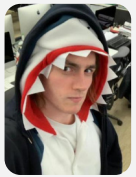
Secure Messaging: Leveraging Rust to Create the Guardian's Anonymous Whistleblowing System

RustConf 2025, Seattle



Sam Cutler

@itsibitzi.dev

Daniel Hugenroth

@lambda.bsky.social

Zeke Hunter-Green

@zekehg.bsky.social

The Guardian

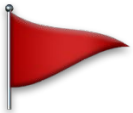
UNIVERSITY OF CAMBRIDGE

Imagine you have discovered wrongdoings and are ready to talk a journalist…

how would you do that?

The **first contact** is
particularly difficult

Avoid leaving **digital footprints**
from the beginning

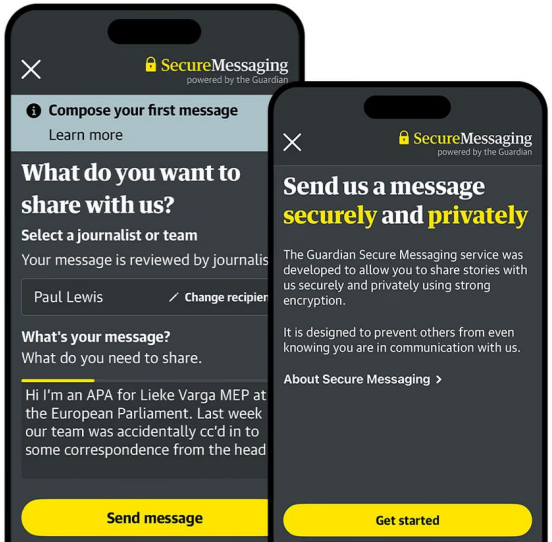


Using anonymity networks (e.g. Tor)
can make one **stand-out**…

…and they can be
difficult to use

Blowing the whistle is impactful, but tricky.

Secure Messaging

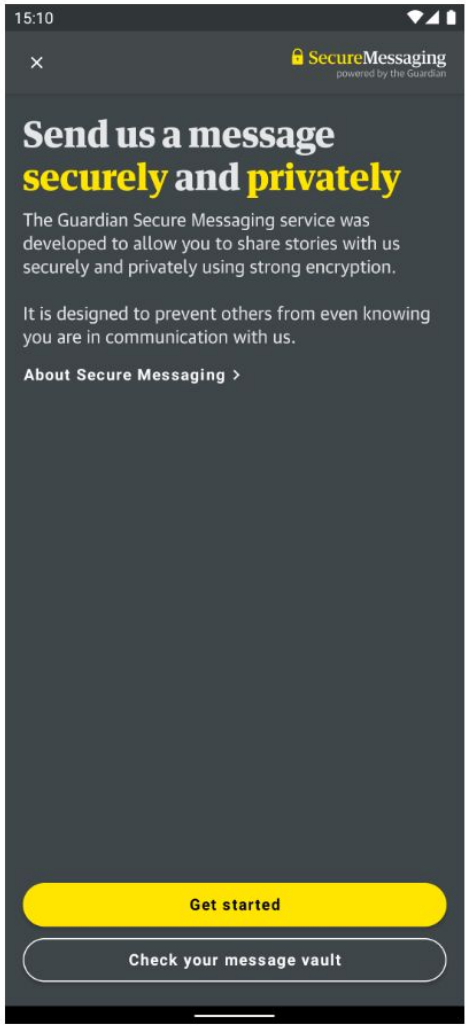


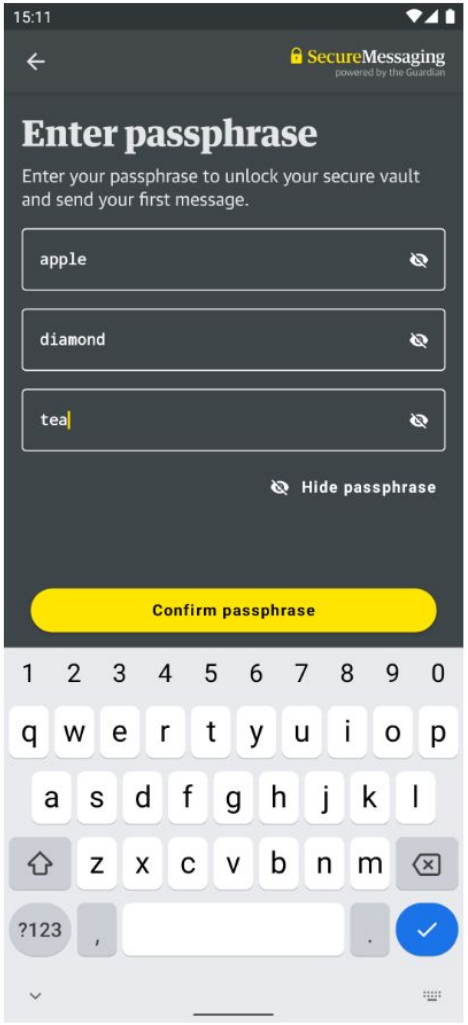
Embedded and activated in every Guardian app

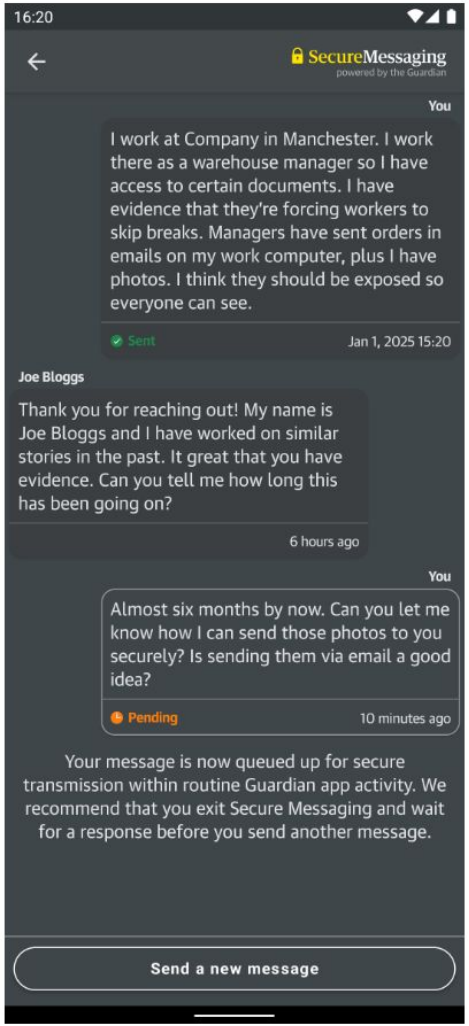
Cover traffic protects metadata of real whistleblower messages

Use existing user base to build large anonymity set

Strong **plausible deniability** for sources


15:10
SecureMessaging
powered by the Guardian
Send us a message
securely and privately
The Guardian Secure Messaging service was developed to allow you to share stories with us securely and privately using strong encryption.
It is designed to prevent others from even knowing you are in communication with us.
About Secure Messaging >
Get started
Check your message vault


15:11
SecureMessaging
powered by the Guardian
Enter passphrase
Enter your passphrase to unlock your secure vault and send your first message.
apple
diamond
tea
Hide passphrase
Confirm passphrase
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123


16:20
SecureMessaging
powered by the Guardian
You
I work at Company in Manchester. I work there as a warehouse manager so I have access to certain documents. I have evidence that they're forcing workers to skip breaks. Managers have sent orders in emails on my work computer, plus I have photos. I think they should be exposed so everyone can see.
Sent
Jan 1, 2025 15:20
Joe Bloggs
Thank you for reaching out! My name is Joe Bloggs and I have worked on similar stories in the past. It great that you have evidence. Can you tell me how long this has been going on?
6 hours ago
You
Almost six months by now. Can you let me know how I can send those photos to you securely? Is sending them via email a good idea?
Pending
10 minutes ago
Your message is now queued up for secure transmission within routine Guardian app activity. We recommend that you exit Secure Messaging and wait for a response before you send another message.
Send a new message

UNIVERSITY OF CAMBRIDGE

August 2022

sciendo

Proceedings on Privacy Enhancing Technologies ; 2022 (2):47–67

Mansoor Ahmed-Rengers*, Diana A. Vasile*, Daniel Hugenroth*, Alastair R. Beresford, and Ross Anderson

CoverDrop: Blowing the Whistle Through A News App

Abstract: Whistleblowing is hazardous in a world of pervasive surveillance, yet many leading newspapers expect sources to contact them with methods that are either insecure or barely usable. In an attempt to do better, we conducted two workshops with British news organisations and surveyed whistleblowing options and guidelines at major media outlets. We concluded that the soft spot is a system for initial contact and trust establishment between sources and reporters. *CoverDrop* is a two-way, secure system to do this. We support secure messaging within a news app, so that all its other users provide cover traffic, which we channel through a threshold mix instantiated in a Trusted Execution Environment within the news organisation. CoverDrop is designed to resist a powerful global adversary with the ability to issue warrants against infrastructure providers, yet it can easily be integrated into existing infrastructure. We present the results from our workshops, describe CoverDrop's design and demonstrate its security and performance.

Keywords: whistleblowing, anonymous communication, mobile application

DOI 10.2478/popets-2022-0035
Received 2021-08-31; revised 2021-12-15; accepted 2021-12-16.

*Corresponding Author: Mansoor Ahmed-Rengers: OpenOrigins Limited and University of Cambridge, E-mail: Mansoor.Ahmed@cl.cam.ac.uk
*Corresponding Author: Diana A. Vasile: Department of Computer Science and Technology, University of Cambridge, E-mail: Diana.Vasile@cl.cam.ac.uk
*Corresponding Author: Daniel Hugenroth: Department of Computer Science and Technology, University of Cambridge, E-mail: Daniel.Hugenroth@cl.cam.ac.uk
Alastair R. Beresford: Department of Computer Science and Technology, University of Cambridge, E-mail: Alastair.Beresford@cl.cam.ac.uk
Ross Anderson: Department of Computer Science and Technology, University of Cambridge, E-mail: Ross.Anderson@cl.cam.ac.uk

1 Introduction

Since the Snowden leaks [15], newspapers and their potential sources have become aware of the mass-surveillance infrastructure available to nation states. This has profound implications for those who wish to expose wrongdoing within government, and in organisations that can call on its assistance.

Whistleblowers may face severe penalties if caught: in the recent past, they have been physically intimidated [28], imprisoned [46, 51], and even assassinated [45]. Even in less extreme cases, they face dismissal [35], litigation, and professional boycotts [41]. Yet, whistleblowing is often the last line of defence against unethical or illegal behaviour by the powerful. In the context of government agencies, it may be the principal means of exposing crimes that may have been covered up under the guise of national security. It is also widely recognised as a crucial component in accountability, and many countries have explicit laws for protecting whistleblowers. Unfortunately, these laws have proven to be insufficient in many recent cases. Since the Snowden revelations, we have seen the emergence of tools such as SecureDrop, and many major news organisations have web pages to advise whistleblowers on how to contact them securely with sensitive information. A preliminary examination convinced us that the tools are often hard to use securely and the advice inadequate. How could we do better?

In order to understand the whistleblowing process in the real world, we conducted two workshops with journalists and IT staff at news organisations. These workshops provided valuable insights into the practical difficulties of supporting whistleblowers and how journalists work with them. The main lesson we drew from these workshops was that there is no secure way for whistleblowers to initiate contact with reporters. Consequently, initial contact is undertaken via insecure means and only later escalated to more secure channels (e.g., SecureDrop), by which point it may already be too late.

In this paper, we present *CoverDrop*, a system which helps potential whistleblowers securely initiate contact and whose design emerged from these discus-

Mansoor Ahmed

Daniel Hugenroth

Diana A. Vasile

Alastair R. Beresford

Ross Anderson

UNIVERSITY OF CAMBRIDGE

The Guardian

August 2022

$ sciendo

Proceedings on Privacy Enhancing Technologies ; 2022 (2):47–67

Mansoor Ahmed-Rengers*, Diana A. Vasile*, Daniel Hugenroth*, Alastair R. Beresford, and Ross Anderson

CoverDrop: Blowing the Whistle Through A News App

Abstract: Whistleblowing is hazardous in a world of pervasive surveillance, yet many leading newspapers expect sources to contact them with methods that are either insecure or barely usable. In an attempt to do better, we conducted two workshops with British news organisations and surveyed whistleblowing options and guidelines at major media outlets. We concluded that the soft spot is a system for initial contact and trust establishment between sources and reporters. *CoverDrop* is a two-way, secure system to do this. We support secure messaging within a news app, so that all its other users provide cover traffic, which we channel through a threshold mix instantiated in a Trusted Execution Environment within the news organisation. CoverDrop is designed to resist a powerful global adversary with the ability to issue warrants against infrastructure providers, yet it can easily be integrated into existing infrastructure. We present the results from our workshops, describe CoverDrop's design and demonstrate its security and performance.

Keywords: whistleblowing, anonymous communication, mobile application

DOI 10.2478/popets-2022-0035
Received 2021-08-31; revised 2021-12-15; accepted 2021-12-16.

***Corresponding Author: Mansoor Ahmed-Rengers:** OpenOrigins Limited and University of Cambridge, E-mail: Mansoor.Ahmed@cl.cam.ac.uk
***Corresponding Author: Diana A. Vasile:** Department of Computer Science and Technology, University of Cambridge, E-mail: Diana.Vasile@cl.cam.ac.uk
***Corresponding Author: Daniel Hugenroth:** Department of Computer Science and Technology, University of Cambridge, E-mail: Daniel.Hugenroth@cl.cam.ac.uk
Alastair R. Beresford: Department of Computer Science and Technology, University of Cambridge, E-mail: Alastair.Beresford@cl.cam.ac.uk
Ross Anderson: Department of Computer Science and Technology, University of Cambridge, E-mail: Ross.Anderson@cl.cam.ac.uk

1 Introduction

Since the Snowden leaks [15], newspapers and their potential sources have become aware of the mass-surveillance infrastructure available to nation states. This has profound implications for those who wish to expose wrongdoing within government, and in organisations that can call on its assistance.

Whistleblowers may face severe penalties if caught: in the recent past, they have been physically intimidated [28], imprisoned [46, 51], and even assassinated [45]. Even in less extreme cases, they face dismissal [35], litigation, and professional boycotts [41]. Yet, whistleblowing is often the last line of defence against unethical or illegal behaviour by the powerful. In the context of government agencies, it may be the principal means of exposing crimes that may have been covered up under the guise of national security. It is also widely recognised as a crucial component in accountability, and many countries have explicit laws for protecting whistleblowers. Unfortunately, these laws have proven to be insufficient in many recent cases. Since the Snowden revelations, we have seen the emergence of tools such as SecureDrop, and many major news organisations have web pages to advise whistleblowers on how to contact them securely with sensitive information. A preliminary examination convinced us that the tools are often hard to use securely and the advice inadequate. How could we do better?

In order to understand the whistleblowing process in the real world, we conducted two workshops with journalists and IT staff at news organisations. These workshops provided valuable insights into the practical difficulties of supporting whistleblowers and how journalists work with them. The main lesson we drew from these workshops was that there is no secure way for whistleblowers to initiate contact with reporters. Consequently, initial contact is undertaken via insecure means and only later escalated to more secure channels (e.g., SecureDrop), by which point it may already be too late.

In this paper, we present *CoverDrop*, a system which helps potential whistleblowers securely initiate contact and whose design emerged from these discus-

Hey, this looks really interesting. Let's build this! Should not take more than a few months… 🖥️

UNIVERSITY OF CAMBRIDGE

August 2022

sciendo

Proceedings on Privacy Enhancing Technologies ; 2022 (2):47–67

Mansoor Ahmed-Rengers*, Diana A. Vasile*, Daniel Hugenroth*, Alastair R. Beresford, and Ross Anderson

CoverDrop: Blowing the Whistle Through A News App

Abstract: Whistleblowing is hazardous in a world of pervasive surveillance, yet many leading newspapers expect sources to contact them with methods that are either insecure or barely usable. In an attempt to do better, we conducted two workshops with British news organisations and surveyed whistleblowing options and guidelines at major media outlets. We concluded that the soft spot is a system for initial contact and trust establishment between sources and reporters. *CoverDrop* is a two-way, secure system to do this. We support secure messaging within a news app, so that all its other users provide cover traffic, which we channel through a threshold mix instantiated in a Trusted Execution Environment within the news organisation. CoverDrop is designed to resist a powerful global adversary with the ability to issue warrants against infrastructure providers, yet it can easily be integrated into existing infrastructure. We present the results from our workshops, describe CoverDrop's design and demonstrate its security and performance.

Keywords: whistleblowing, anonymous communication, mobile application

DOI 10.2478/popets-2022-0035
Received 2021-08-31; revised 2021-12-15; accepted 2021-12-16.

1 Introduction

Since the Snowden leaks [15], newspapers and their potential sources have become aware of the mass-surveillance infrastructure available to nation states. This has profound implications for those who wish to expose wrongdoing within government, and in organisations that can call on its assistance.

Whistleblowers may face severe penalties if caught: in the recent past, they have been physically intimidated [28], imprisoned [46, 51], and even assassinated [45]. Even in less extreme cases, they face dismissal [35], litigation, and professional boycotts [41]. Yet, whistleblowing is often the last line of defence against unethical or illegal behaviour by the powerful. In the context of government agencies, it may be the principal means of exposing crimes that may have been covered up under the guise of national security. It is also widely recognised as a crucial component in accountability, and many countries have explicit laws for protecting whistleblowers. Unfortunately, these laws have proven to be insufficient in many recent cases. Since the Snowden revelations, we have seen the emergence of tools such as SecureDrop, and many major news organisations have web pages to advise whistleblowers on how to contact them securely with sensitive information. A preliminary examination convinced us that the tools are often hard to use securely and the advice inadequate. How could we do better?

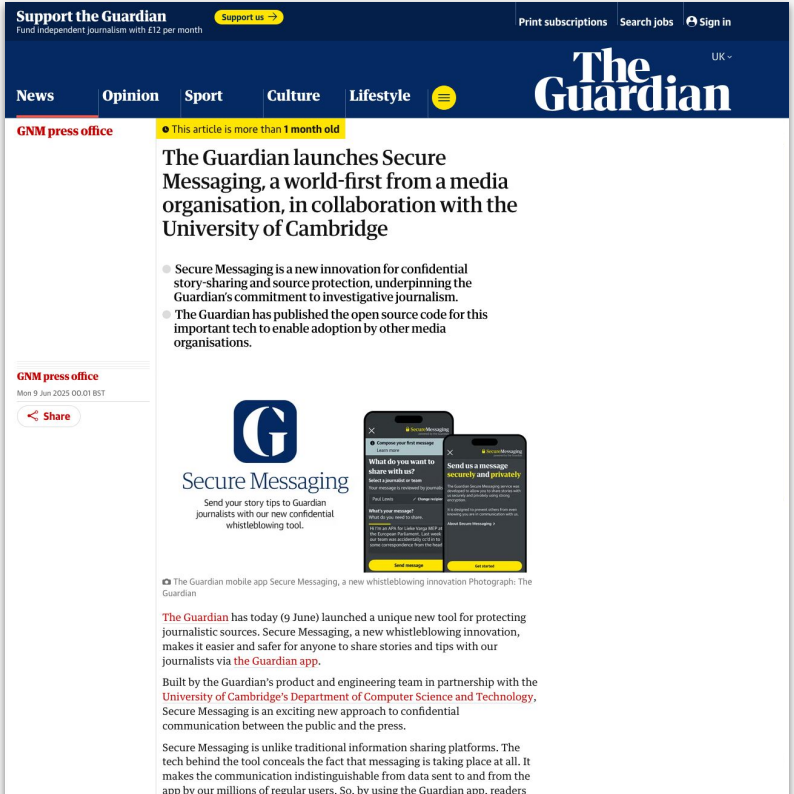
In order to understand the whistleblowing process in the real world, we conducted two workshops with journalists and IT staff at news organisations. These workshops provided valuable insights into the practical difficulties of supporting whistleblowers and how journalists work with them. The main lesson we drew from these workshops was that there is no secure way for whistleblowers to initiate contact with reporters. Consequently, initial contact is undertaken via insecure means and only later escalated to more secure channels (e.g., SecureDrop), by which point it may already be too late.

In this paper, we present *CoverDrop*, a system which helps potential whistleblowers securely initiate contact and whose design emerged from these discus-

*Corresponding Author: Mansoor Ahmed-Rengers: OpenOrigins Limited and University of Cambridge, E-mail: Mansoor.Ahmed@cl.cam.ac.uk
*Corresponding Author: Diana A. Vasile: Department of Computer Science and Technology, University of Cambridge, E-mail: Diana.Vasile@cl.cam.ac.uk
*Corresponding Author: Daniel Hugenroth: Department of Computer Science and Technology, University of Cambridge, E-mail: Daniel.Hugenroth@cl.cam.ac.uk
Alastair R. Beresford: Department of Computer Science and Technology, University of Cambridge, E-mail: Alastair.Beresford@cl.cam.ac.uk
Ross Anderson: Department of Computer Science and Technology, University of Cambridge, E-mail: Ross.Anderson@cl.cam.ac.uk

The Guardian

June 2025



Support the Guardian
Fund independent journalism with £12 per month
Support us
Print subscriptions | Search jobs | Sign in
UK

The Guardian

News | Opinion | Sport | Culture | Lifestyle

GNM press office

This article is more than 1 month old

The Guardian launches Secure Messaging, a world-first from a media organisation, in collaboration with the University of Cambridge

- Secure Messaging is a new innovation for confidential story-sharing and source protection, underpinning the Guardian's commitment to investigative journalism.
- The Guardian has published the open source code for this important tech to enable adoption by other media organisations.

GNM press office
Mon 9 Jun 2025 00.01 BST

Share

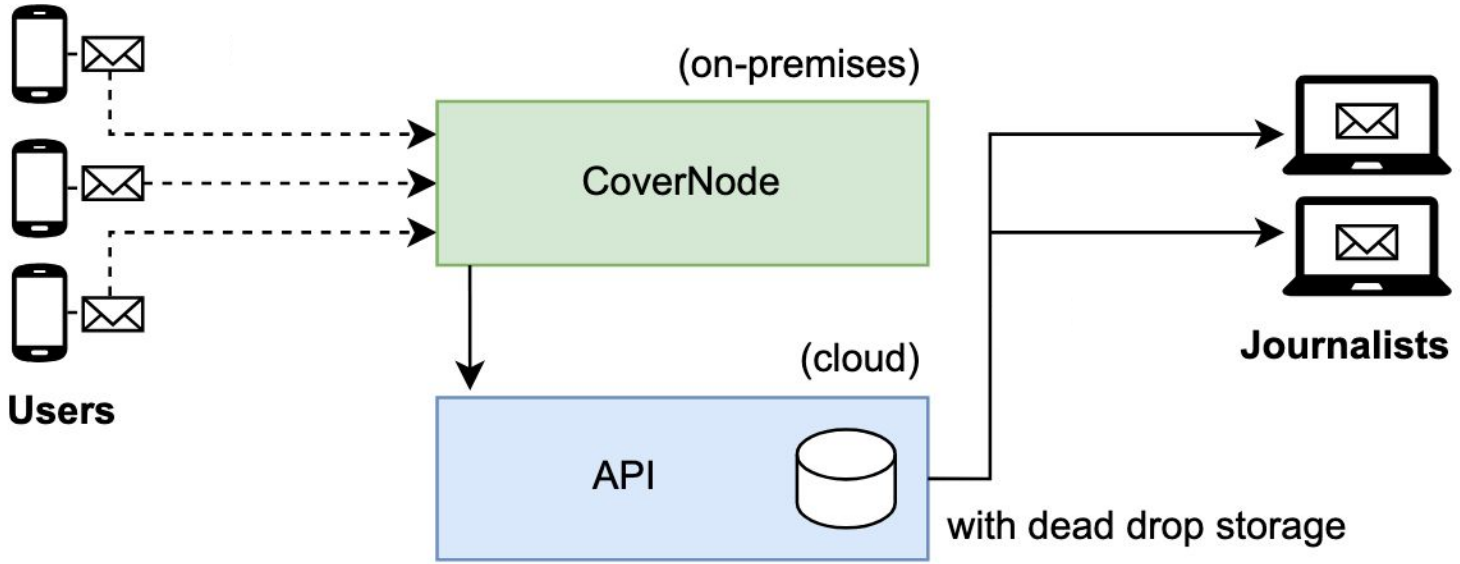
Secure Messaging
Send your story tips to Guardian journalists with our new confidential whistleblowing tool.

The Guardian mobile app Secure Messaging, a new whistleblowing innovation Photograph: The Guardian

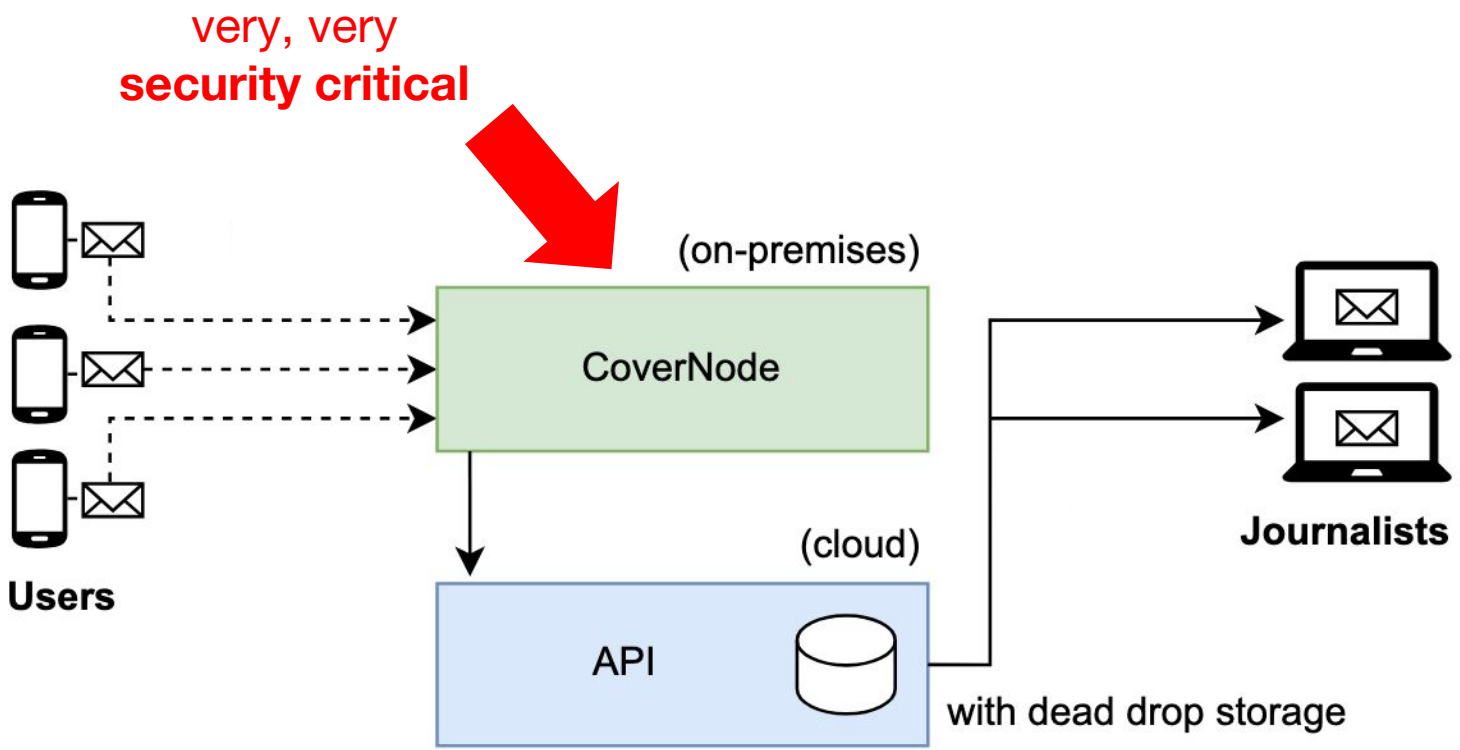
The Guardian has today (9 June) launched a unique new tool for protecting journalistic sources. Secure Messaging, a new whistleblowing innovation, makes it easier and safer for anyone to share stories and tips with our journalists via the Guardian app.

Built by the Guardian's product and engineering team in partnership with the University of Cambridge's Department of Computer Science and Technology, Secure Messaging is an exciting new approach to confidential communication between the public and the press.

Secure Messaging is unlike traditional information sharing platforms. The tech behind the tool conceals the fact that messaging is taking place at all. It makes the communication indistinguishable from data sent to and from the app by our millions of regular users. So, by using the Guardian app, readers


(on-premises)
CoverNode
(cloud)
API
with dead drop storage
Users
Journalists

So, where does **Rust** fit into here?

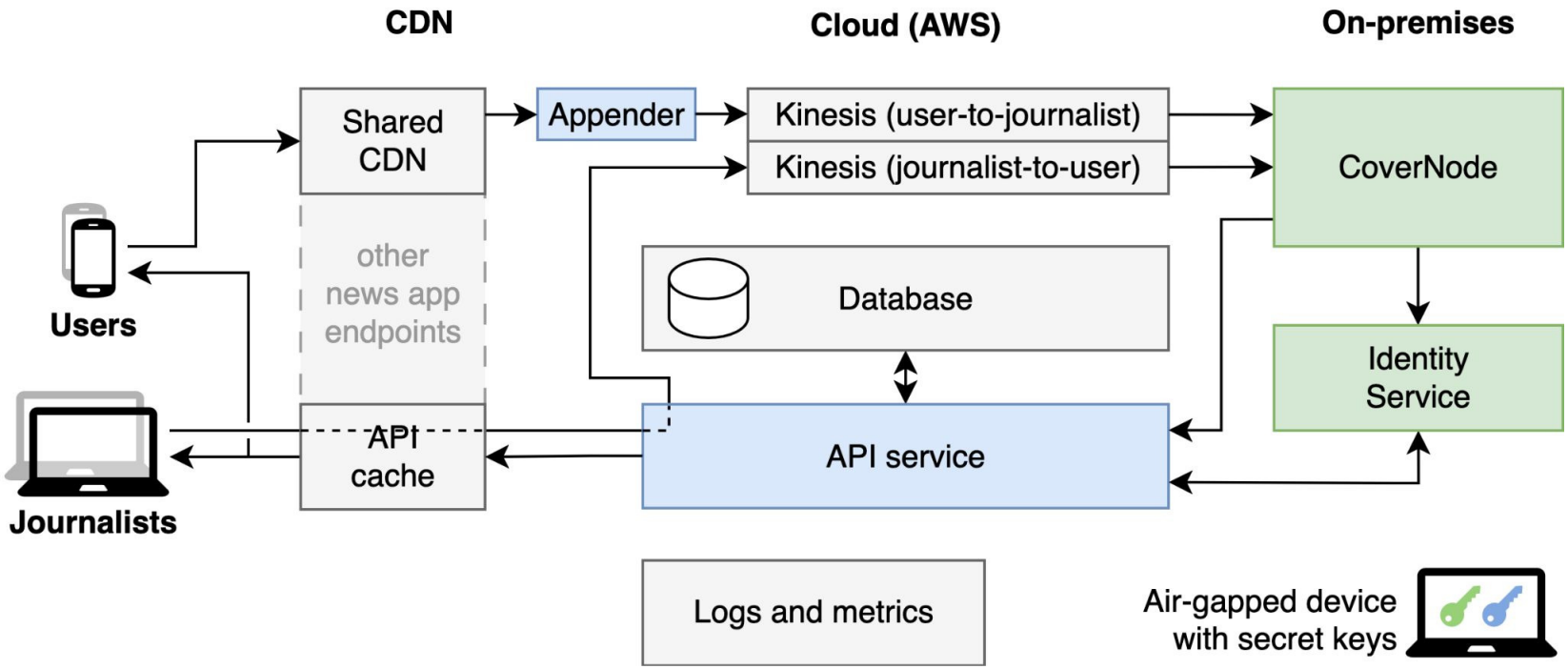

very, very
security critical
(on-premises)
CoverNode
Users
(cloud)
API
with dead drop storage
Journalists

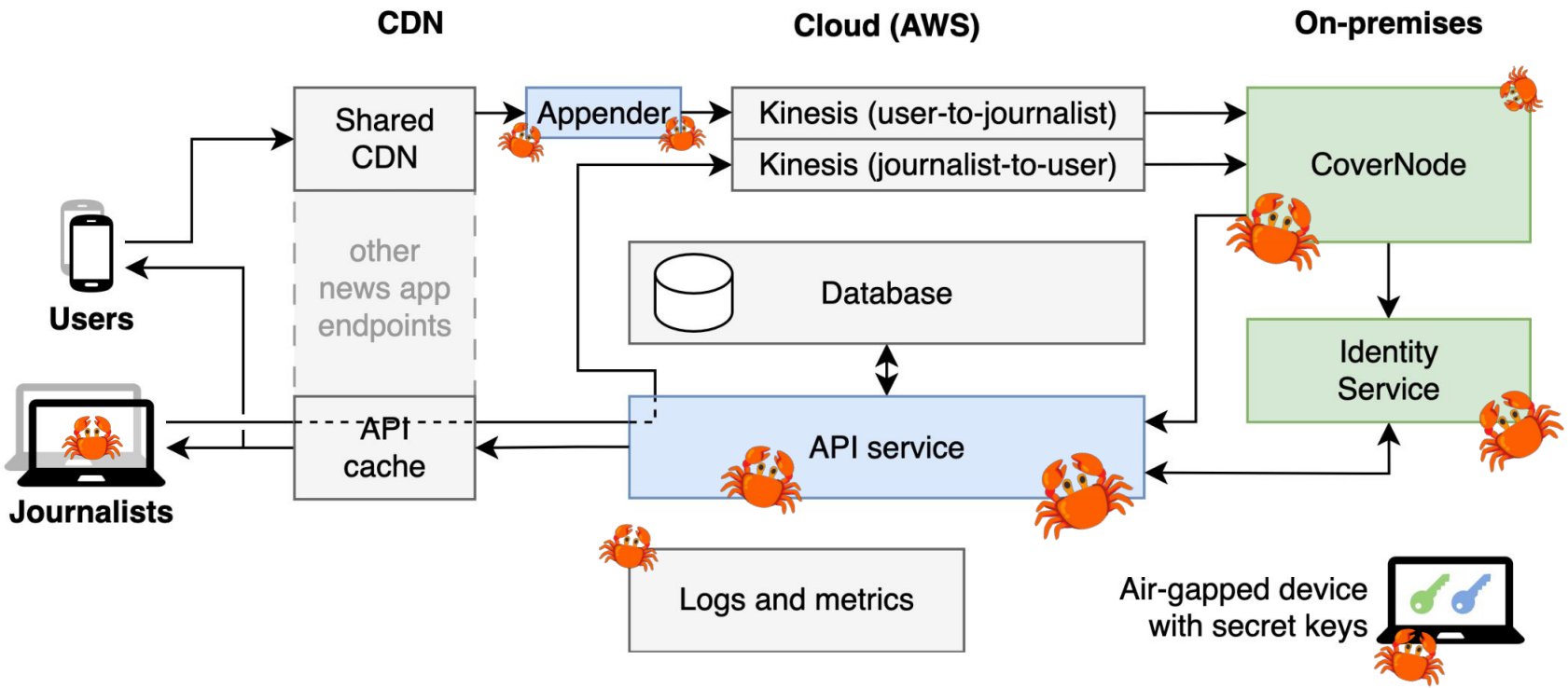
Leveraging **types** for
guaranteeing safety

Integrated **testing and**
observability approaches

Making **performance**
a non-issue

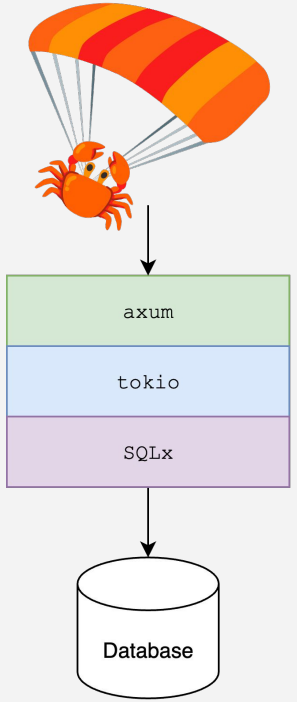
It’s all about building meaningful systems and still **sleeping well at night**.


CDN
Cloud (AWS)
On-premises
Shared CDN
Appender
Kinesis (user-to-journalist)
Kinesis (journalist-to-user)
CoverNode
other news app endpoints
Database
Identity Service
Users
API cache
API service
Journalists
Logs and metrics
Air-gapped device with secret keys

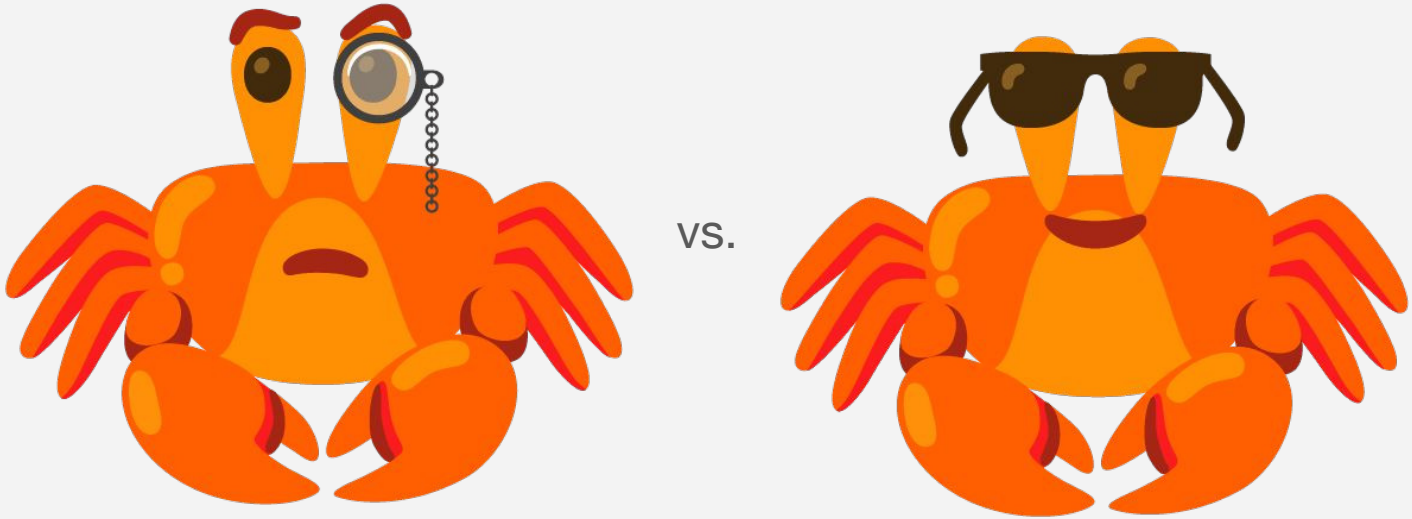

CDN
Cloud (AWS)
On-premises
Shared CDN
Appender
Kinesis (user-to-journalist)
Kinesis (journalist-to-user)
CoverNode
other news app endpoints
Database
Identity Service
Users
API cache
API service
Journalists
Logs and metrics
Air-gapped device with secret keys

Our Rust project setup

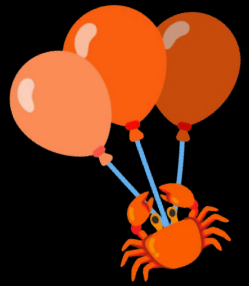
- **`tokio/async`** Rust throughout
- **`axum`** for web services
- **`anyhow`** for error handling
- **`sqlx`** for talking to the database



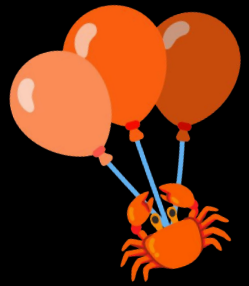
Blazingly fast vs. easy-mode?



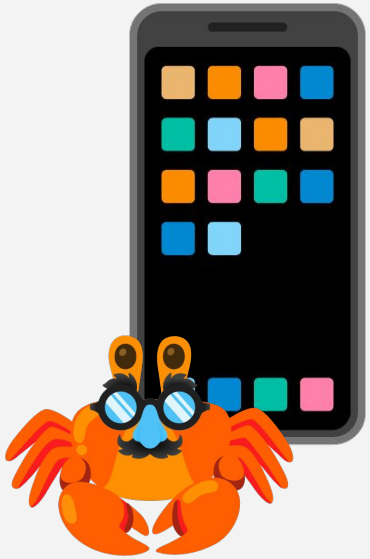
How can we use Rust to improve **misuse resistance**?



Types!



New Types!



Compressed

Padded

Encrypted

CoverNode

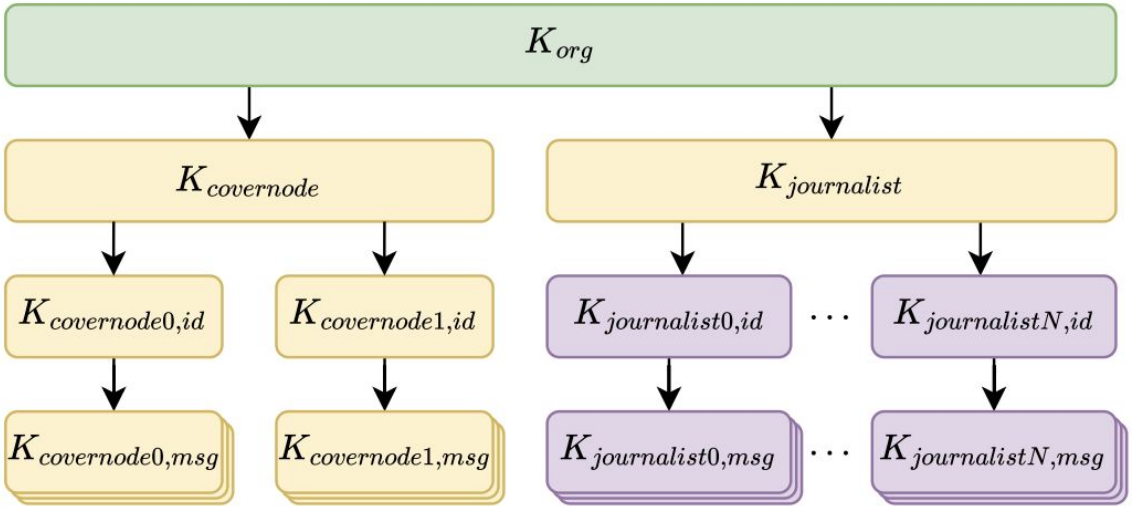
```rust
#[derive(Debug, Clone, Eq, PartialEq, Serialize, Deserialize)]
#[serde(transparent, deny_unknown_fields)]
pub struct PaddedCompressedString<const PAD_TO: usize>(Vec<u8>);
```

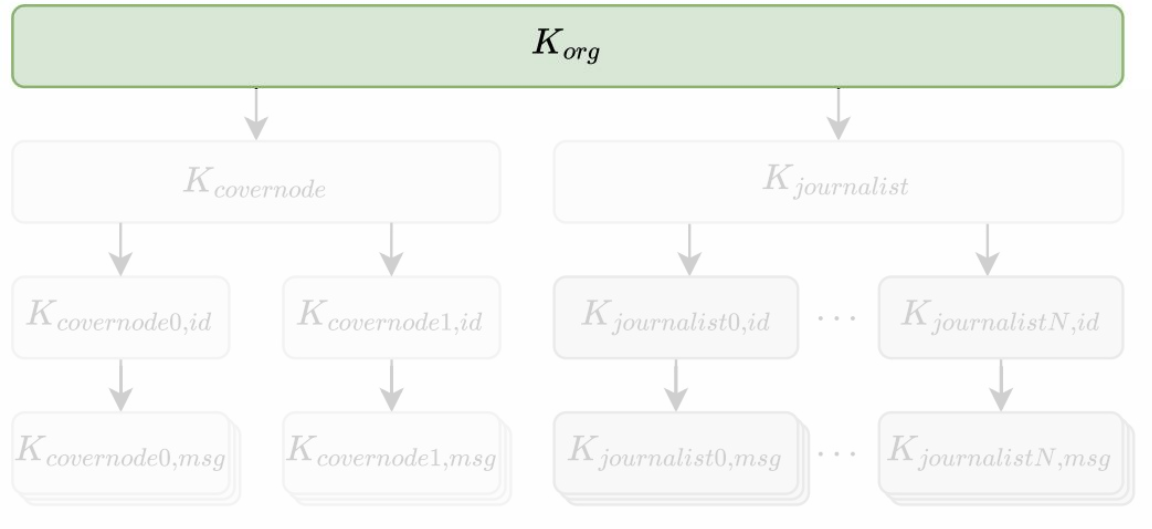
```rust
#[derive(Debug, Clone, Eq, PartialEq, Serialize, Deserialize)]
#[serde(transparent, deny_unknown_fields)]
pub struct PaddedCompressedString<const PAD_TO: usize>(Vec<u8>);
```

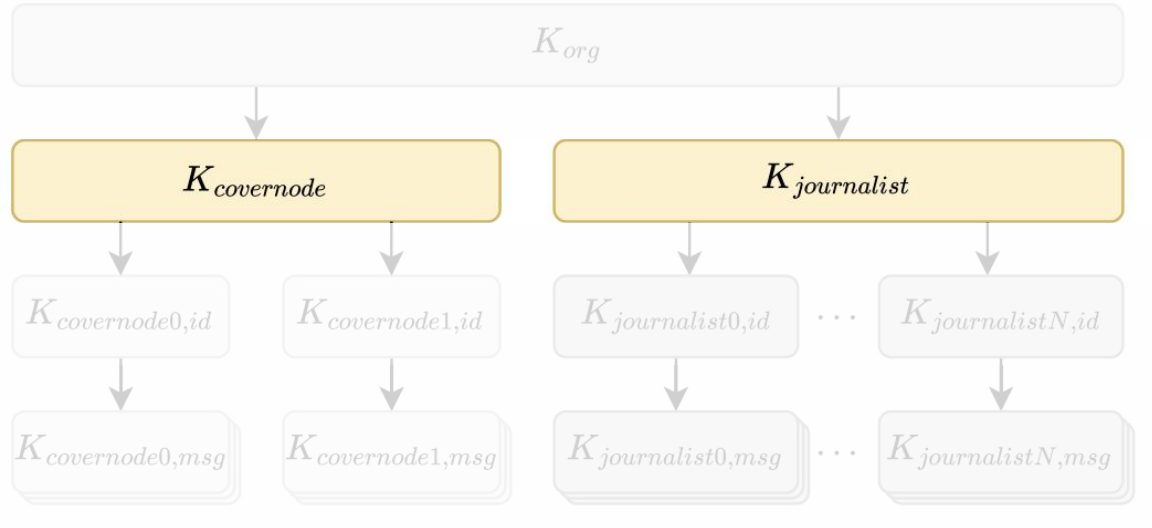
```rust
#[derive(Debug, Clone, Eq, PartialEq, Serialize, Deserialize)]
#[serde(transparent, deny_unknown_fields)]
pub struct PaddedCompressedString<const PAD_TO: usize>(Vec<u8>);
```

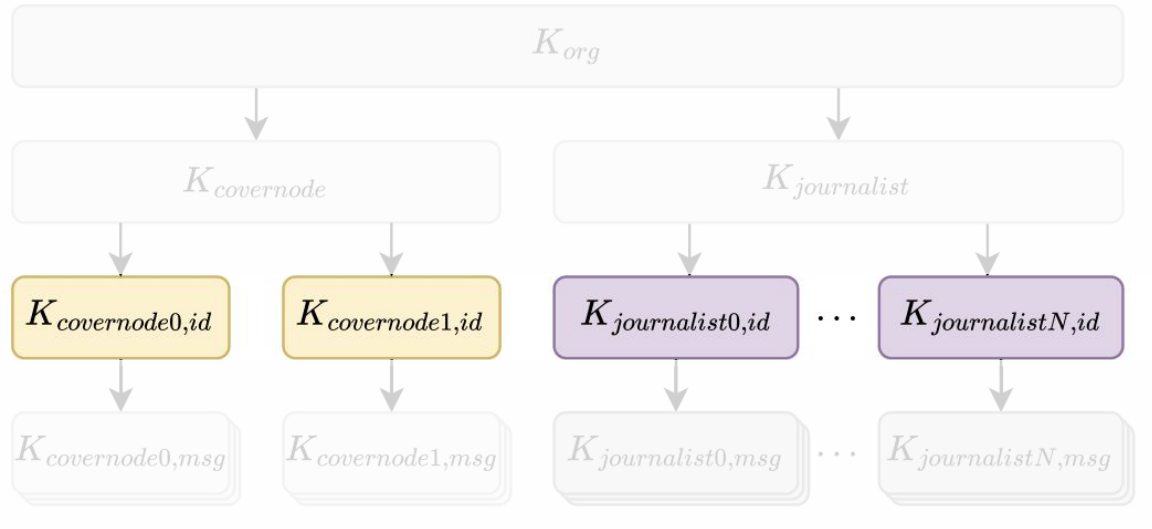
Provides **extra properties**
to basic types!

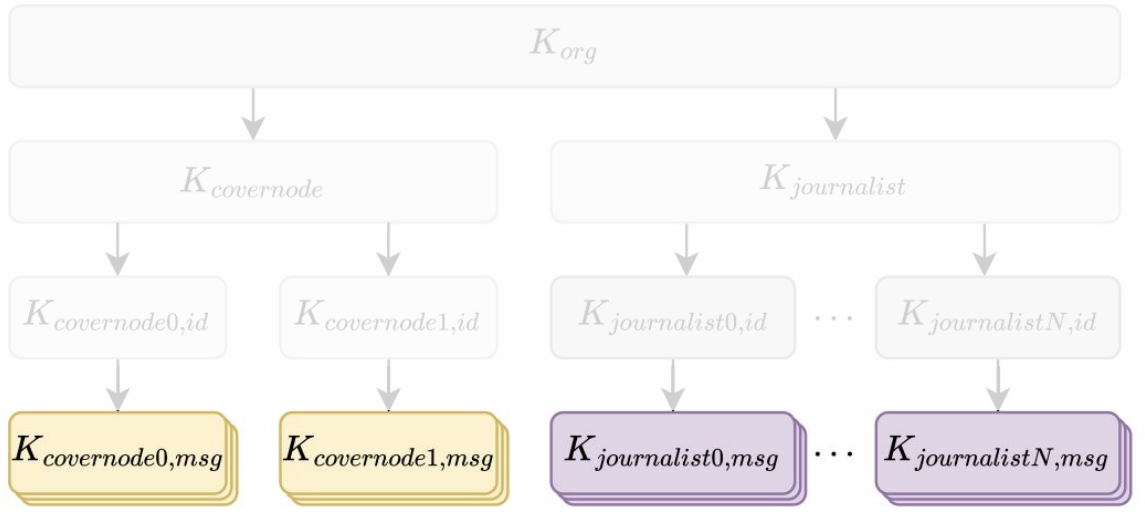
But what about the **cryptography!?**


K_{org}
$K_{covernode}$
$K_{journalist}$
$K_{covernode0,id}$
$K_{covernode1,id}$
$K_{journalist0,id}$
$\cdots$
$K_{journalistN,id}$
$K_{covernode0,msg}$
$K_{covernode1,msg}$
$K_{journalist0,msg}$
$\cdots$
$K_{journalistN,msg}$

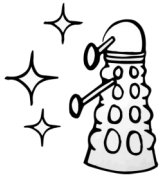

K_{org}
$K_{covernode}$
$K_{journalist}$
$K_{covernode0,id}$
$K_{covernode1,id}$
$K_{journalist0,id}$
$\cdots$
$K_{journalistN,id}$
$K_{covernode0,msg}$
$K_{covernode1,msg}$
$K_{journalist0,msg}$
$\cdots$
$K_{journalistN,msg}$


K_{org}
$K_{covernode}$
$K_{journalist}$
$K_{covernode0,id}$
$K_{covernode1,id}$
$K_{journalist0,id}$
$\cdots$
$K_{journalistN,id}$
$K_{covernode0,msg}$
$K_{covernode1,msg}$
$K_{journalist0,msg}$
$\cdots$
$K_{journalistN,msg}$


K_{org}
$K_{covernode}$
$K_{journalist}$
$K_{covernode0,id}$
$K_{covernode1,id}$
$K_{journalist0,id}$
$\cdots$
$K_{journalistN,id}$
$K_{covernode0,msg}$
$K_{covernode1,msg}$
$K_{journalist0,msg}$
$\cdots$
$K_{journalistN,msg}$

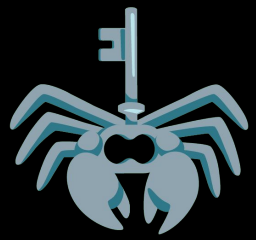

K_{org}
$K_{covernode}$
$K_{journalist}$
$K_{covernode0,id}$
$K_{covernode1,id}$
$K_{journalist0,id}$
$\cdots$
$K_{journalistN,id}$
$K_{covernode0,msg}$
$K_{covernode1,msg}$
$K_{journalist0,msg}$
$\cdots$
$K_{journalistN,msg}$

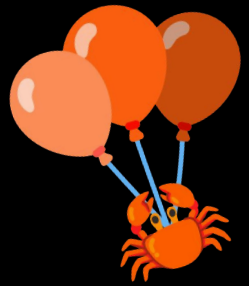
libsodium



curve25519-dalek

Roles and **Verification**





Type States!

Type state pattern

1. Operations on an object only exist when it is in the appropriate state.

2. States are encoded into the types. Attempts to use the operations in the wrong state fail to compile

3. Types have functions that allow them to transition to other states, adding and removing certain functionality

Type state pattern

1. Operations on an object only exist when it is in the appropriate state.

2. States are encoded into the types. Attempts to use the operations in the wrong state fail to compile

3. Types have functions that allow them to transition to other states, adding and removing certain functionality

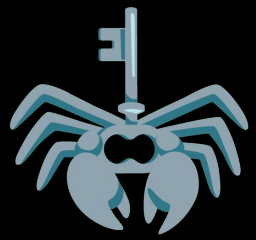
Type state pattern

1. Operations on an object only exist when it is in the appropriate state.

2. States are encoded into the types. Attempts to use the operations in the wrong state fail to compile

3. Types have functions that allow them to transition to other states, adding and removing certain functionality

Roles



They see me `Role`-ing

Different keys have different security properties

- Different capabilities
- Long lived vs short lived

We can use the type system to prevent accidentally using more powerful keys or just the wrong key.

```
define_role!(Organization);
define_role!(JournalistProvisioning);
define_role!(JournalistId);
define_role!(JournalistMessaging);
```

They see me `Role`-ing

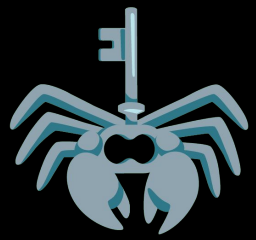
```
// covernode_provisioning_key_pair: SigningKeyPair<CoverNodeProvisioning>
create_journalist(name, /* ... */, covernode_provisioning_key_pair);
```

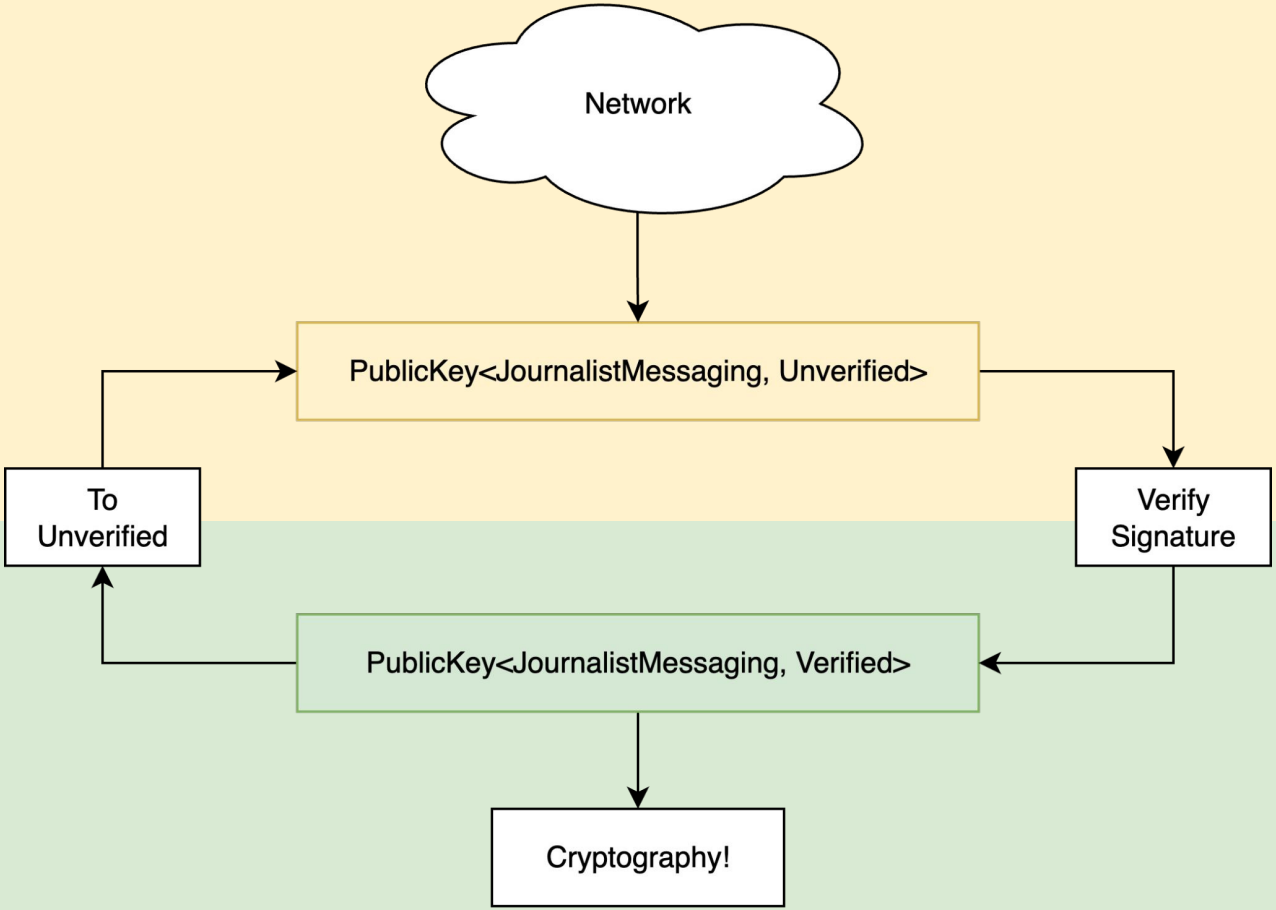
❌

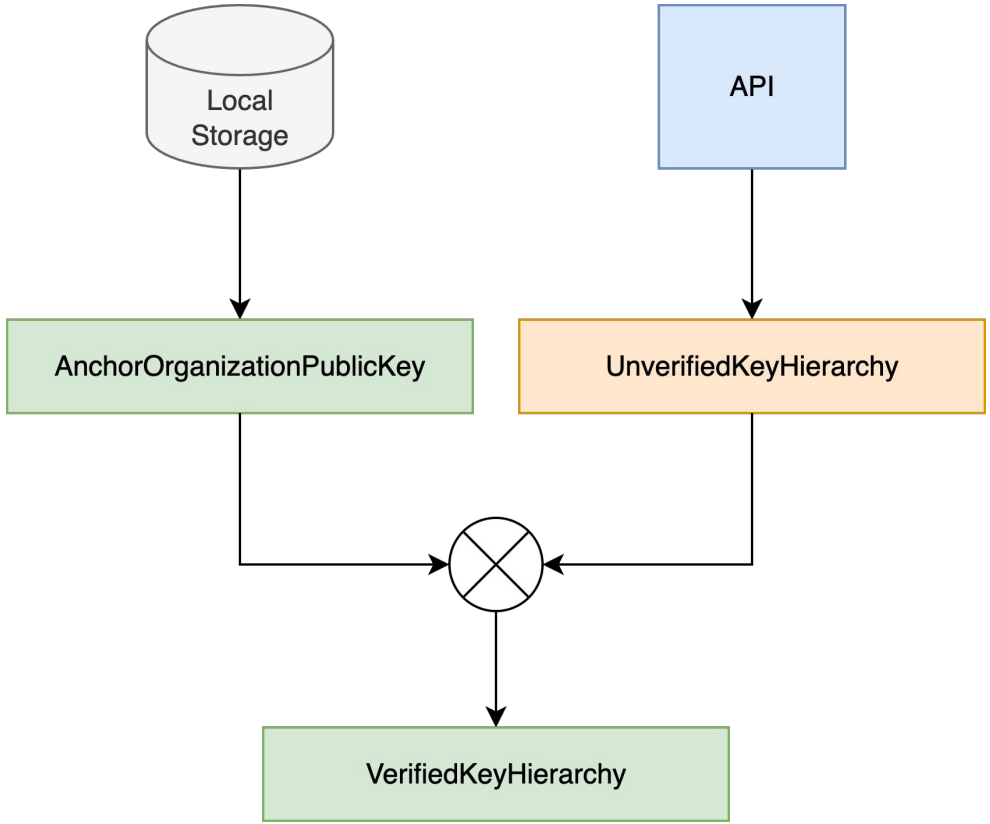
```
// journalist_provisioning_key_pair: SigningKeyPair<JournalistProvisioning>
create_journalist(name, /* ... */, journalist_provisioning_key_pair);
```

✅

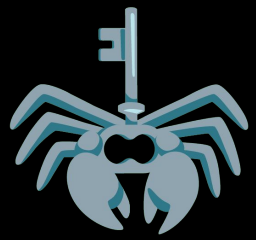
Verification




Network
PublicKey<JournalistMessaging, Unverified>
To Unverified
Verify Signature
PublicKey<JournalistMessaging, Verified>
Cryptography!


Local
Storage
API
AnchorOrganizationPublicKey
UnverifiedKeyHierarchy
VerifiedKeyHierarchy

Signed Forms



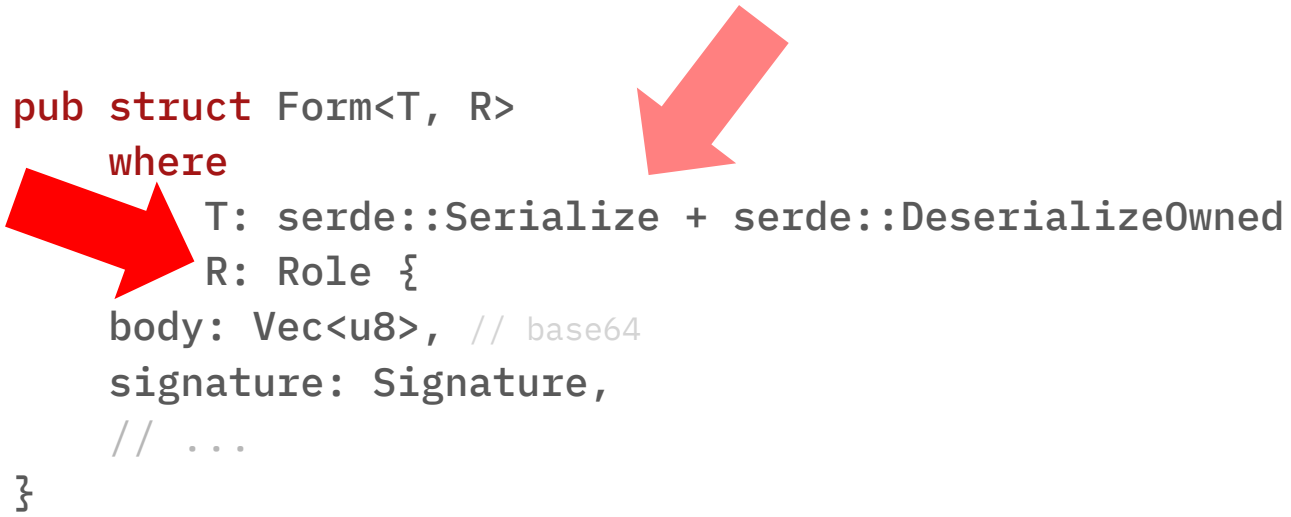
Signed & typed forms for API calls

```
pub struct Form<T, R>
    where
        T: serde::Serialize + serde::DeserializeOwned
        R: Role {
    body: Vec<u8>, // base64
    signature: Signature,
    // ...
}
```

Signed & typed forms for API calls

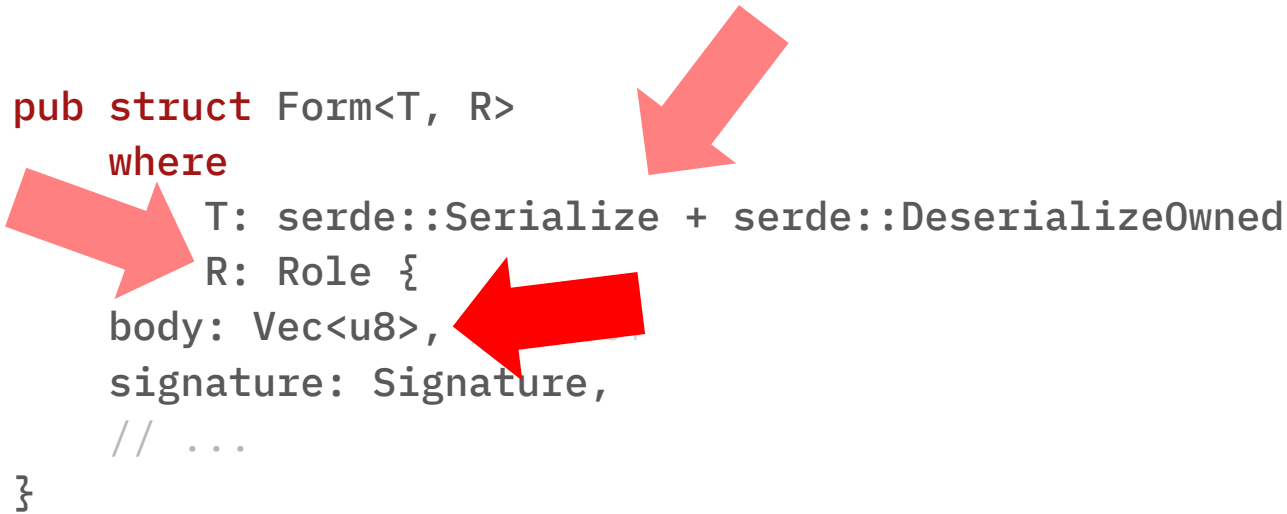
```
pub struct Form<T, R>
    where
        T: serde::Serialize + serde::DeserializeOwned
        R: Role {
    body: Vec<u8>, // base64
    signature: Signature,
    // ...
}
```

Signed & typed forms for API calls

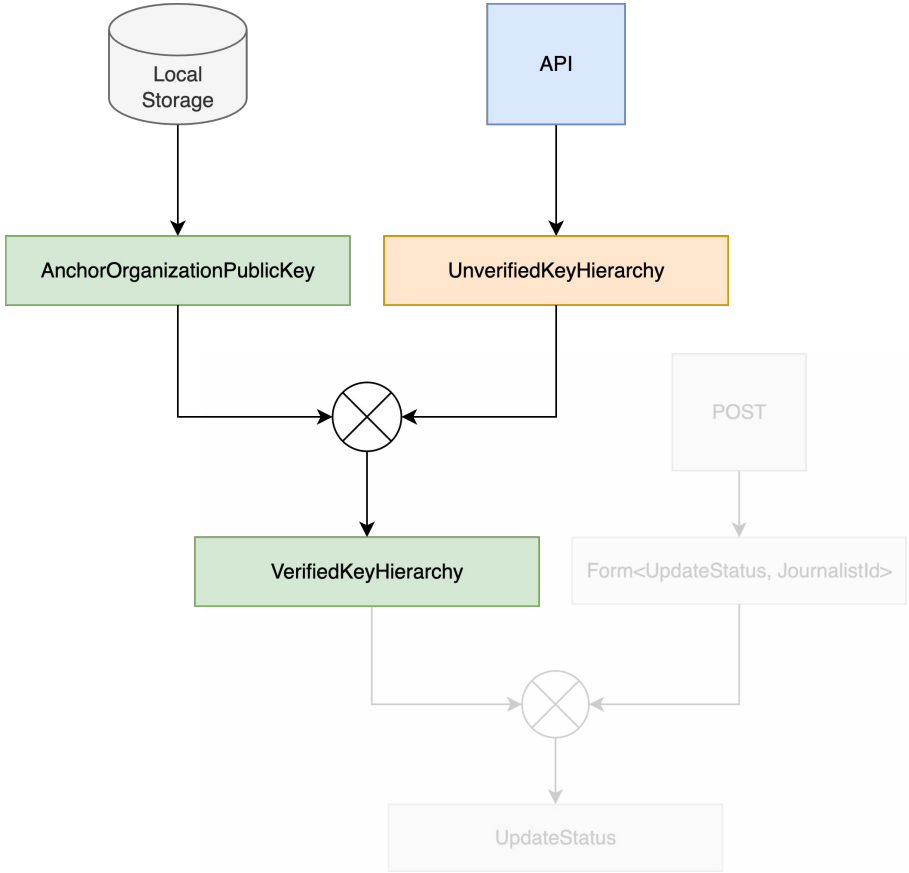


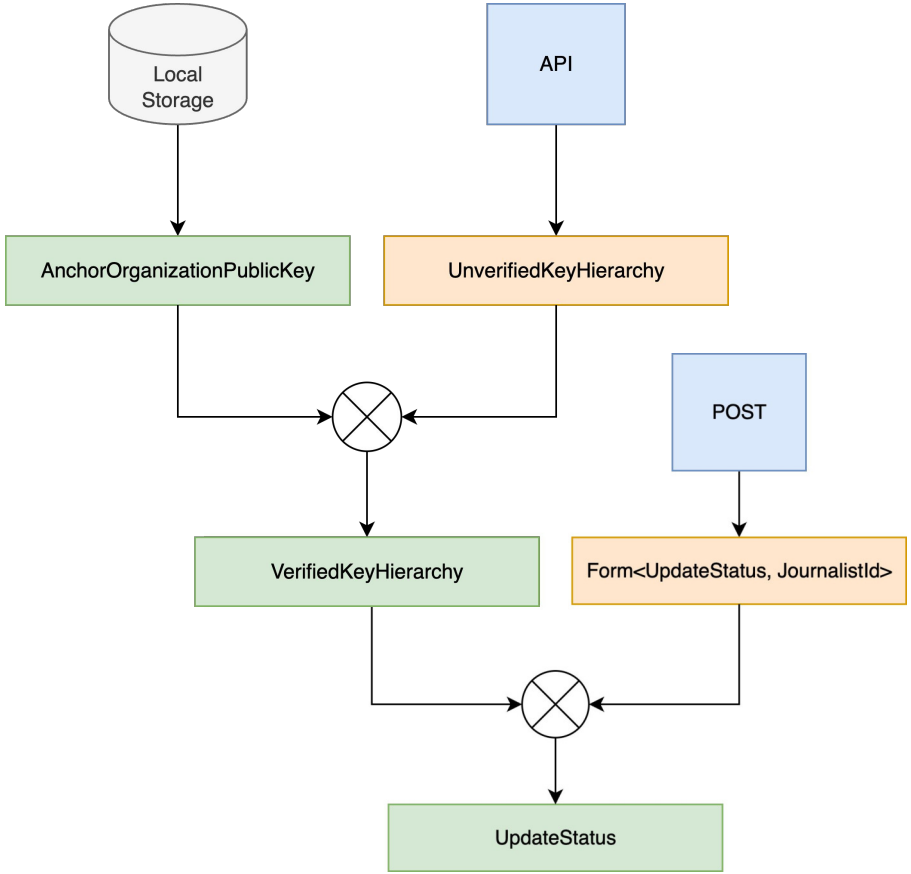
```
pub struct Form<T, R>
    where
        T: serde::Serialize + serde::DeserializeOwned
        R: Role {
    body: Vec<u8>, // base64
    signature: Signature,
    // ...
}
```

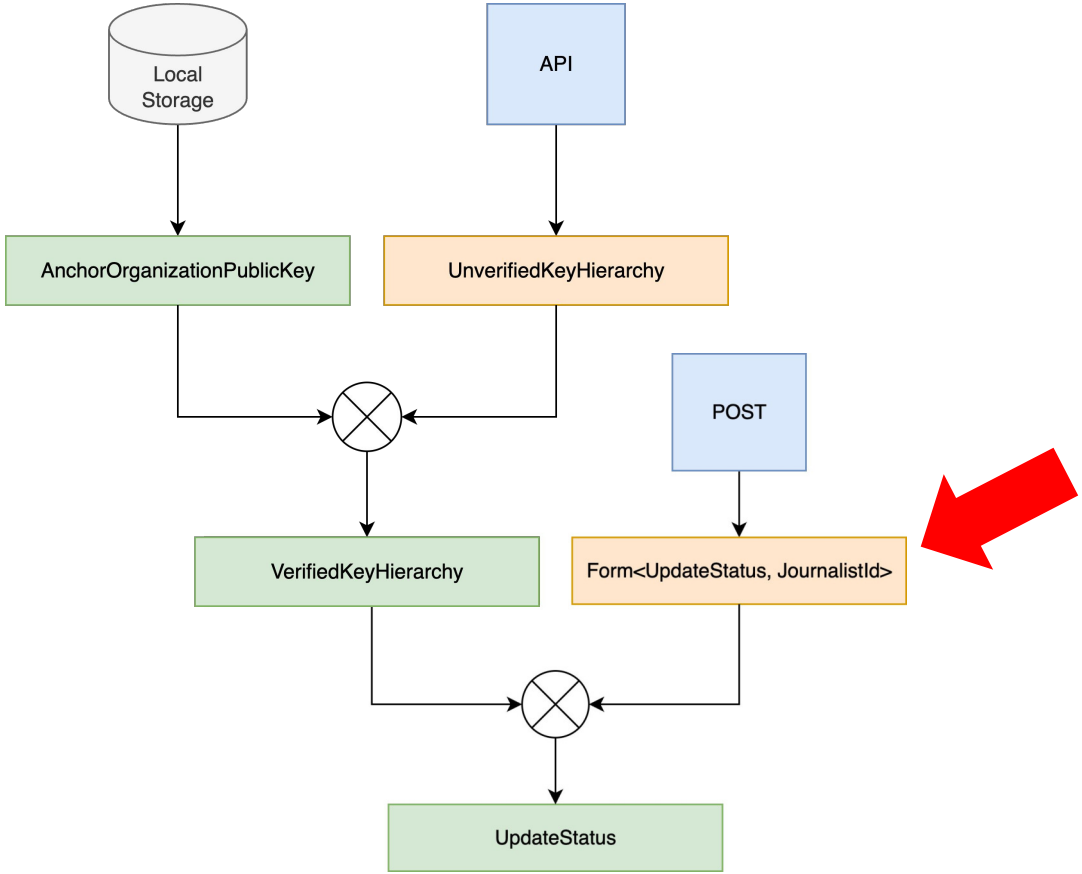
Signed & typed forms for API calls

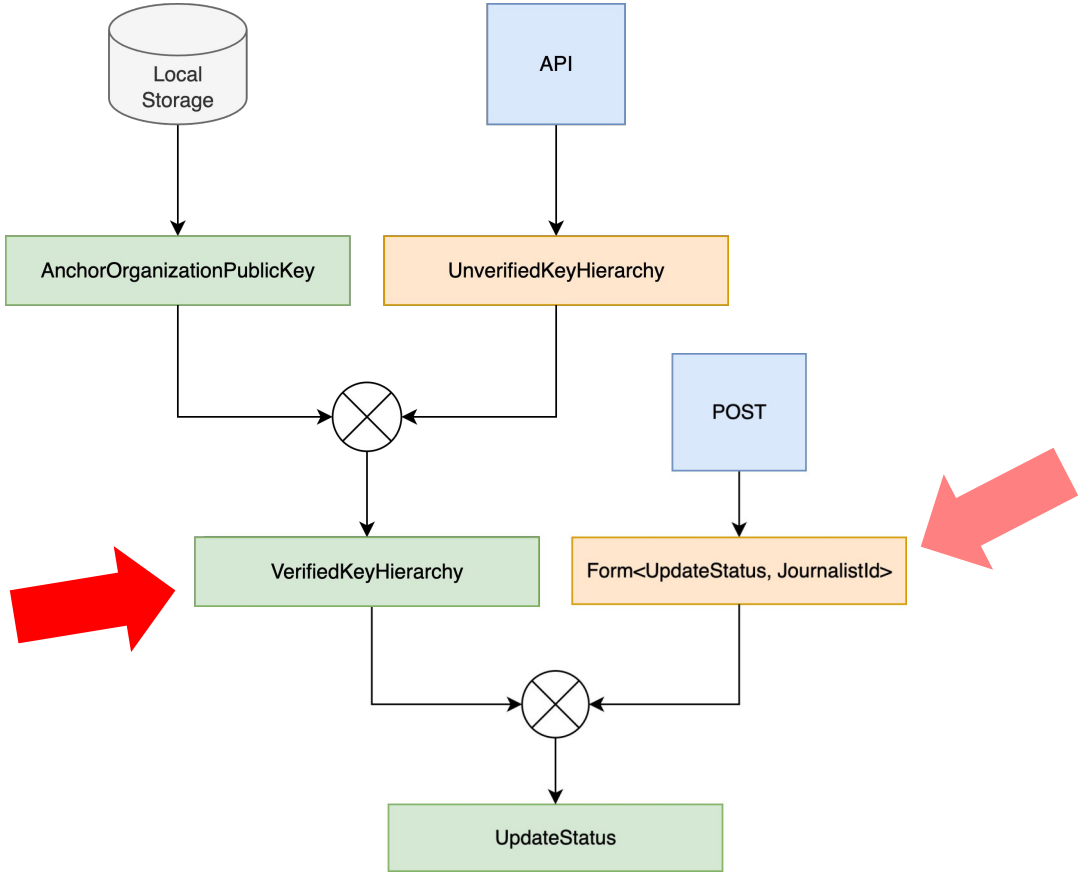


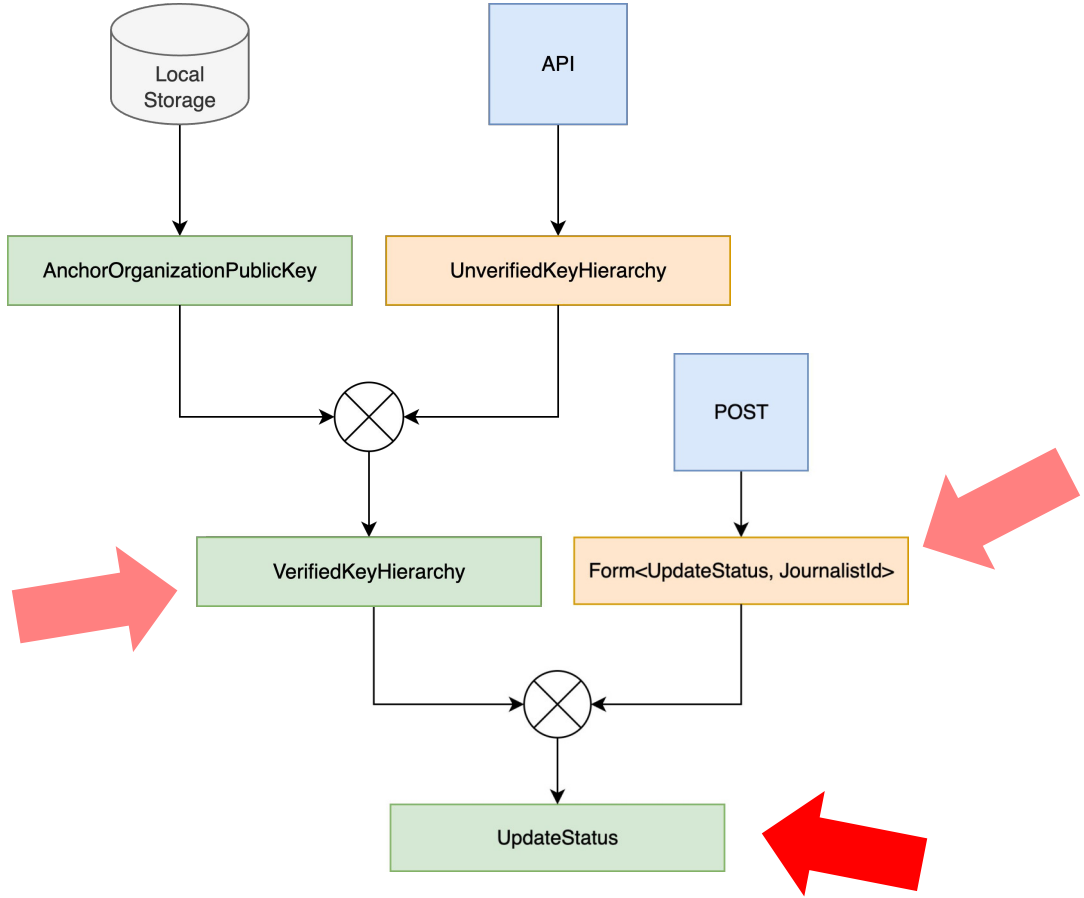
```
pub struct Form<T, R>
    where
        T: serde::Serialize + serde::DeserializeOwned
        R: Role {
    body: Vec<u8>,
    signature: Signature,
    // ...
}
```


Local Storage
API
AnchorOrganizationPublicKey
UnverifiedKeyHierarchy
POST
VerifiedKeyHierarchy
Form<UpdateStatus, JournalistId>
UpdateStatus


Local Storage
API
AnchorOrganizationPublicKey
UnverifiedKeyHierarchy
POST
VerifiedKeyHierarchy
Form<UpdateStatus, JournalistId>
UpdateStatus


Local Storage
API
AnchorOrganizationPublicKey
UnverifiedKeyHierarchy
POST
VerifiedKeyHierarchy
Form<UpdateStatus, JournalistId>
UpdateStatus


Local Storage
API
AnchorOrganizationPublicKey
UnverifiedKeyHierarchy
POST
VerifiedKeyHierarchy
Form<UpdateStatus, JournalistId>
UpdateStatus

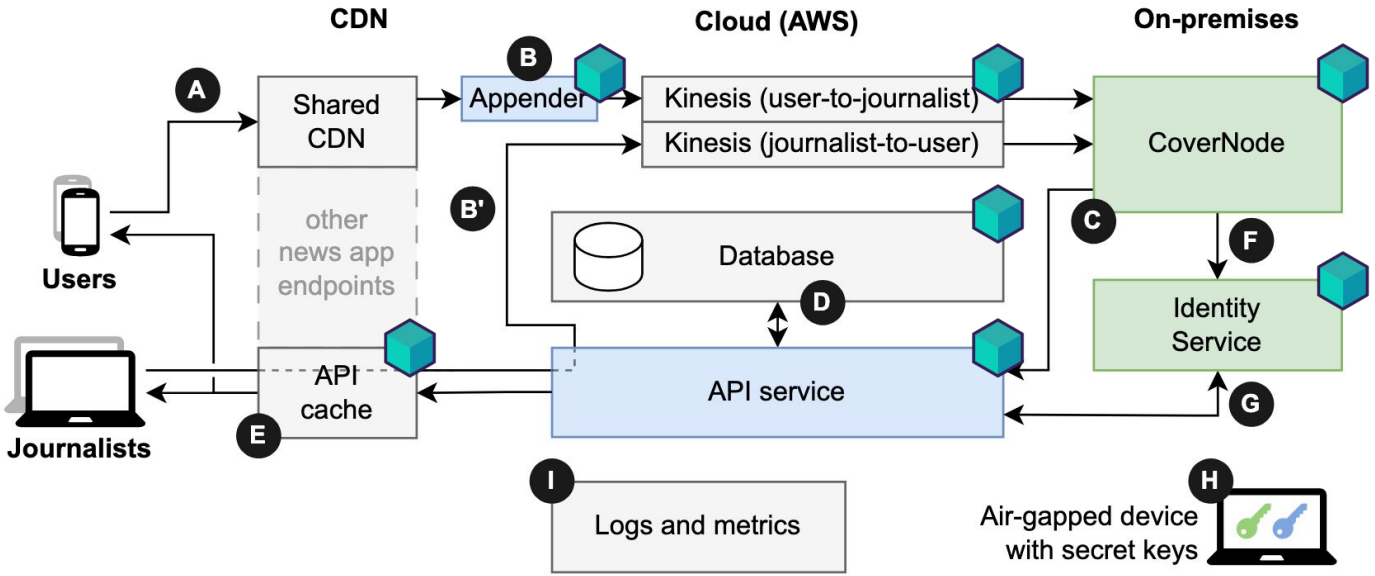

Local Storage
API
AnchorOrganizationPublicKey
UnverifiedKeyHierarchy
POST
VerifiedKeyHierarchy
Form<UpdateStatus, JournalistId>
UpdateStatus

Types give us **superpowers**!
But they are not the end of the story

Testing and Observability

Integration testing: test containers

Integration testing: test containers



Integration testing: time travel

common/src/time.rs

```rust
#[cfg(debug_assertions)]
pub fn now() -> DateTime<Utc> {
    read_fake_time_from_file()
}

#[cfg(not(debug_assertions))]
pub fn now() -> DateTime<Utc> {
    Utc::now()
}
```

Integration testing: time travel

integration-tests/journalist_key_rotations.rs

```
// ...
new_key = create_journalist_key_and_publish_to_api(stack.now());
keys = fetch_keys_from_api();
// assert new_key in api response

stack.time_travel(stack.now() + Duration::days(14))

keys = fetch_keys_from_api();
// assert new_key NOT in api response since it's expired!
```

Integration testing: test vectors

integration-tests/journalist_key_rotations.rs

```
// ...
new_key = create_journalist_key_and_publish_to_api(stack.now());
keys = fetch_keys_from_api();
// assert new_key in api response

save_test_vector!("journalist_key_created", &stack);

stack.time_travel(stack.now() + Duration::days(14))

keys = fetch_keys_from_api();
// assert new_key NOT in api response since it's expired!
```

Observability

METRICS
High-performance, protocol-agnostic instrumentation.

Amazon Cloudwatch

Grafana

Observability

covernode/src/from_user_polling_service.rs

```
metrics::counter!("U2JMessagesFromKinesis").increment(num_messages);
```

Observability

metrics_cloudwatch Public

api/src/main.rs

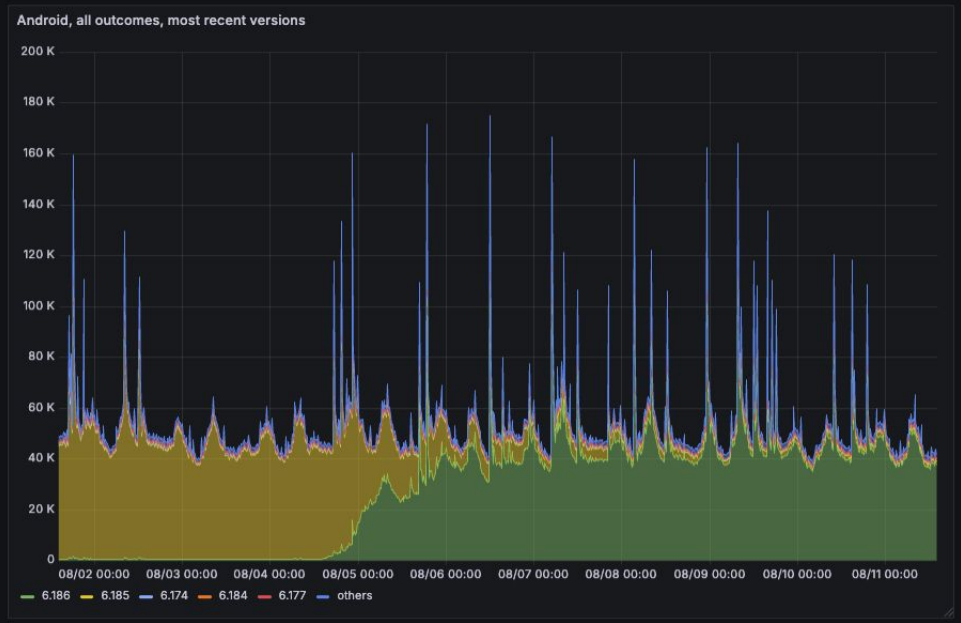
```rust
async fn main() -> anyhow::Result<()> {
    let config = aws_config::load_from_env().await;
    let cloudwatch_client = aws_sdk_cloudwatch::Client::new(&config);

    metrics_cloudwatch::Builder::new()
        .cloudwatch_namespace("API")
        .send_interval_secs(60)
        .init_thread(cloudwatch_client)

    // ...
}
```

Grafana

U2J Appender



Observability

∅ axum-metrics Public

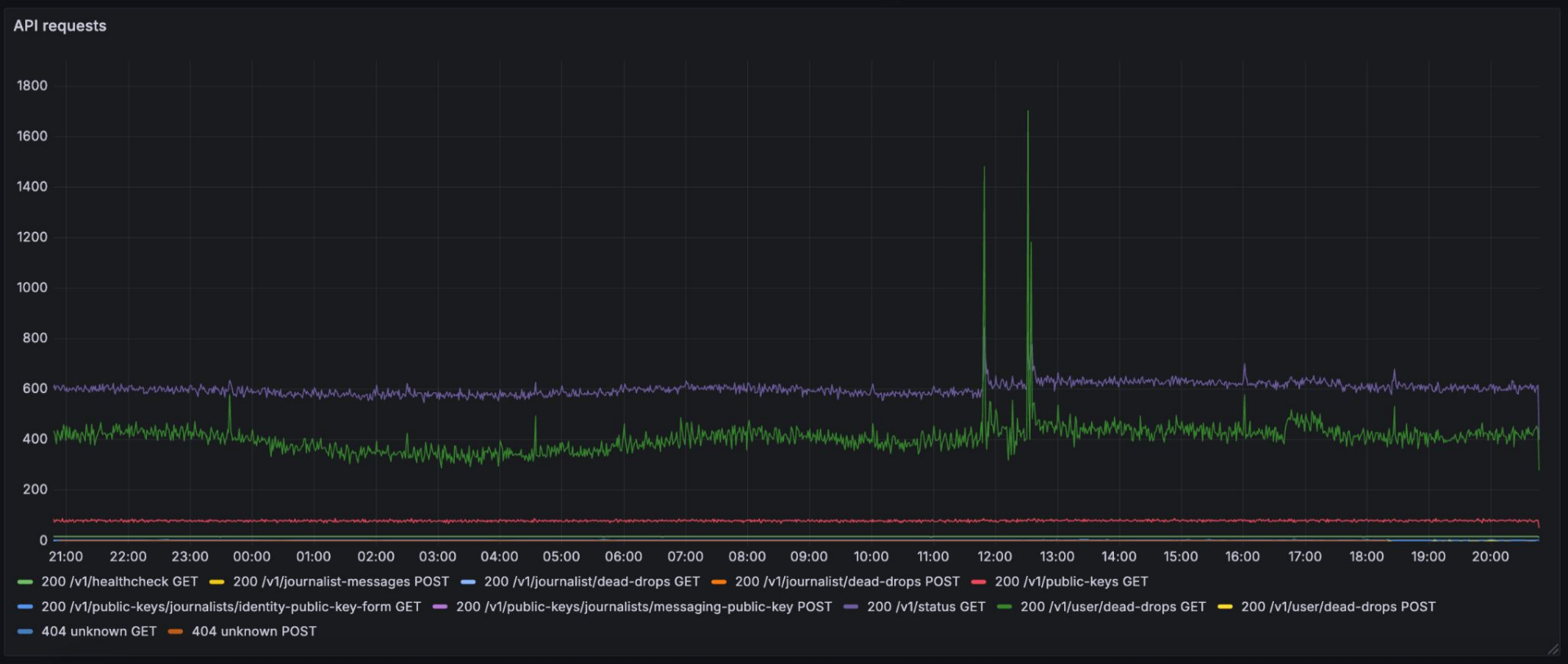
api/src/main.rs

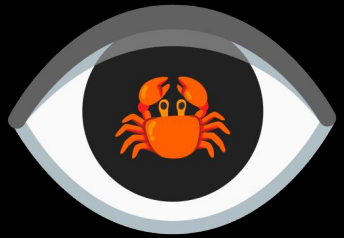
```rust
async fn main() -> anyhow::Result<()> {
    // ...
    let app = Router::new()
        .route("/public-keys", get(get_public_keys))
        // more routes ...
        .layer(axum_metrics::MetricLayer::default());

    axum::serve(listener, app).await?;

    // ...
}
```

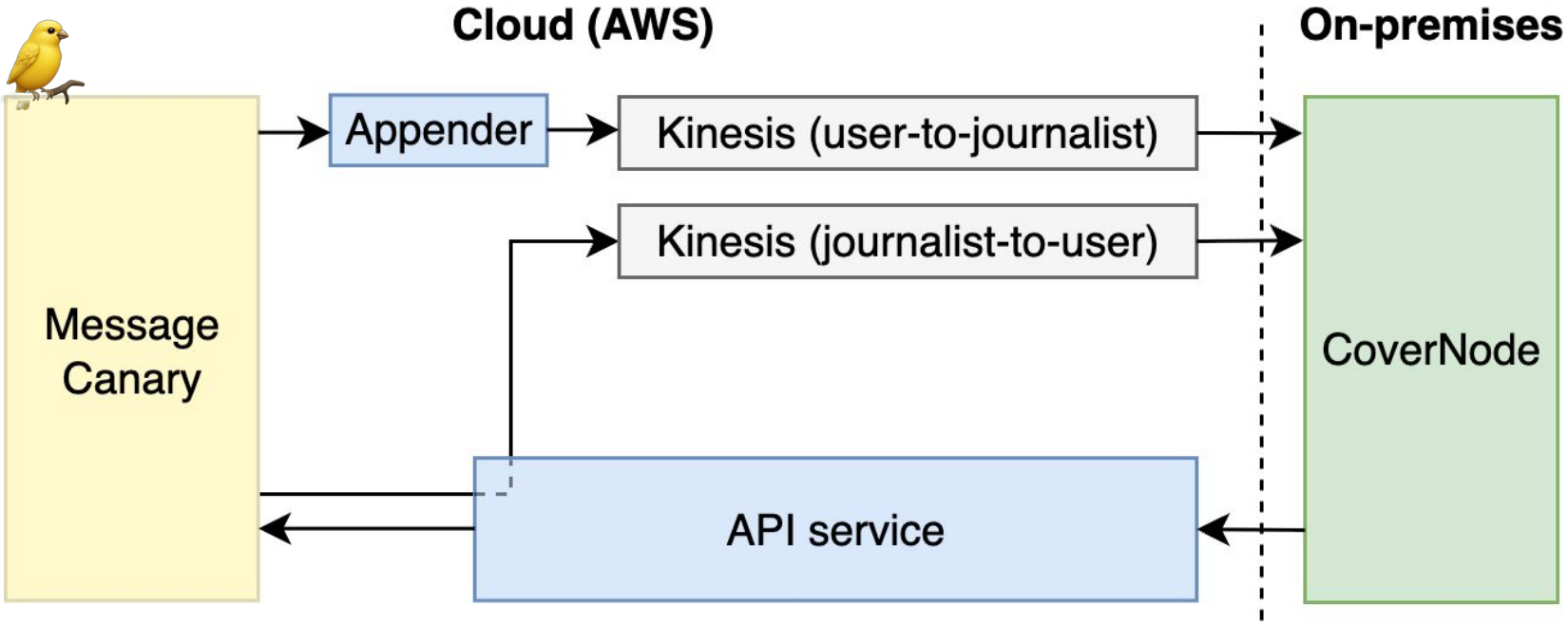
Grafana


API requests
1800
1600
1400
1200
1000
800
600
400
200
0
21:00
22:00
23:00
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
200 /v1/healthcheck GET
200 /v1/journalist-messages POST
200 /v1/journalist/dead-drops GET
200 /v1/journalist/dead-drops POST
200 /v1/public-keys GET
200 /v1/public-keys/journalists/identity-public-key-form GET
200 /v1/public-keys/journalists/messaging-public-key POST
200 /v1/status GET
200 /v1/user/dead-drops GET
200 /v1/user/dead-drops POST
404 unknown GET
404 unknown POST



Monitoring

an opaque system


Cloud (AWS)
On-premises
Message Canary
Appender
Kinesis (user-to-journalist)
Kinesis (journalist-to-user)
CoverNode
API service

Grafana

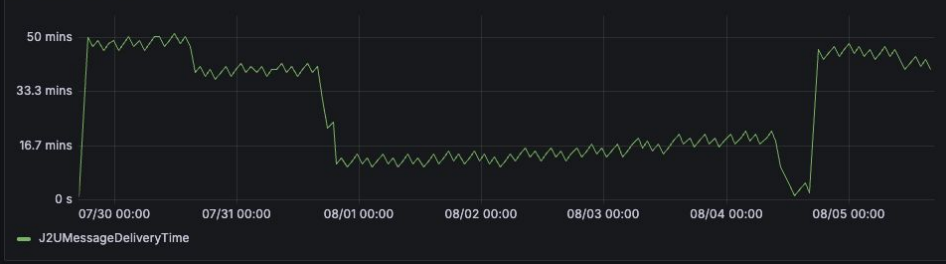
Message Canary

Undelivered U2J Messages

Undelivered J2U Messages

U2J delivery time

J2U delivery time



Our use of Rust

Type system

Compile time-guarantees; roles and key verification checks

Mature ecosystem

Web services, metrics, tracing, error handling, testing, …

Cargo

Trusted partner; great integration with ecosystem

SecureMessaging is live!

Handling 5,000,000+ cover messages per day

0 to 100 roll-out over a few weeks

Academic partnership

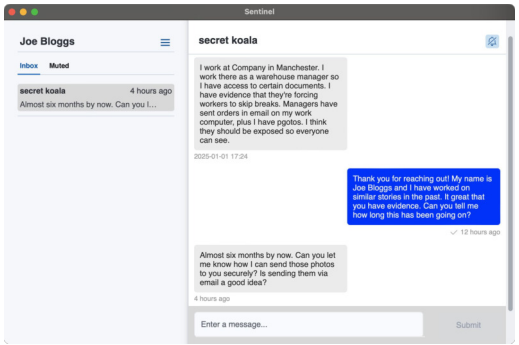
Published white paper and incorporated learned lessons in curriculum (P79).

Available as open-source

Check-out our repository on GitHub: `github.com/guardian/coverdrop`

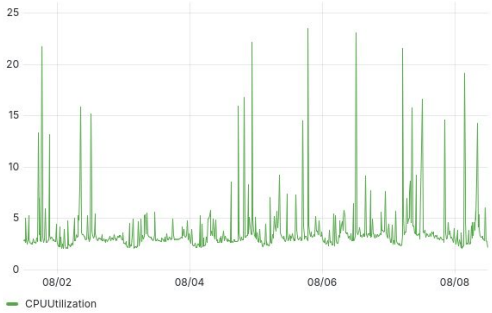
What we couldn’t cover today…

Journalist client



We use **Tauri** for the UI which allows us to reuse a lot of the `common` shared codebase.

Performance wins



The **main ingress endpoint** runs at <10% CPU on a small AWS instance. Migrated from a webassembly CDN function.

The actual protocol

There is a lot of interesting protocol and cryptography work that covered in the **white paper**.

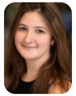
Thank you!

Learn more: coverdrop.org

The original CoverDrop research team


Mansoor Ahmed

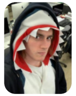

Daniel Hugenroth


Diana A. Vasile


Alastair R. Beresford


Ross Anderson

The Guardian crew and contributors


Sam Cutler

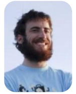

Luke Hoyland

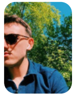

Philip McMahon


Zeke Hunter-Green


Sabina Bejasa-Dimmock


Marjan Kalanaki


Dom Kendrick


Mario Savarese

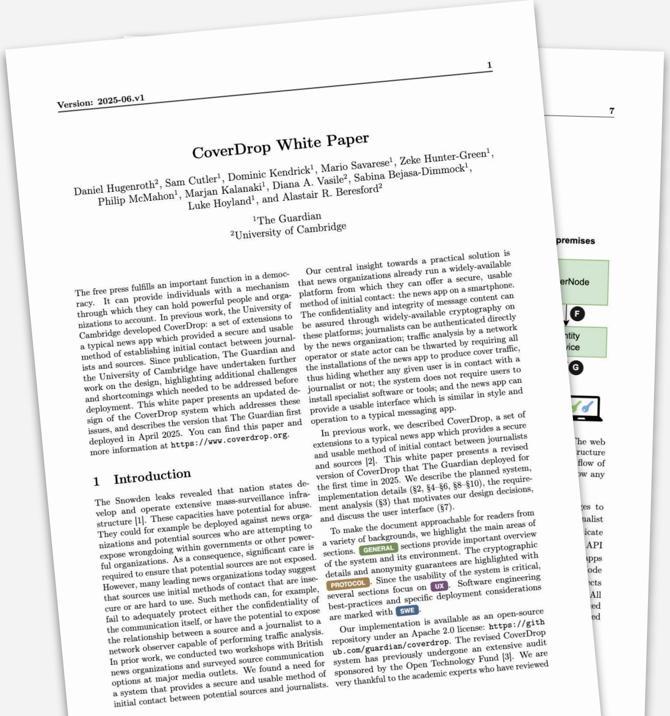

Chloe Kirton

UNIVERSITY OF CAMBRIDGE Department of Computer Science and Technology

The Guardian

NOKIA BELL LABS

OPEN TECHNOLOGY FUND


Version: 2025-06.v1 1

CoverDrop White Paper

Daniel Hugenroth[2], Sam Cutler[1], Dominic Kendrick[1], Mario Savarese[1], Zeke Hunter-Green[1], Philip McMahon[1], Marjan Kalanaki[1], Diana A. Vasile[2], Sabina Bejasa-Dimmock[1], Luke Hoyland[1], and Alastair R. Beresford[2]

[1]The Guardian
[2]University of Cambridge

The free press fulfills an important function in a democracy. It can provide individuals with a mechanism through which they can hold powerful people and organizations to account. In previous work, the University of Cambridge developed CoverDrop: a set of extensions to a typical news app which provided a secure and usable method of establishing initial contact between journalists and sources. Since publication, The Guardian and the University of Cambridge have undertaken further work on the design, highlighting additional challenges and shortcomings which needed to be addressed before deployment. This white paper presents an updated design of the CoverDrop system which addresses these issues, and describes the version that The Guardian first deployed in April 2025. You can find this paper and more information at `https://www.coverdrop.org`.

1 Introduction

The Snowden leaks revealed that nation states develop and operate extensive mass-surveillance infrastructure [1]. These capacities have potential for abuse. They could for example be deployed against news organizations and potential sources who are attempting to expose wrongdoing within governments or other powerful organizations. As a consequence, significant care is required to ensure that potential sources are not exposed. However, many leading news organizations today suggest that sources use initial methods of contact that are insecure or are hard to use. Such methods can, for example, fail to adequately protect either the confidentiality of the communication itself, or have the potential to expose the relationship between a source and a journalist to a network observer capable of performing traffic analysis. In prior work, we conducted two workshops with British news organizations and surveyed source communication options at major media outlets. We found a need for a system that provides a secure and usable method of initial contact between potential sources and journalists.

Our central insight towards a practical solution is that news organizations already run a widely-available platform from which they can offer a secure, usable method of initial contact: the news app on a smartphone. The confidentiality and integrity of message content can be assured through widely-available cryptography on these platforms; journalists can be authenticated directly by the news organization; traffic analysis by a network operator or state actor can be thwarted by requiring all the installations of the news app to produce cover traffic, thus hiding whether any given user is in contact with a journalist or not; the system does not require users to install specialist software or tools; and the news app can provide a usable interface which is similar in style and operation to a typical messaging app.

In previous work, we described CoverDrop, a set of extensions to a typical news app which provides a secure and usable method of initial contact between journalists and sources [2]. This white paper presents a revised version of CoverDrop that The Guardian deployed for the first time in 2025. We describe the planned system, implementation details (§2, §4–§6, §8–§10), the requirement analysis (§3) that motivates our design decisions, and discuss the user interface (§7).

To make the document approachable for readers from a variety of backgrounds, we highlight the main areas of sections. GENERAL sections provide important overview of the system and its environment. The cryptographic details and anonymity guarantees are highlighted with PROTOCOL. Since the usability of the system is critical, several sections focus on UX. Software engineering best-practices and specific deployment considerations are marked with SWE.

Our implementation is available as an open-source repository under an Apache 2.0 license: `https://github.com/guardian/coverdrop`. The revised CoverDrop system has previously undergone an extensive audit sponsored by the Open Technology Fund [3]. We are very thankful to the academic experts who have reviewed

fin.