Keeping your data safe from prying eyes

Sam Barker
Principal Software Engineer @ Red Hat

Why are we worried about it? GDPR

"For especially severe violations, ..., the fine framework can be up to **20 million euros, or ..., up to 4 %** of their total global turnover of the preceding fiscal year**, whichever is higher.**[1]

[1] https://gdpr-info.eu/issues/fines-penalties/

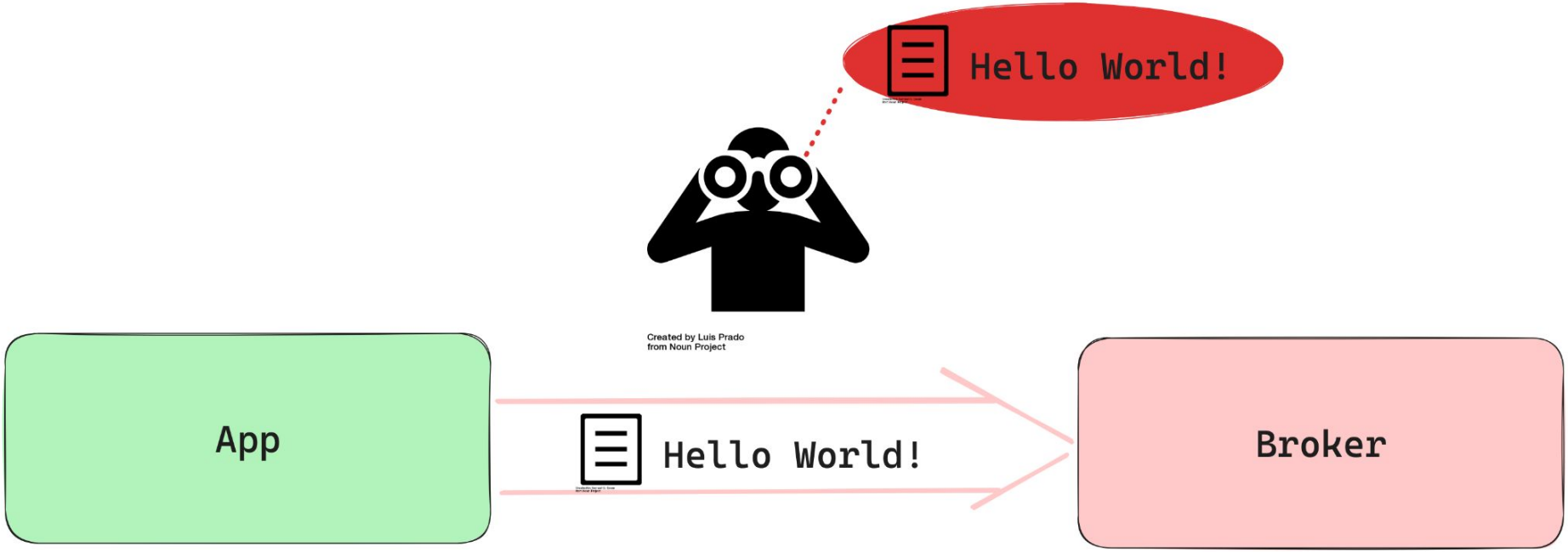
Why are we worried about it? SEC

“personal liability for CISOs. Cases like Uber, where CSO Joe Sullivan faced personal charges after a data breach, and SolarWinds, where CISO Timothy Brown encountered financial penalties, serve as stark cautionary tales. “[3]

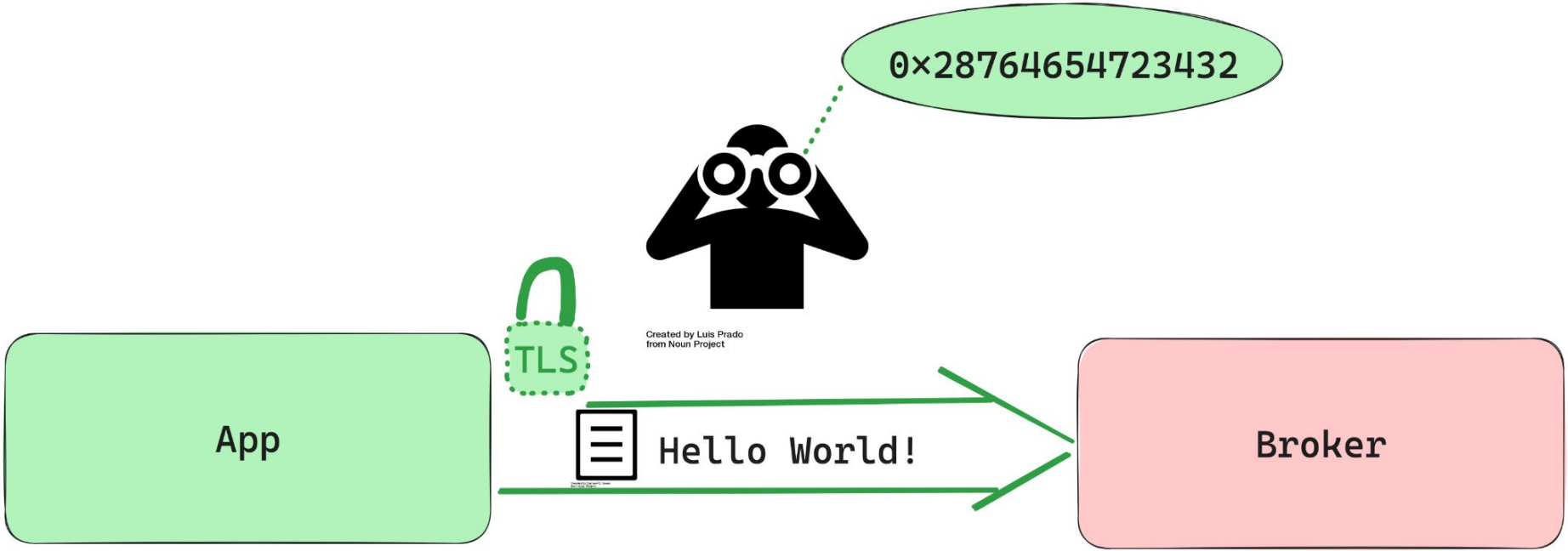
[3] https://www.safe.security/resources/blog/ciso-personal-liability/

What exactly are we worried about here?

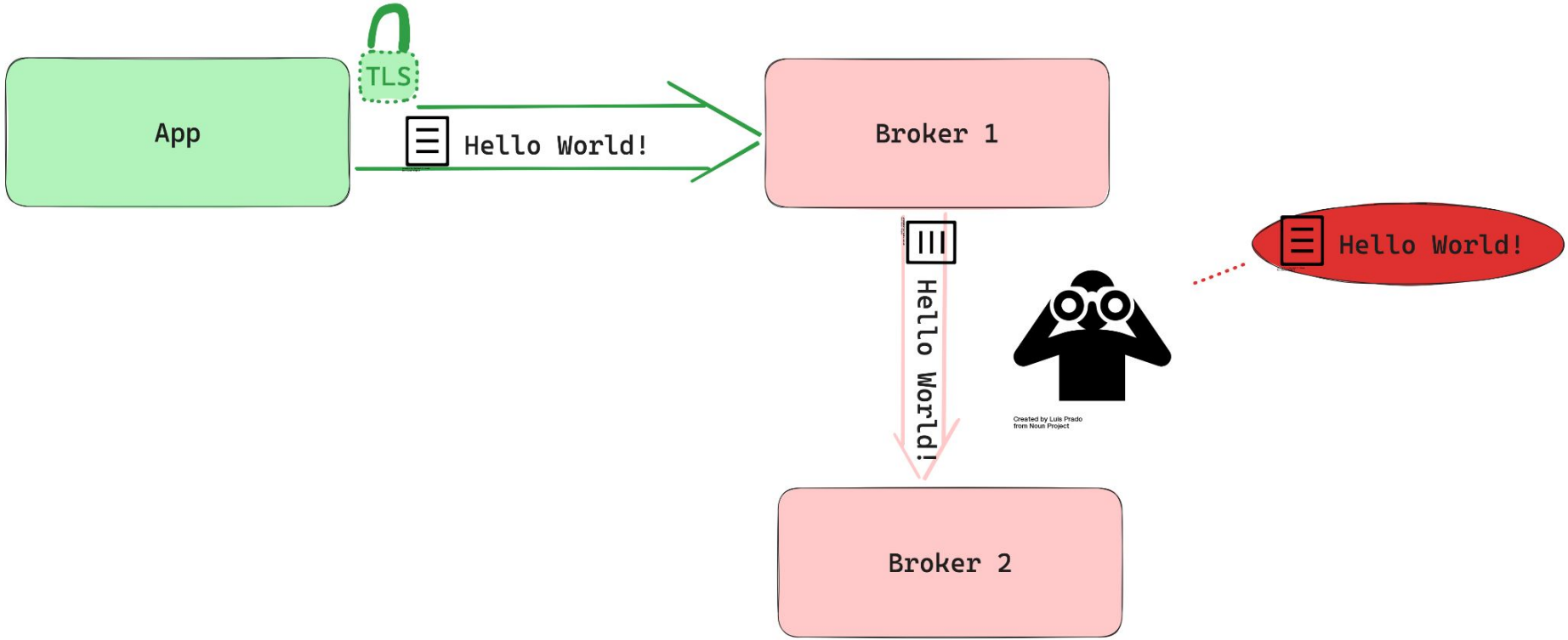
Intercepting client traffic



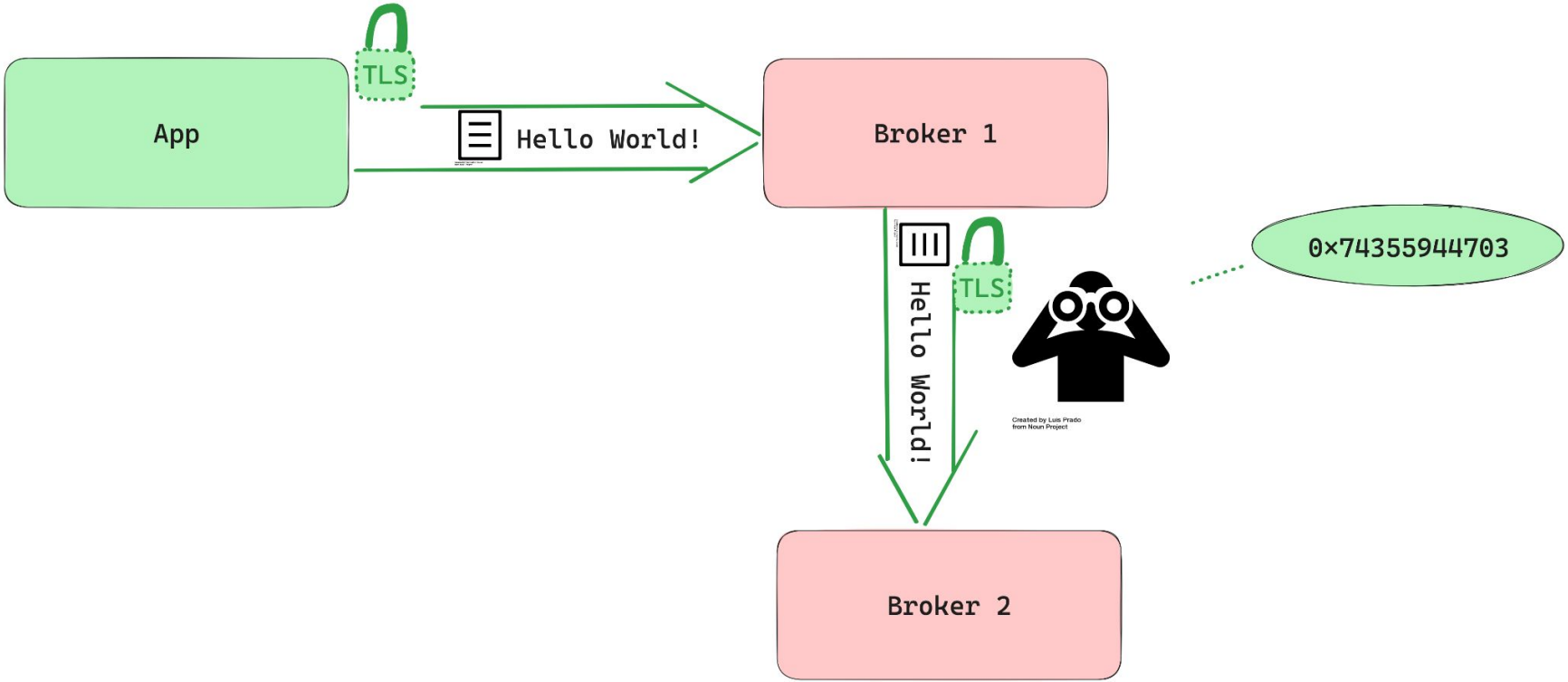
TLS to the rescue



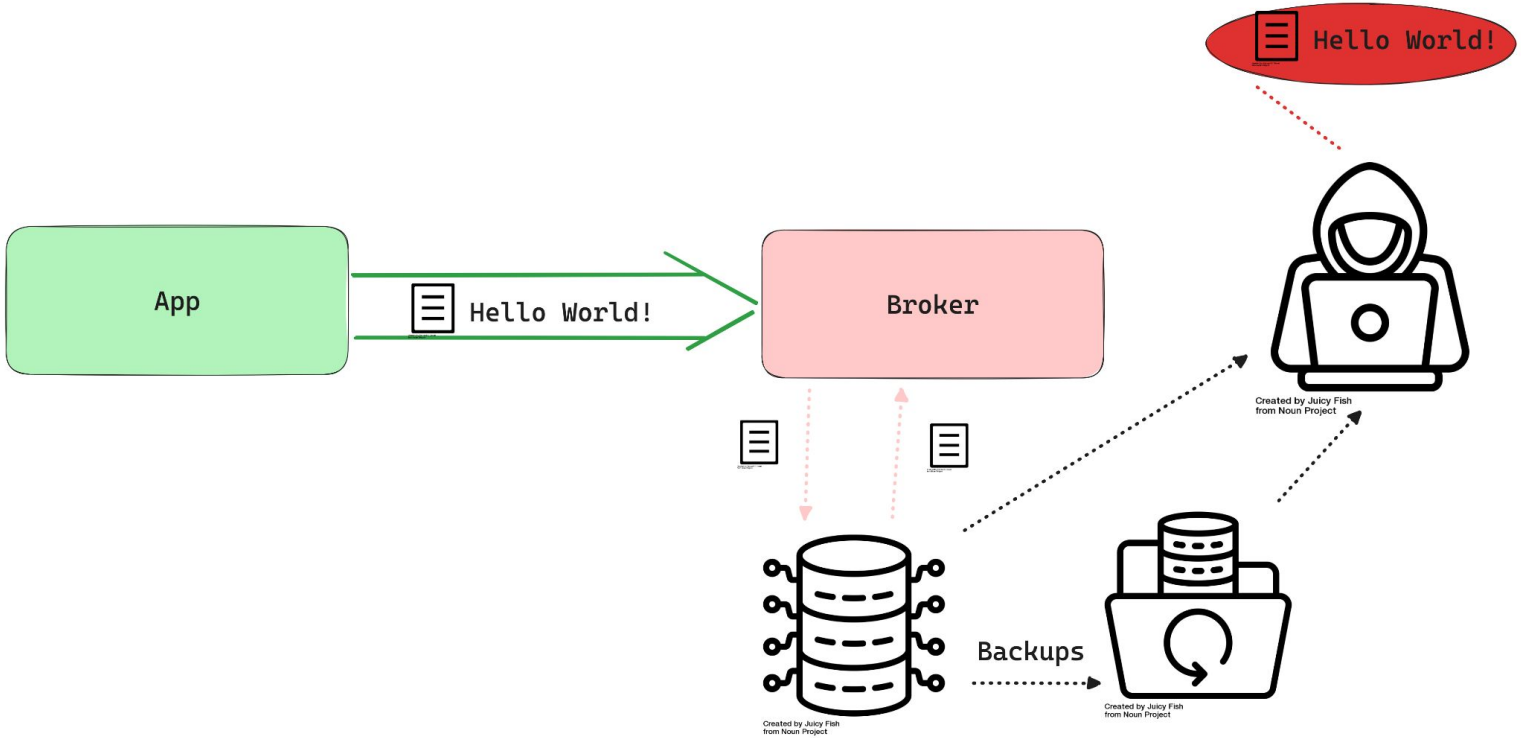
Intercepting inter broker traffic



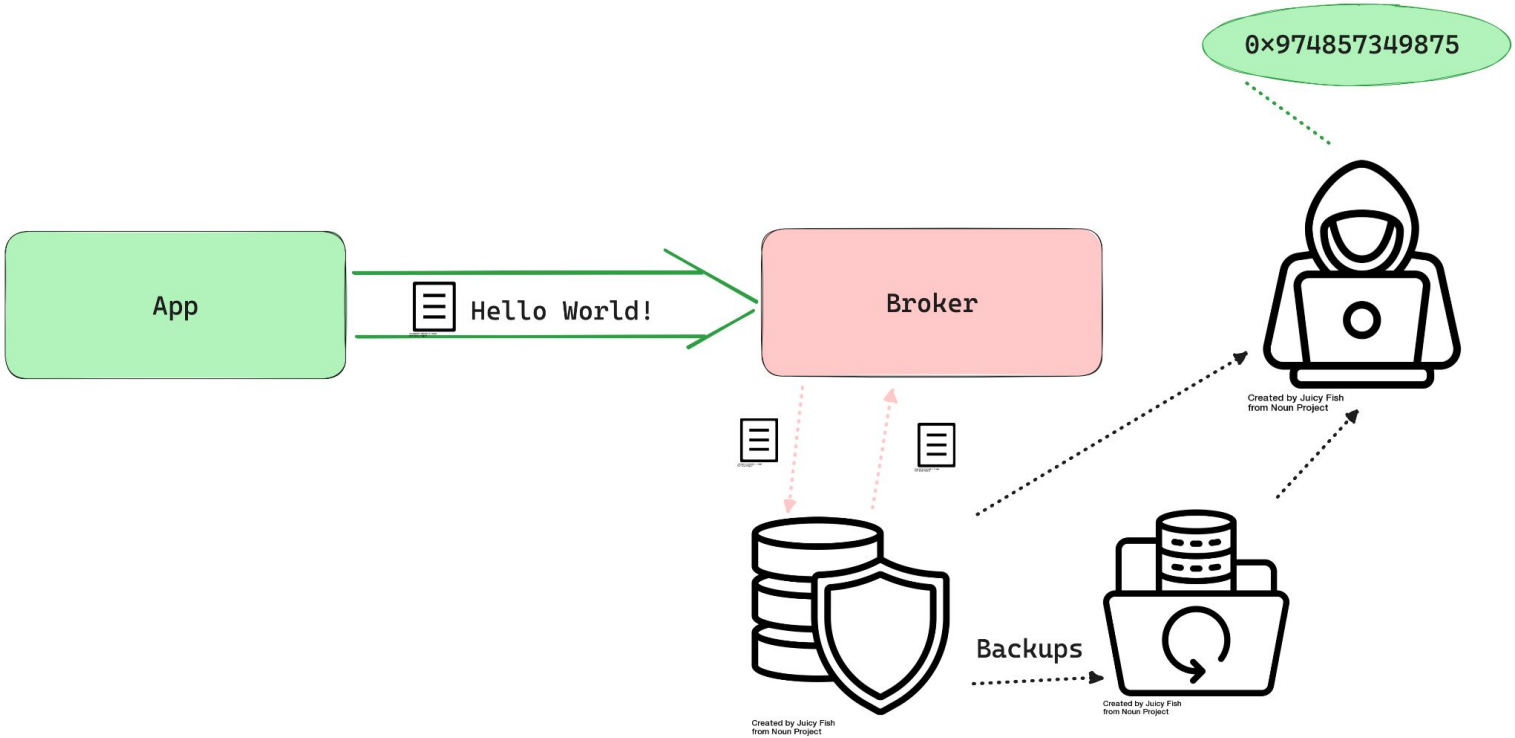
TLS saves the day again



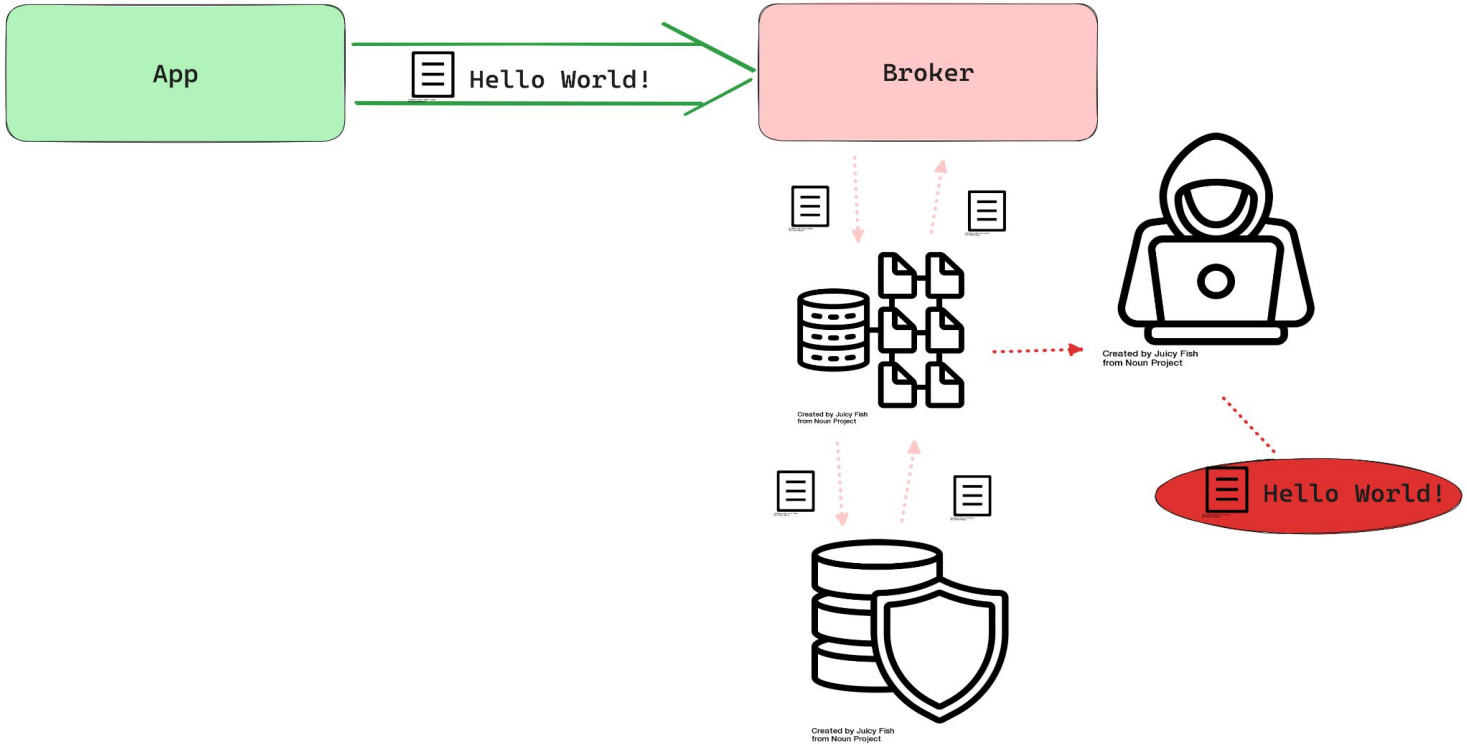
Stolen disks/backups



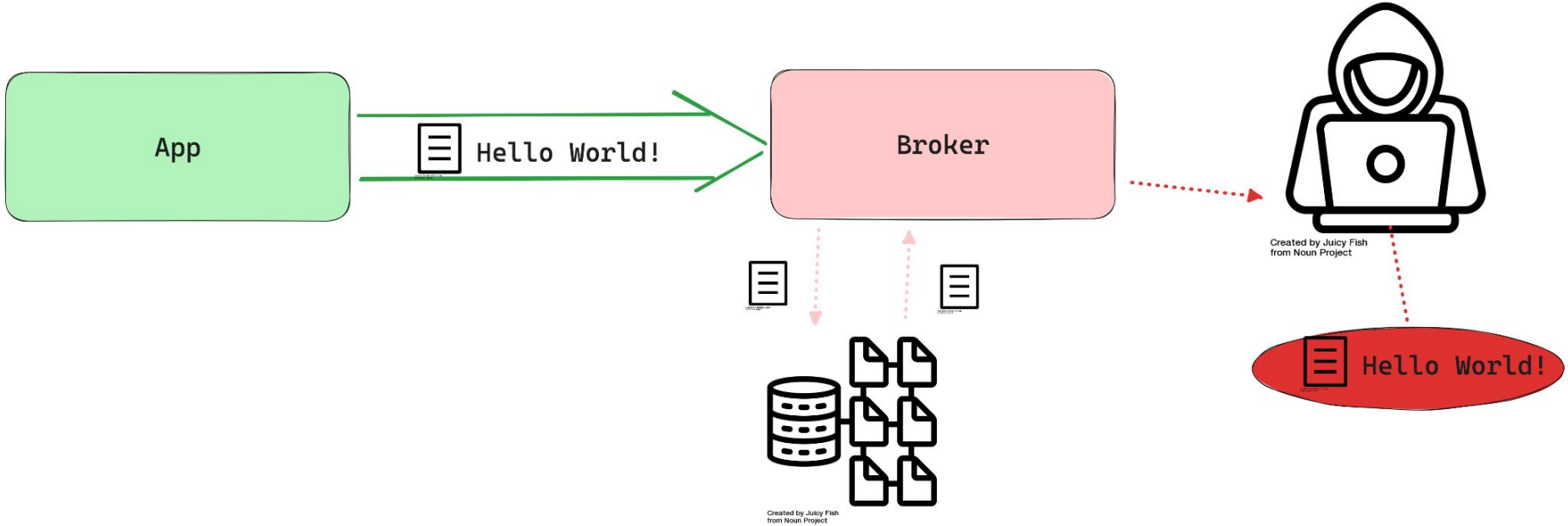
Disk encryption



File system access??



Heap Dumps??


App
Hello World!
Broker
Created by Juicy Fish
from Noun Project
Hello World!
Created by Juicy Fish
from Noun Project

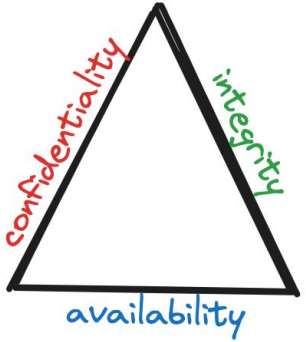
A threat model

Existing tools are good!

But can't do EVERYTHING!

We are specifically thinking about

- Nosy systems administrators
- Passive snooping
- Insider Threats



In CIA Triad terms we are:

- Providing confidentiality
- Improving Integrity
- Leaving availability unchanged.

How do we defend against this?

Fix it in the clients?

Advantages:

- ✓ Already available!
- ✓ Ensures brokers never see plain text

Challenges:

- ✗ Not all clients support the same levels of sophistication and robustness
- ✗ Every team needs to implement encryption separately and correctly!
- ✗ Wide distribution of key material
- ✗ Can’t address third party applications

BUT that means every client

Fix it in the Broker?

Advantages:

- ✓ Do it once, do it right!
- ✓ Limited distribution of key material
- ✓ Limited set of languages and crypto libraries to worry about

Challenges:

- ✗ We don't have the hooks
- ✗ Scaling CPU without scaling I/O
- ✗ Do we trust our broker host with our sensitive material?
- ✗ Hosted brokers? We can't add code there

Fix it in a proxy?

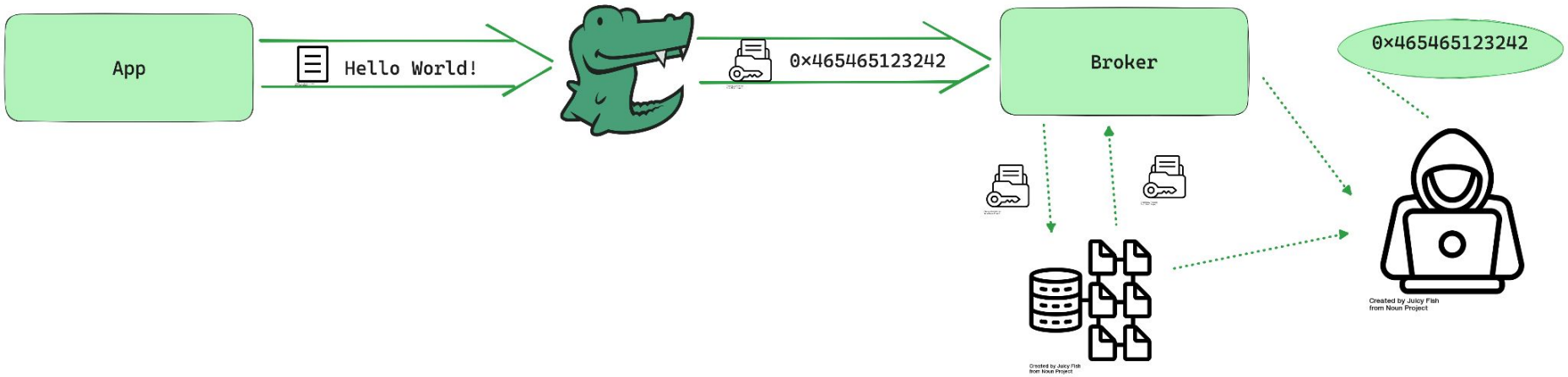
Advantages:

- ✓ Ensures brokers never see plain text
- ✓ Do it once, do it right!
- ✓ Limited distribution of key material
- ✓ Limited set of languages and crypto libraries to worry about
- ✓ Centralised configuration
- ✓ Transparent to all clients

Challenges:

- ✗ Another tier to run, maintain and manage.
- ✗ Introduces additional network hops
- ✗ Using TLS certificates for Authentication is problematic

Encrypt it in the middle: A proxy!



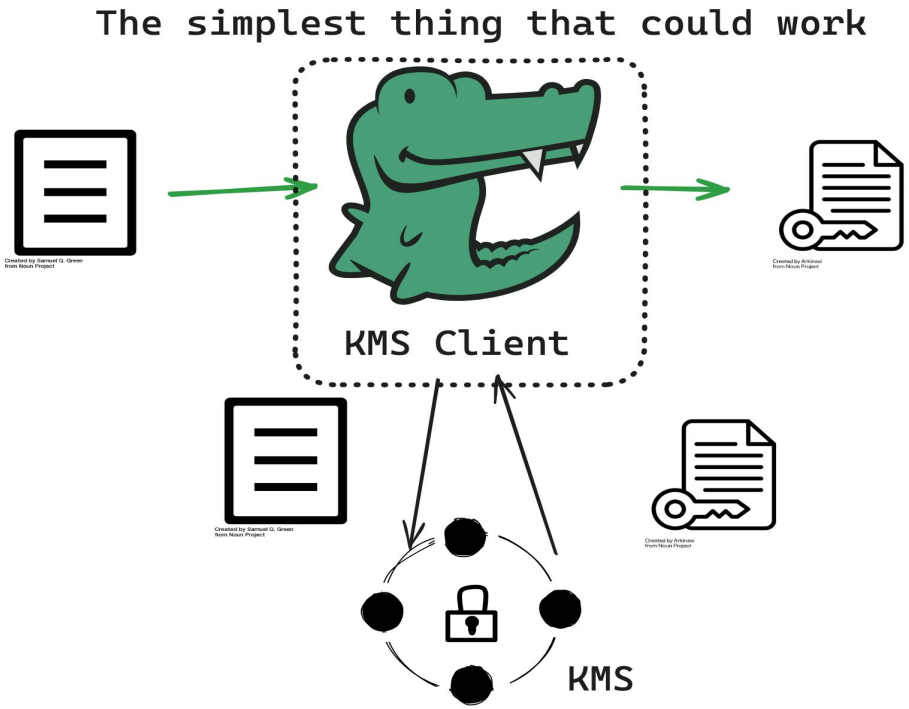
The details: Envelope Encryption

Kafka Records

```
{
   "timestamp": 123654,
   "offset": null,
   "key": {"accountId": "sam-216546546"},
   "value": {"action": "credit", "amount": 1000.0},
   "headers": [
     {"key": "h1", "value": "v1"},
     {"key": "h2", "value": "v2"},
     {"key": "hN", "value": "vN"}
   ]
 }
```

- We can't encrypt
  - ✗ Timestamps
  - ✗ Offsets
- We CAN encrypt:
  - ✓ Everything else 🎊

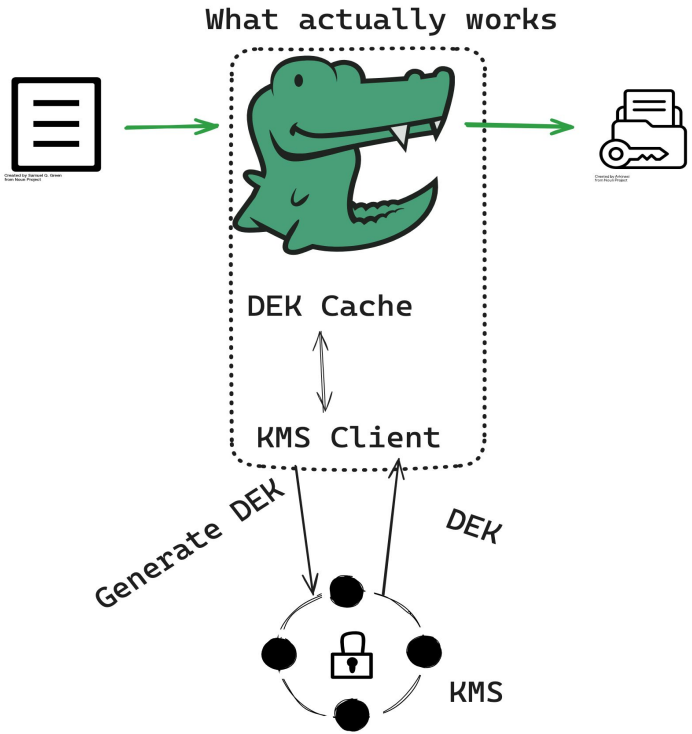
Encrypt Everything!!



- Use a Key Management System (KMS) to encrypt records
- KMS has to keep the Key **FOREVER***
- Keys have a finite cryptographic capacity

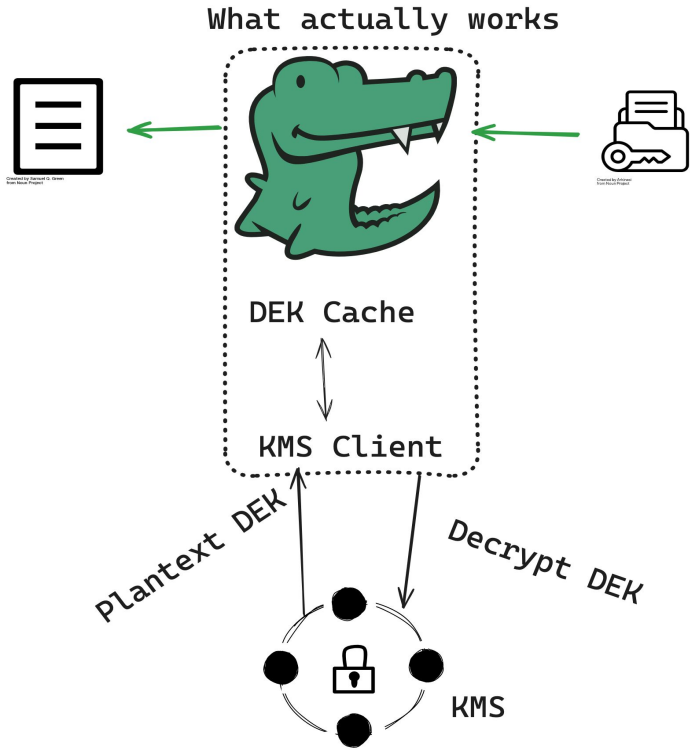
* at least as long as Kafka retention

Envelope encryption: Producing



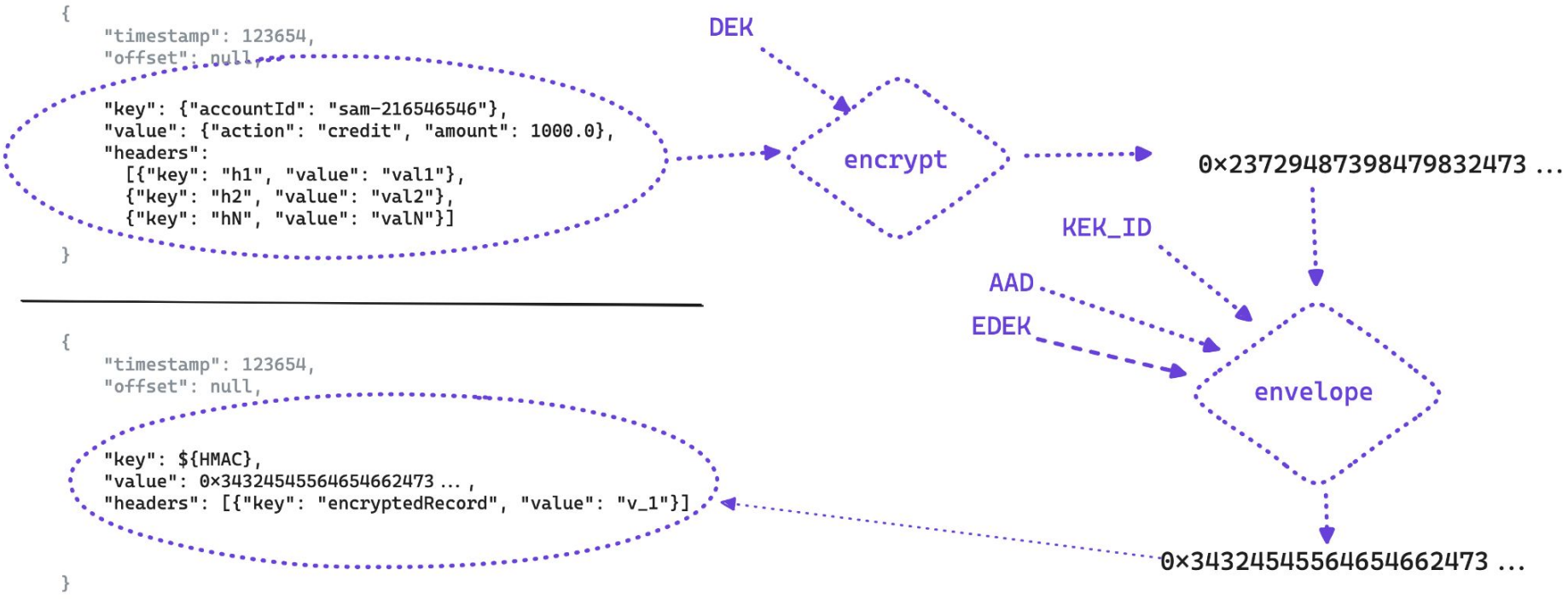
- Generates a dedicated Data Encryption Key (DEK) for encrypting messages
- Include the encrypted DEK with the cipher text sent to the broker
- Including DEK ensures that the message is decryptable later

Envelope encryption: Consuming



- Read Encrypted DEK from envelope
- KMS decrypts the DEK
- Use the decrypted DEK to decrypt the message.

The Envelope



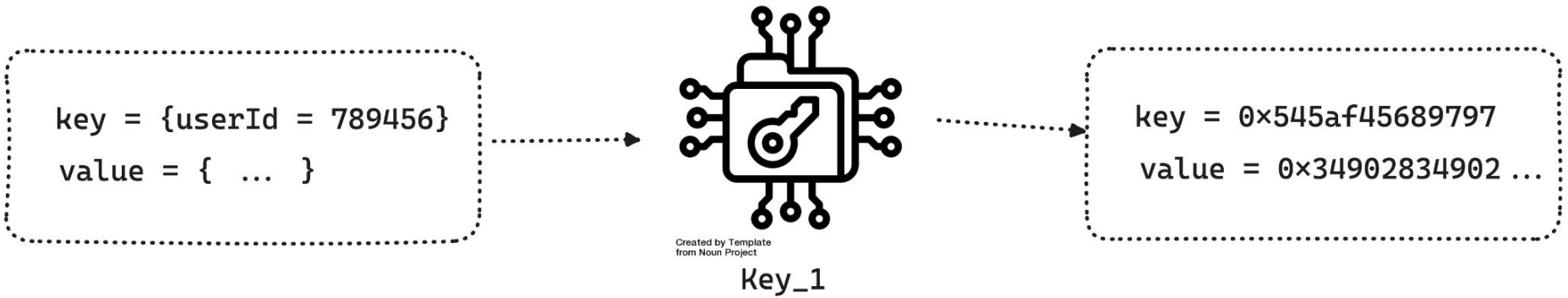
What we learnt along the way

Encrypting Record Keys

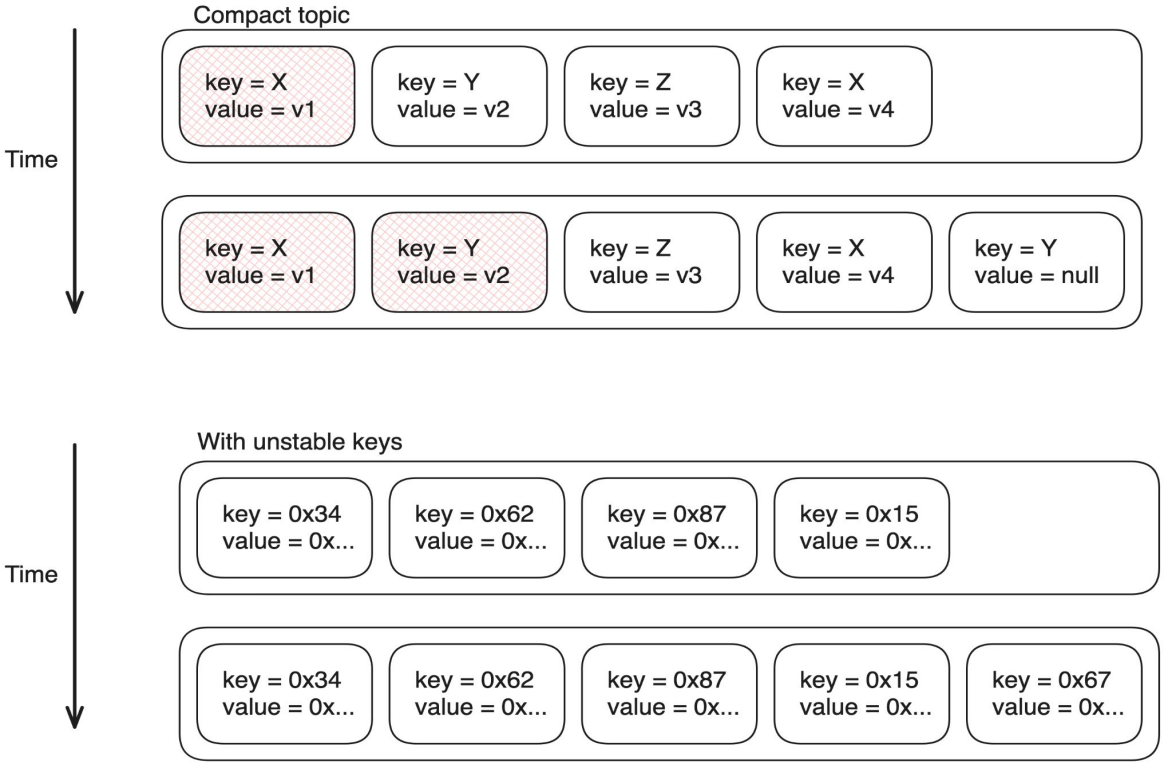
Keys are often sensitive so we want to secure them.

However they are critical to Kafka operation:

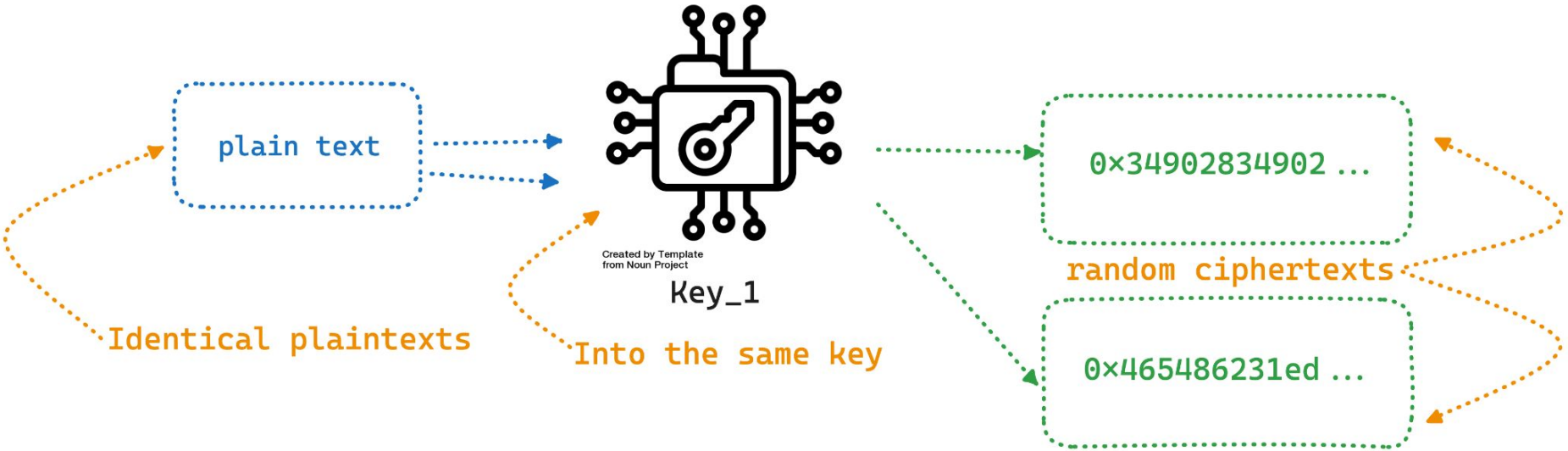
1. Partitioning
2. Compacted topics



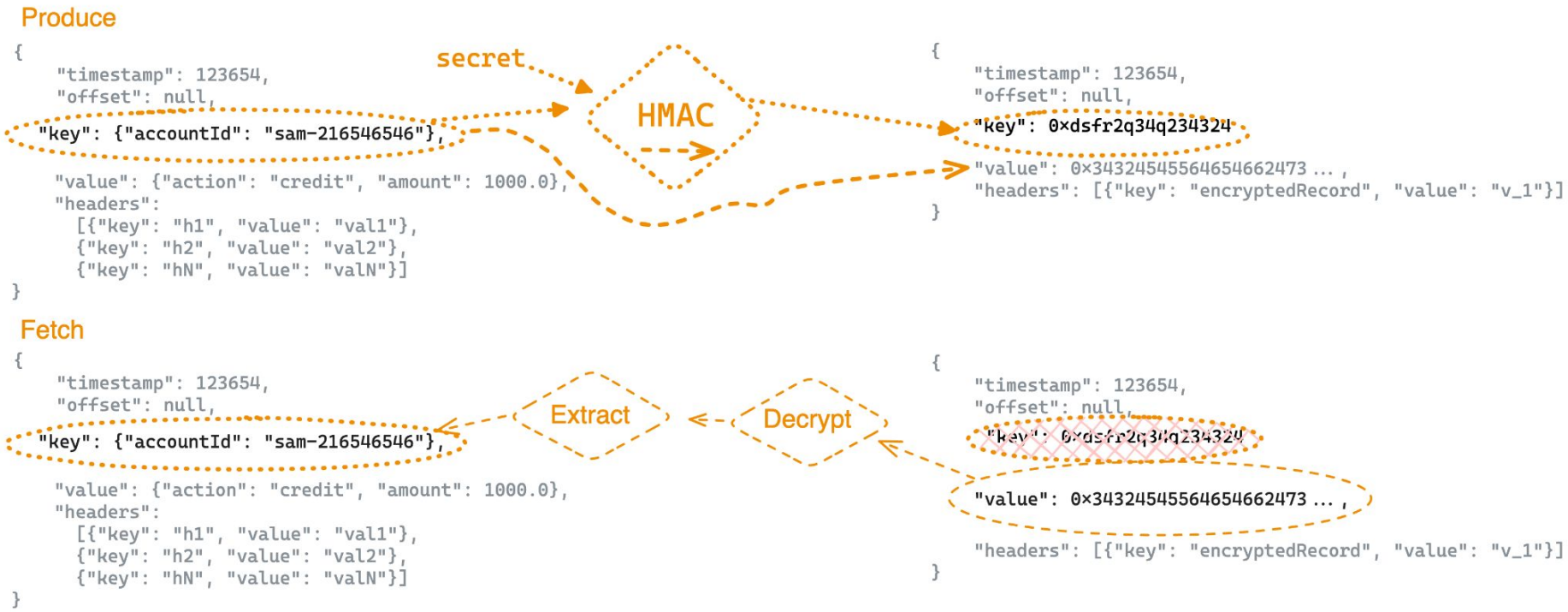
The problem with message keys



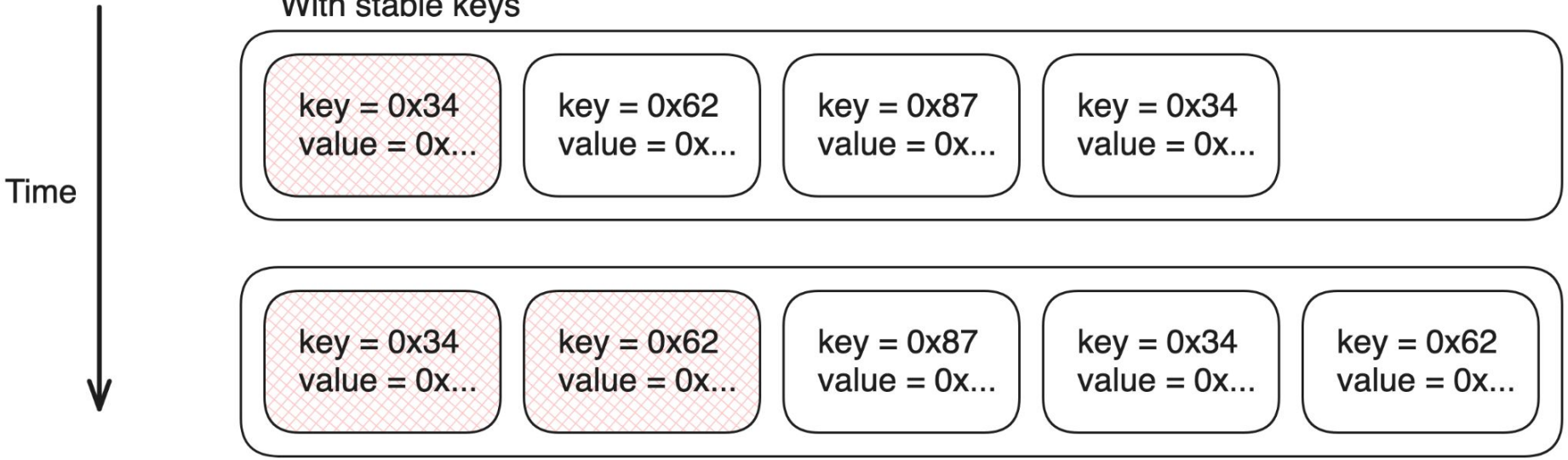
Semantic Encryption



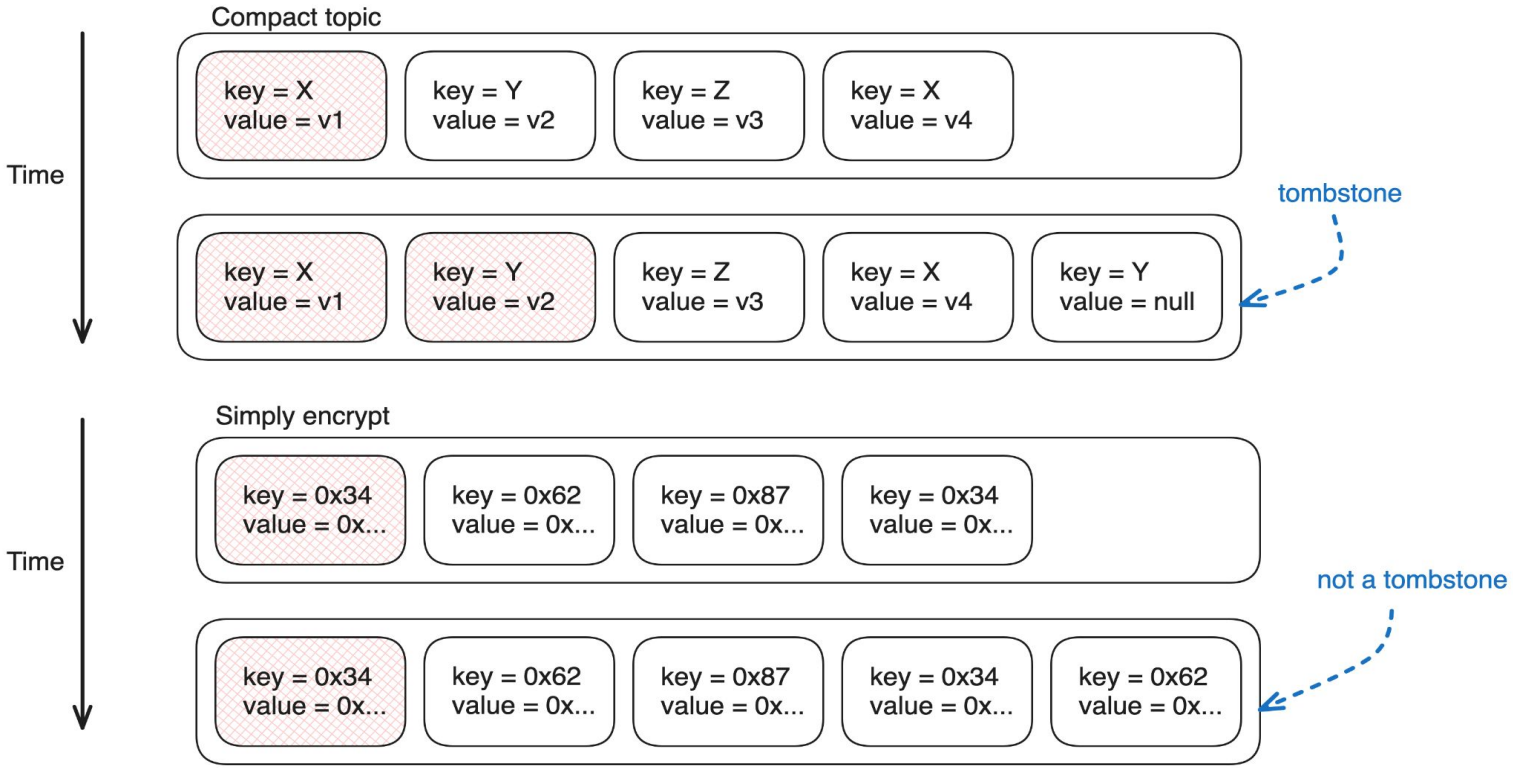
Stabilising the record key



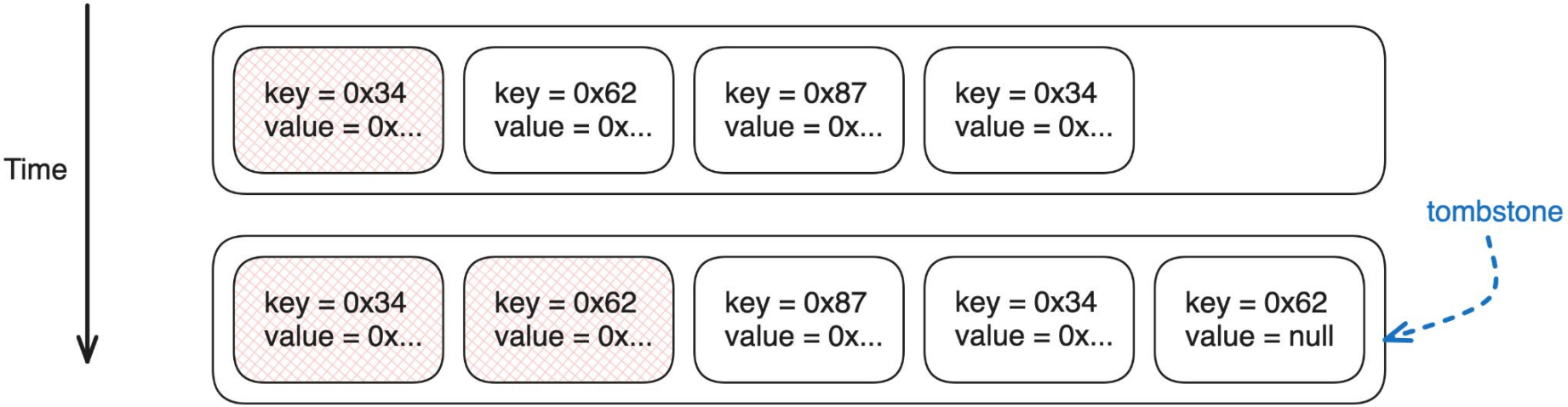
Stable keys!


With stable keys
Time
key = 0x34
value = 0x...
key = 0x62
value = 0x...
key = 0x87
value = 0x...
key = 0x34
value = 0x...
key = 0x34
value = 0x...
key = 0x62
value = 0x...
key = 0x87
value = 0x...
key = 0x34
value = 0x...
key = 0x62
value = 0x...

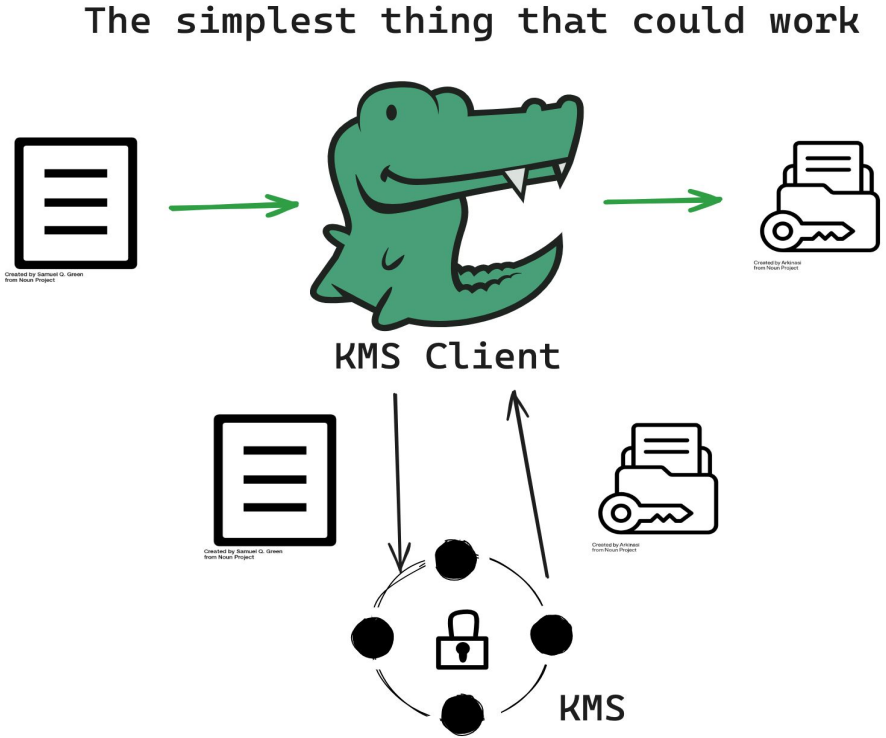
Compacting...


Compact topic
Time
key = X
value = v1
key = Y
value = v2
key = Z
value = v3
key = X
value = v4
tombstone
key = X
value = v1
key = Y
value = v2
key = Z
value = v3
key = X
value = v4
key = Y
value = null
Simply encrypt
Time
key = 0x34
value = 0x...
key = 0x62
value = 0x...
key = 0x87
value = 0x...
key = 0x34
value = 0x...
not a tombstone
key = 0x34
value = 0x...
key = 0x62
value = 0x...
key = 0x87
value = 0x...
key = 0x34
value = 0x...
key = 0x62
value = 0x...

Null detection



Overloading the KMS



- KMS has to scale to Kafka message rates...
- That costs a lot of:
  - Latency
  - Cost per call in the cloud

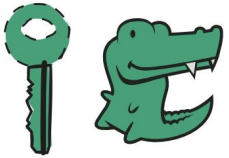
Key Rotation

Keys have finite encryption capacity in:

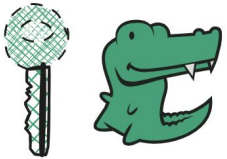
- Bytes
- Time

Created by Juicy Fish
from Noun Project

We don't rotate persisted data

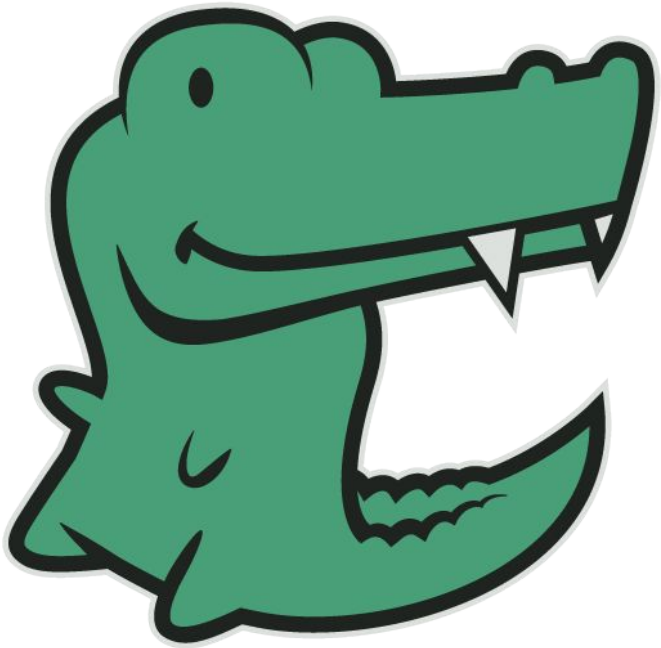


Each instance gets its own key



Instances use KMS to generate new DEKs as required.

Kroxylicious, the snappy open source proxy for Apache Kafka®



- Kafka Protocol aware proxy
- Transparent to clients and brokers
- We have a record encryption filter!
- Fully open source: Apache Software License version 2.0
- Source code: github.com/kroxylicious/kroxylicious/

Could kroxylicious work for you? Come and help us make that happen!

Questions? Comments? Places you'd like to deploy?

Thank you for your time

www.kroxylicious.io

Abstract

Description

How much can you trust your Kafka cluster? Does it have the right access or audit controls? What about at the level of the file system? Data privacy and integrity is a crucial issue for many deployments for reasons of customer confidentiality, corporate security, national laws and regulations or just applying industry best practices.

In this talk we'll explore the measures Apache Kafka takes to protect your cluster and your data, and why it isn't always enough. We'll take a look at some practical approaches to achieving encryption-at-rest and their risks and benefits, and how proxy servers can be a useful tool for addressing these problems. Lastly, exploring my team's journey to build a fully open source Layer 7 proxy for Apache Kafka, and what we've learned so far about proxying the Kafka protocol.

Attendees will gain a deeper understanding of the potential impacts of improperly secured data and the flow impacts that can have for data integrity, as well as the techniques and challenges of achieving at-rest security in Apache Kafka. They will leave with the knowledge they need to protect their own data's confidentiality, integrity, and availability.

Session: 45m with 10-15 or Q&A