Kyber crypto library and Onet networking framework
  - Available open-source at https://github.com/dedis/paper_chainiac

Evaluation Methodology

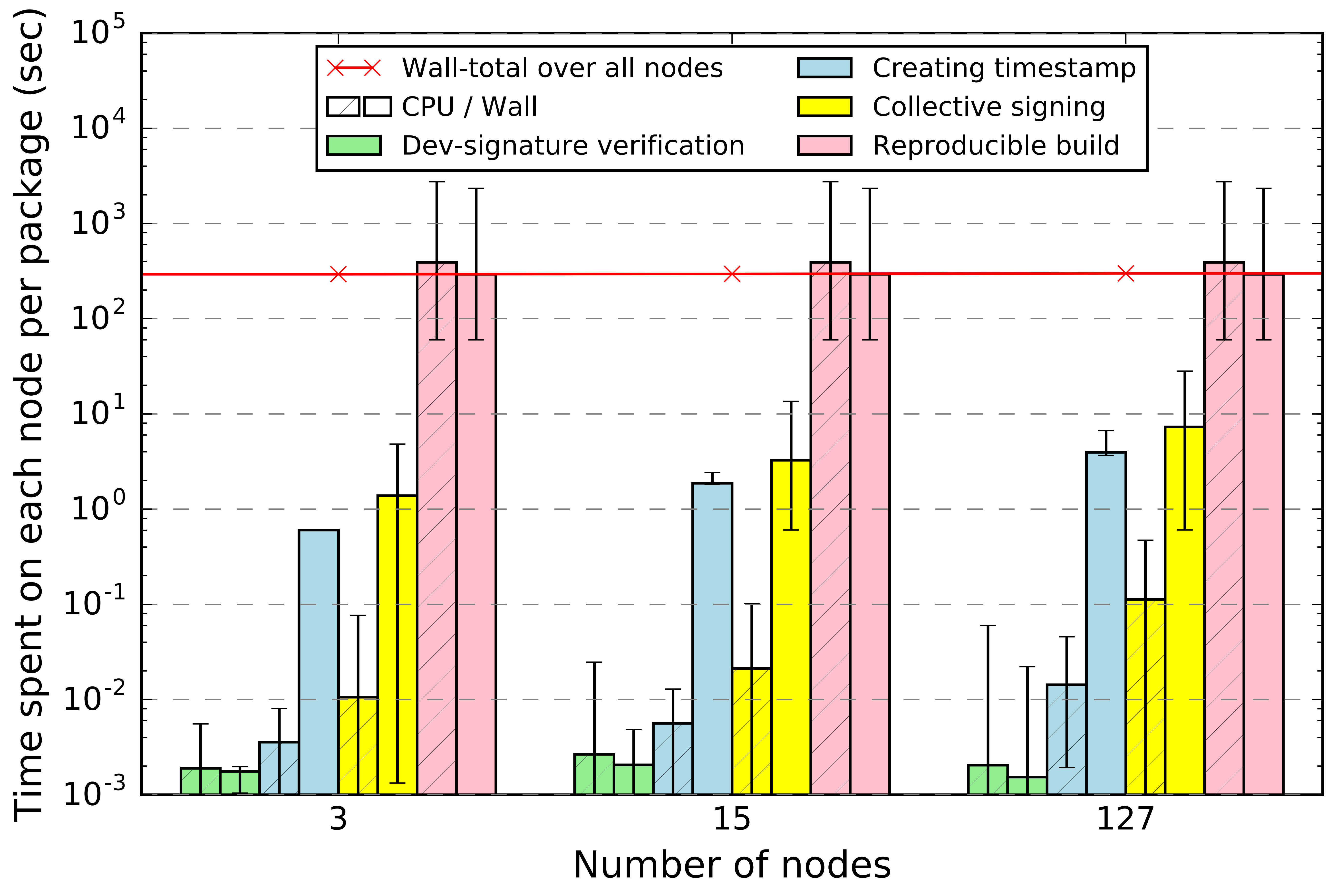
What is the cost effect of CHAINIAC on cothority nodes and on clients?

- Cothority-node CPU cost of validating releases and maintaining transparency release log
  - The average values for six Debian packages over two years

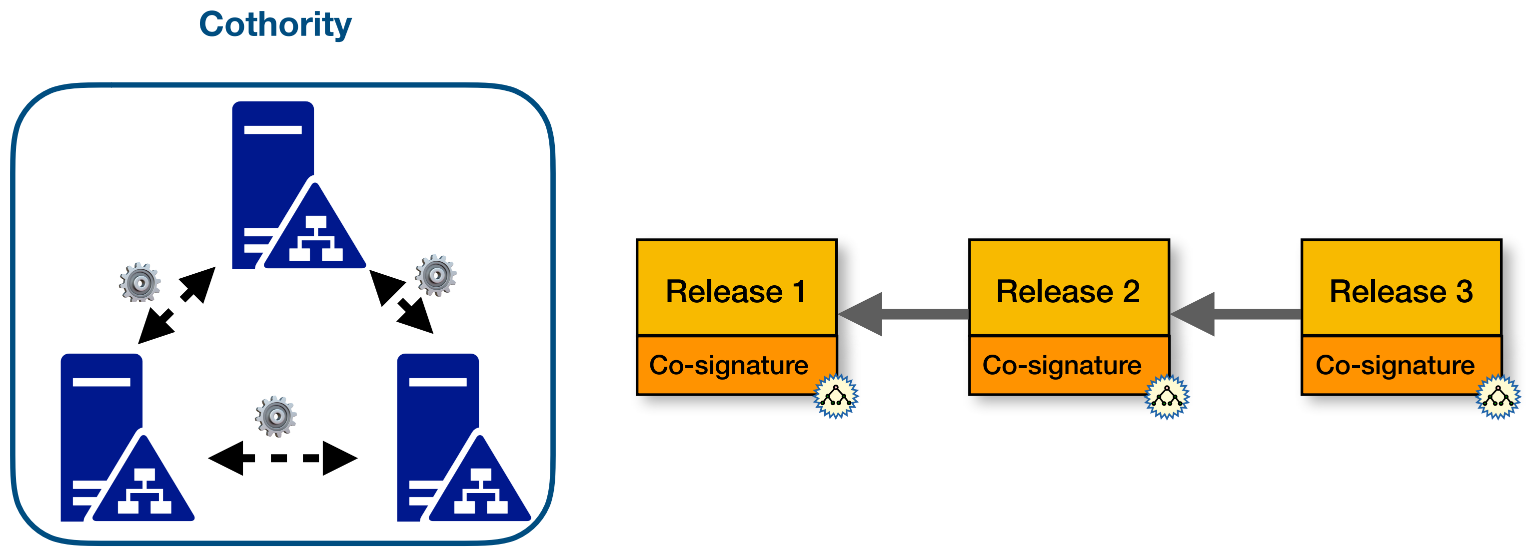
Evaluation



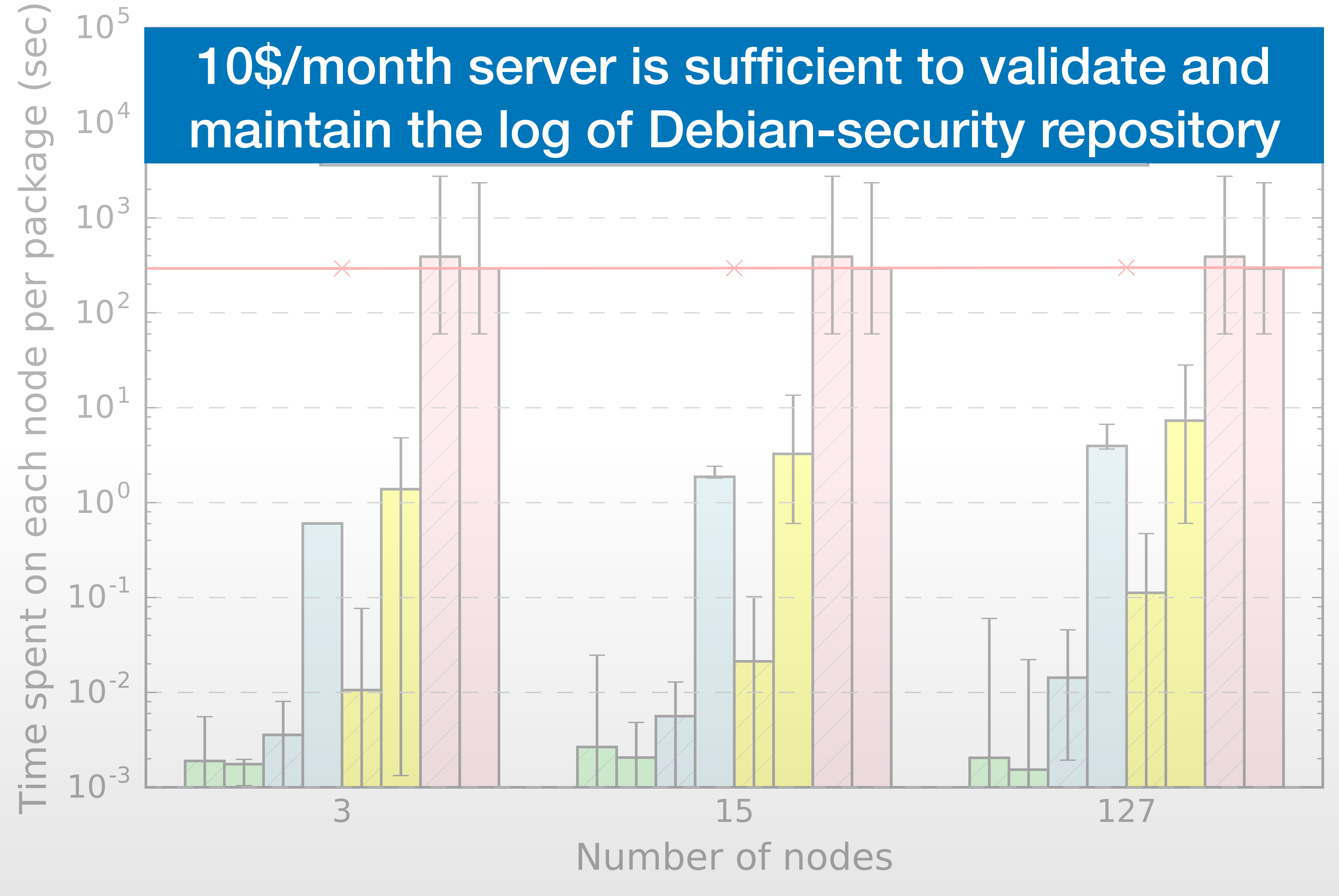
1. Cothority-node CPU cost of validating releases and maintaining release log



Evaluation



1. Cothority-node CPU cost of validating releases and maintaining release log



Evaluation Methodology

What is the cost effect of CHAINIAC on cothority nodes and on clients?

- Cothority-node CPU cost of validating releases and maintaining transparency release log
  - The average values of six *required* Debian packages
- CPU cost of reproducing packages on cothority nodes
  - From 1.5 to 30 minutes to reproduce a package
- Skipchain effect on communication cost
  - Reducing the cost by the factor of 30 on 1.5 million update-requests from the PyPI repository
- CPU and bandwidth cost of securing a multi-package distribution
  - ~20 sec to create a snapshot of >50k-packages Debian repository

Conclusion

- CHAINIAC decentralizes each step of the software-update process to increase trustworthiness and to eliminate single points of failure

- Skipchain structure for efficient logging and secure key evolution;
  See https://bford.github.io/2017/08/01/skipchain/ for more applications

- Verified builds as an improvement over reproducible builds

- Role-based architecture, multi-package Chainiac and more are in the paper

---

Kirill Nikitin

kirill.nikitin@epfl.ch

@ni_kirill