Google | Privacy, Safety & Security

How to Encrypt a Cloud

Cryptographic Applications Workshop, May 2024

Fernando Lobato Meeser (felobato@google.com)
Moreno Ambrosin (ambrosin@google.com)
Qiushi Wang (qiushi@google.com)

Outline

1. Storage System Threat Model
2. Our Goals
3. Cryptographic Constructions and Primitives
4. Real World Storage Systems
5. Additional Real World Constraints
6. Conclusion

Storage System Threat Model

Encryption at Rest

Encryption at Rest

Storage Systems

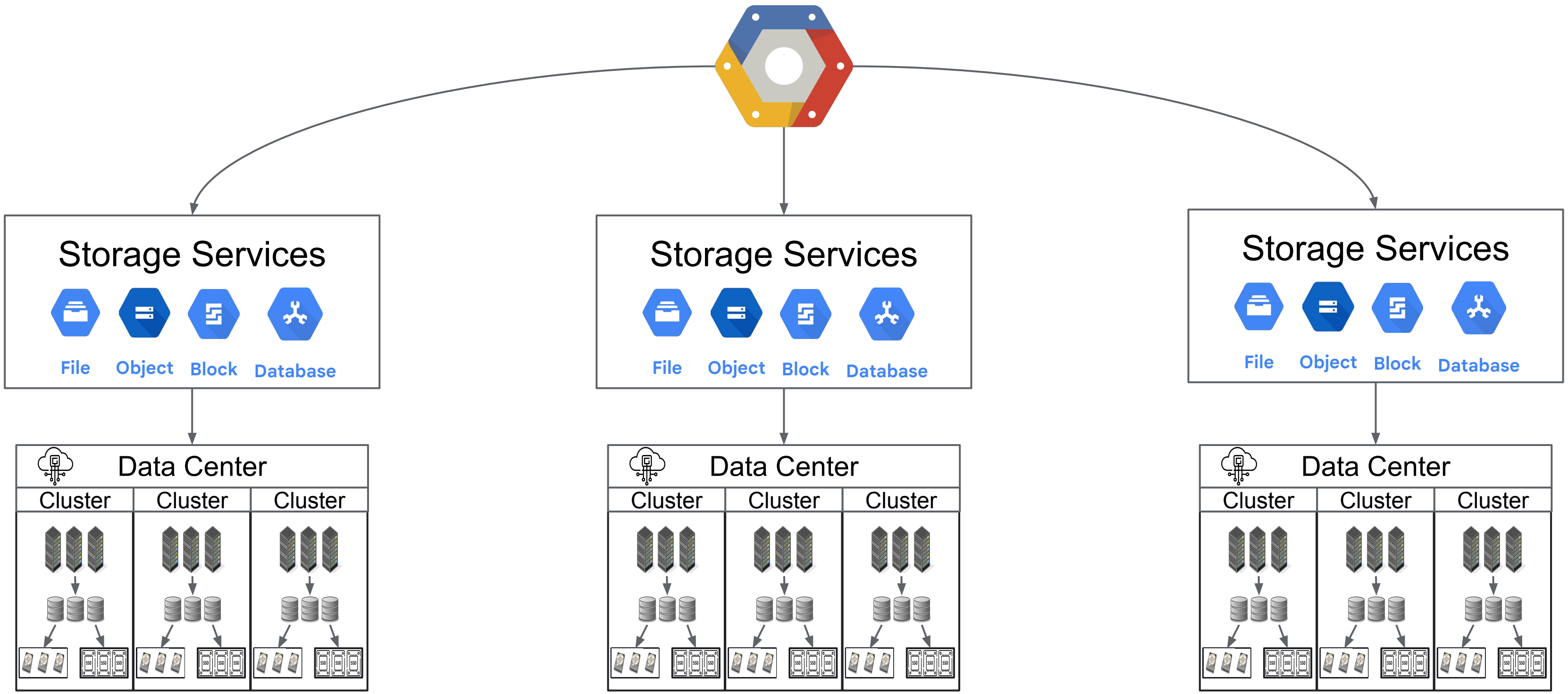
File

Object

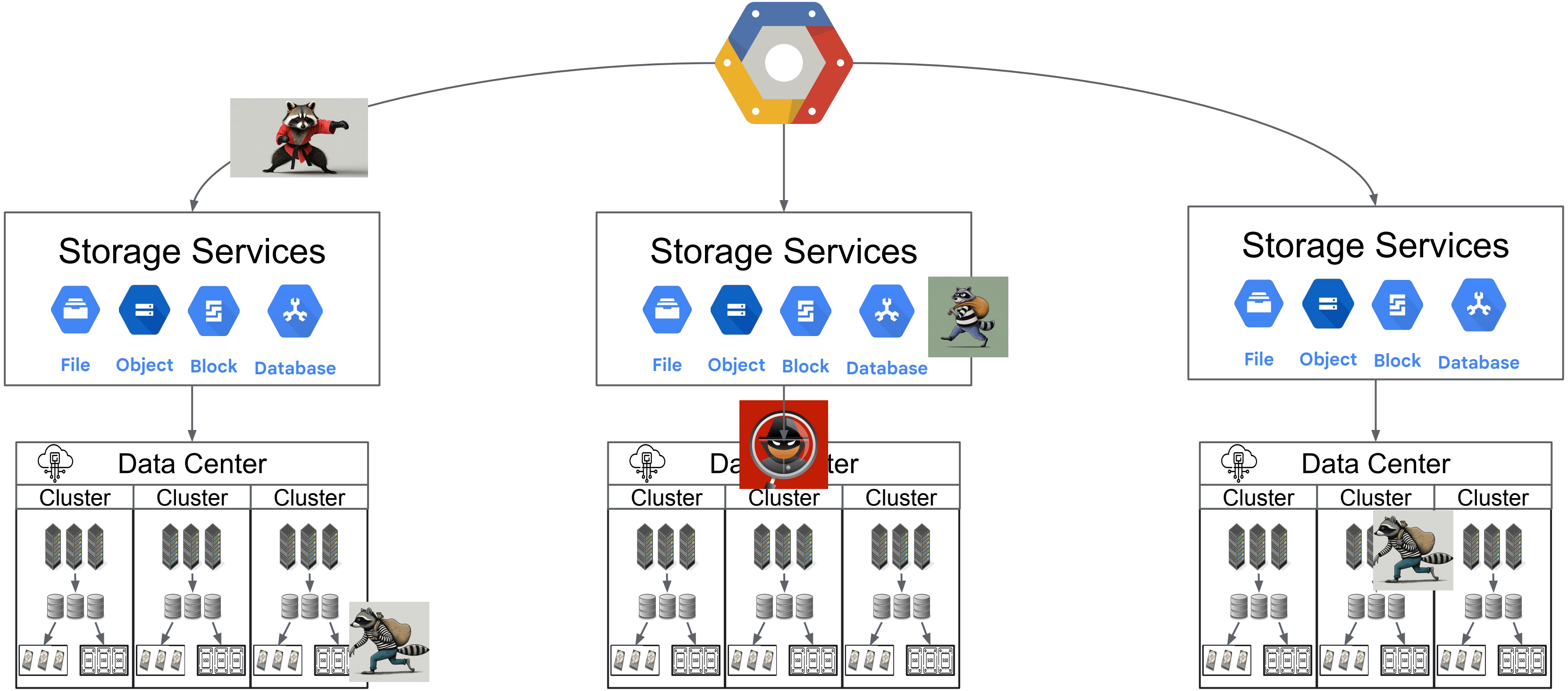
Block

Database

Encryption at ~~Rest~~ Scale



Encryption at ~~Rest~~ Scale



Our Goals

Our Goals

Uniform Threat Modeling For Storage Encryption

All data is protected with well understood security properties and hardened, unified implementations.

Unique API for Storage Encryption at Google

Provide a single point of adoption for storage wide initiatives such as silent data corruption, hardware offloads, performance optimizations.

Threat Model Guidelines

- Key Compartmentalization
  - Which key? Who has access to keys? etc.
- Minimize trust assumption in the infrastructure
  - Maintain security in the case of lateral compromise

Security Properties

Define an individual data unit (File, Object, Disk, Database*). Properties over the unit:

- **Confidentiality**
- **Authenticity**
- **Resistance vs Segment Reordering Attacks**
- **Resistance vs Segments Swap or Append Across Units**

Cryptographic Constructions and Primitives

Primitives

File

Object

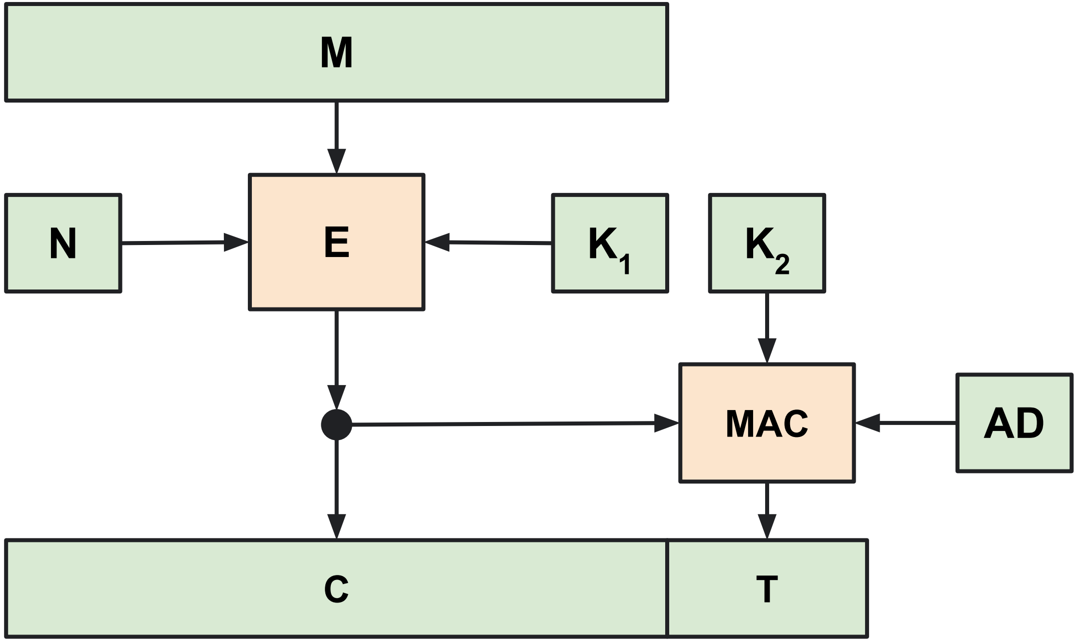
Block

Database

Online AEAD

AEAD

AEAD



Online AEAD

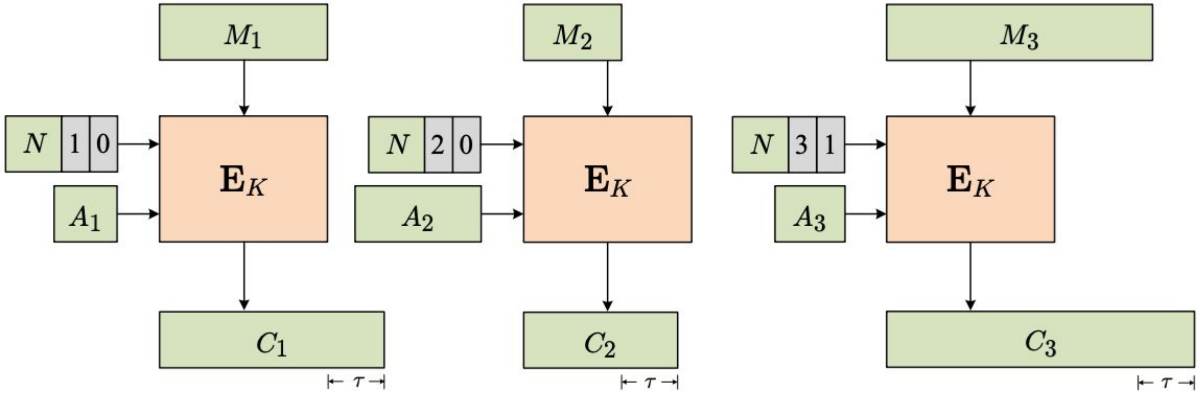


Fig. 10: **The STREAM construction for nOAE.** Encryption scheme $\Pi = (\mathbf{K}, \mathbf{E}, \mathbf{D})$ secure as an nAE with ciphertext expansion τ is turned into a segmented-AE scheme $\Pi' = (\mathcal{K}, \mathcal{E}, \mathcal{D}) = \mathbf{STREAM}[\Pi, \langle\cdot\rangle]$ with key space $\mathcal{K} = \mathbf{K}$.

Hoang, Viet Tung, et al. "Online authenticated-encryption and its nonce-reuse misuse-resistance." Advances in Cryptology--CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I 35. Springer Berlin Heidelberg, 2015.

Tink StreamingAEAD mainly follows this approach with some differences.

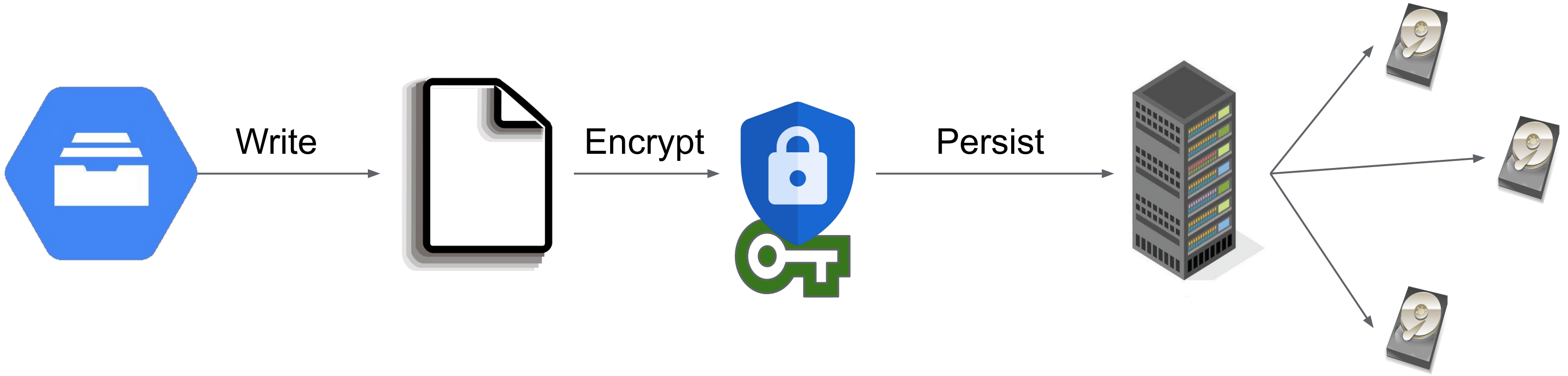
Real World Storage Systems

Append-only file system

Requirements:

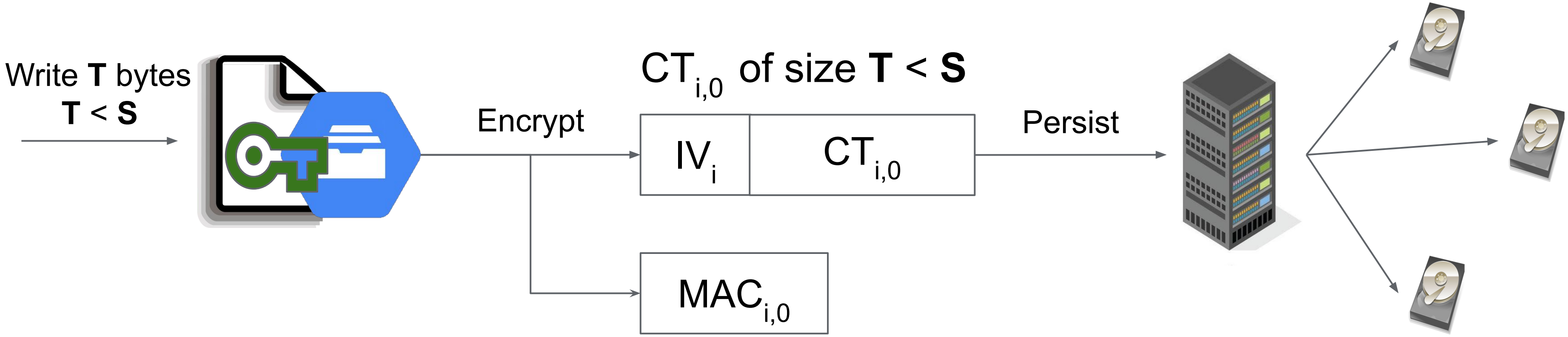
- Efficient substring reads/random reads to any particular offset (Fixed segment size)
- Incremental appends to end of file (Flush)
- Reopen a file to keep appending
- Truncate a file
- Rollback a file to a previous version, then continue appending

Append-only file system



Incremental Appends

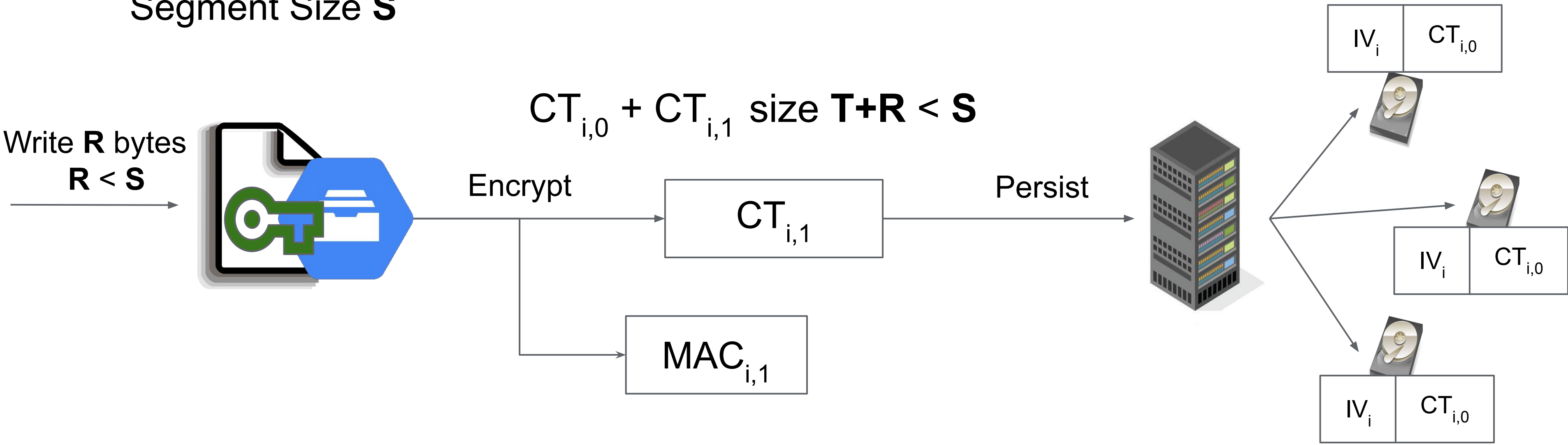
Segment Size **S**



- IV_i || $CT_{i,0}$ are persisted and replicated
- $MAC_{i,0}$ is stored temporarily separately

Incremental Appends

Segment Size **S**

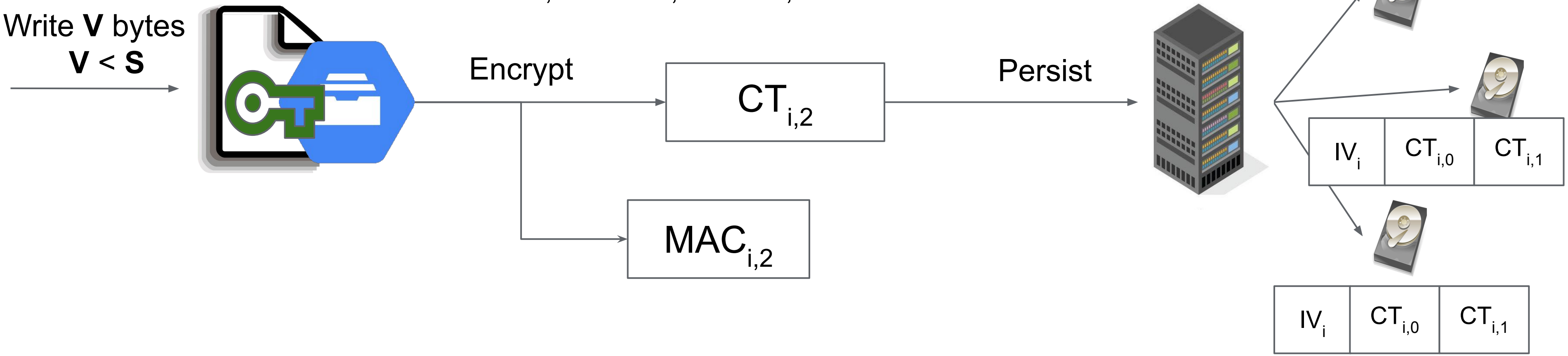


- Segment ciphertext is now: IV_i || $CT_{i,0}$ || $CT_{i,1}$
- $MAC_{i,1}$ replaces $MAC_{i,0}$

Incremental Appends

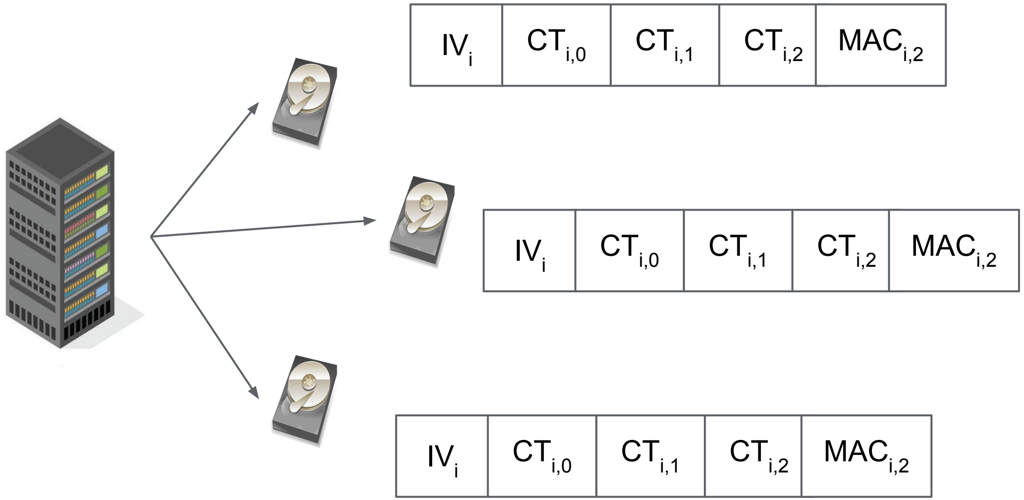
Segment Size **S**

$CT_{i,0} + CT_{i,1} + CT_{i,2}$ size $\mathbf{V + T + R >= S}$



Segment ciphertext is now: $IV_i \,||\, CT_{i,0} \,||\, CT_{i,1} \,||\, CT_{i,2} \,||\, MAC_{i,2}$

Incremental Appends



Segment ciphertext is now: $IV_i \mathbin{||} CT_{i,0} \mathbin{||} CT_{i,1} \mathbin{||} CT_{i,2} \mathbin{||} MAC_{i,2}$

Why not STREAM?

- Ability to append to existing ciphertext (no finalize bit)
  - Files use frequently snapshotting
- Re-encryption is expensive (read only FS)
  - After writing, file can be replicated.

Incremental STREAM?

We need a mode that combines Incremental AEAD with Online AEAD.

Subtle points:

- No end of stream: Append == Truncate
- Same Key, IV for more than one MAC
- Can’t use nonce-misuse resistant schemes (double pass)

Sasaki, Y., Yasuda, K. (2016). A New Mode of Operation for Incremental Authenticated Encryption with Associated Data. In: Dunkelman, O., Keliher, L. (eds) Selected Areas in Cryptography – SAC 2015. SAC 2015. Lecture Notes in Computer Science(), vol 9566. Springer, Cham.

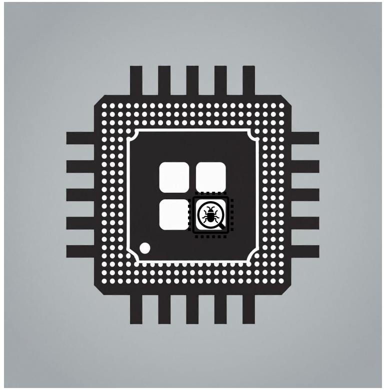
AEAD Algorithms Limitations

- Using deterministic IVs in a stateful distributed systems is a bad idea
- Number of invocation on safe invocation on an AEAD
    - AES-GCM: 2^32 isn't much for cloud scale
- Constant re-keying is expensive + multi user setting attacks
- Performance is critical

Additional Real World Constraints

Silent Data Corruption

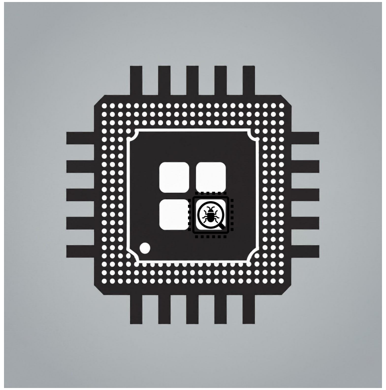
- Issue that impacts various levels (memory, storage, network, CPU)
    - HEAP stomping, SW bugs
- SDC occurs when an impacted CPU causes errors/miscalculations
- May be caused by “mercurial cores”[1]
    - Defects in processors
    - Faults can be deterministic
    - Don’t always manifest



[1]*Hochschild, Peter H., et al. "Cores that don't count." Proceedings of the Workshop on Hot Topics in Operating Systems. 2021.*

Silent Data Corruption

- SDC poses unique challenges for cryptography
- Ciphertexts indistinguishable from random (hard to validate correctness)
- Random IV means encrypting twice gets two different results
- Corrupted encryption = data loss (crypto shredding)
- Cryptographic integrity expensive (and may require RPC)



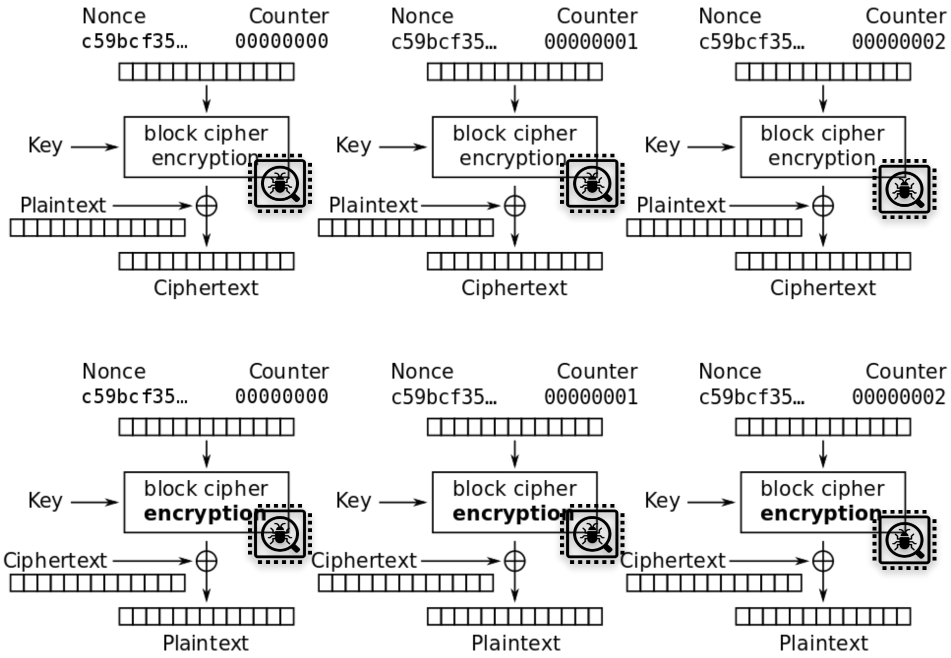
Silent Data Corruption

- Faults can happen during encryption, or to the encryption context!
- **General heuristic:** Encrypt, checksum, then verify (decrypt right away)
    - E.g., CRC32C
    - Decryption is not free
    - May not protect against *deterministic hardware faults*
        - One may pin to a different core (expensive!)
        - Alternative circuit? Self-verifying construction?

Silent Data Corruption

Encryption

Decryption



Counter (CTR) mode decryption

Compliance

- Security != Compliance
- Limited set of tools at our disposal - often can’t use new, shiny things!
- Systems grow, get connected to other systems.
- It’s easier to build with compliance in mind from inception.

Conclusion

- Standard cryptographic primitives and constructions don’t fully match the real world.

- At scale, fault tolerance against faults is extremely important.

- Compliance can limit the algorithms available to us, as well as the way in which we can use such algorithms.

Q&A