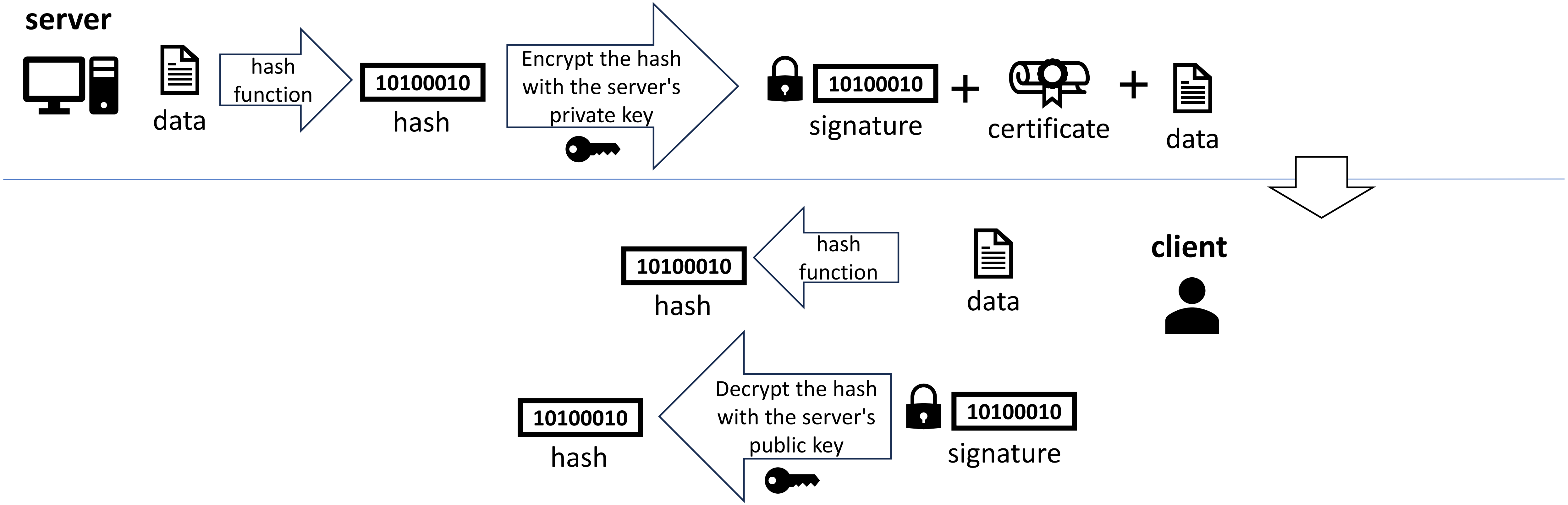
Portunus: Re-imagining Access Control in Distributed Systems

ATC'23

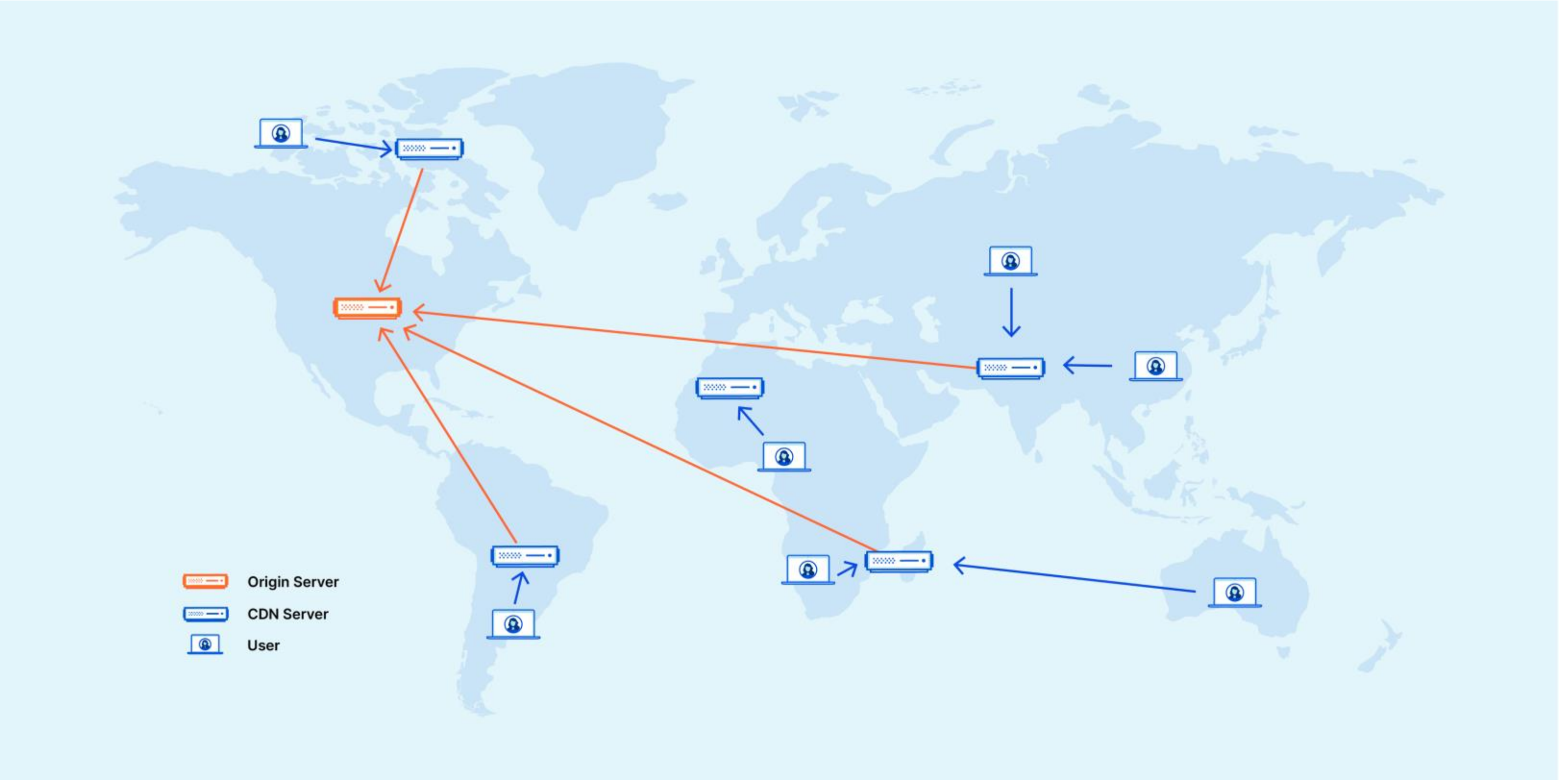
Background : TLS Handshake

- TLS is an **encryption** and **authentication** protocol.



Background : TLS termination

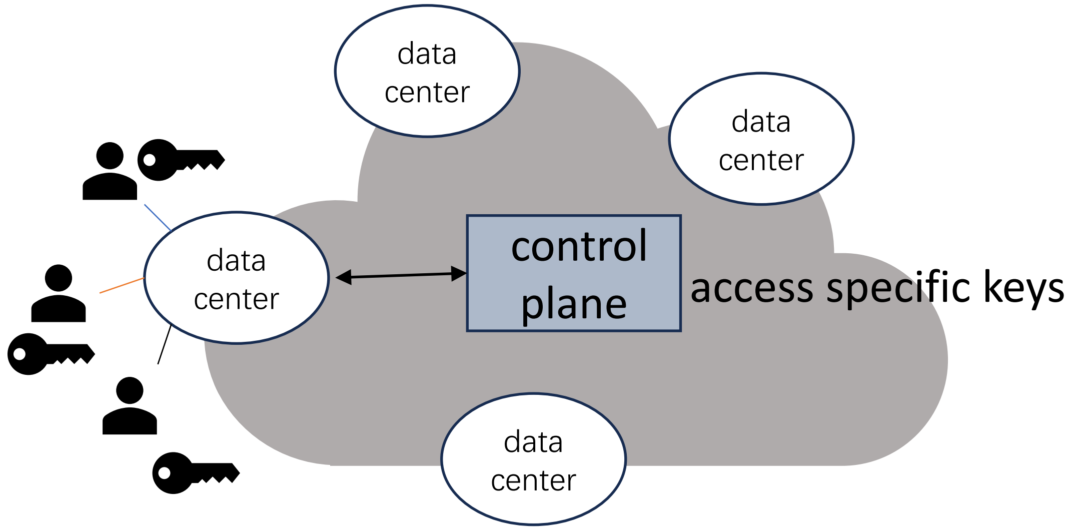
- The process of intercepting a TLS connection at an intermediary point in the network is called **TLS termination**.
- Website operators often enlist the services of **infrastructure providers** like Content Delivery Networks (CDNs).

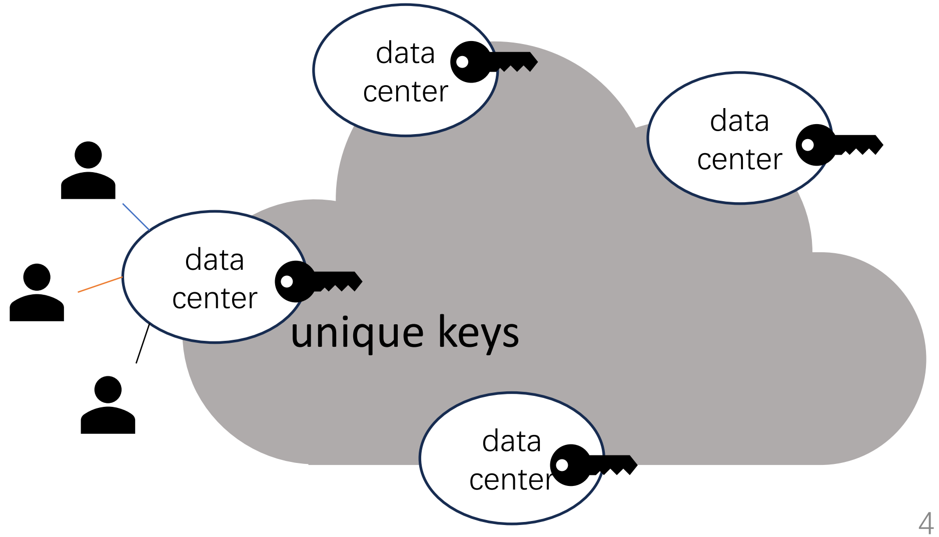


service providers require access to **the private key** of customers

Background : Access Control

- Customers would like providers to **control access** to their key material based on **geographical** and **security properties**.
- Traditional access control mechanisms :
  1. Centralized method
  2. Use standard public-key encryption





Problem

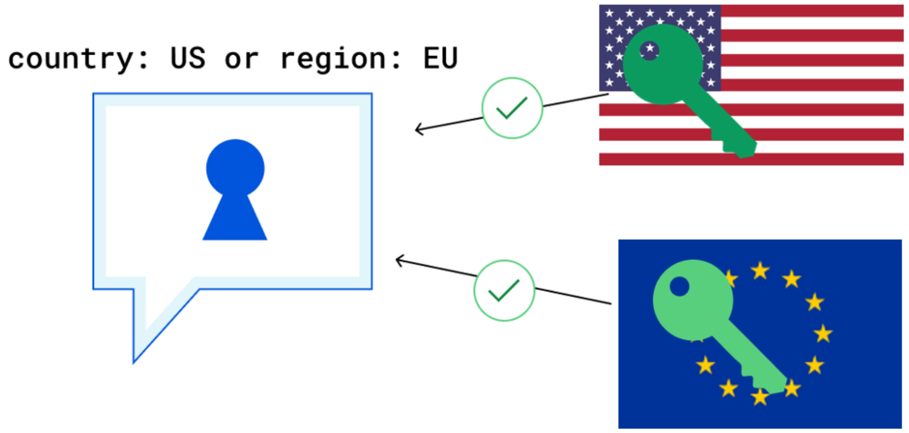
- Centralized method :
  1. expensive **round-trip** which adds latency and reduces reliability.

- Use standard public-key encryption:
  1. Becomes rather complex to manage in the face of **heterogeneous policies** and **large scale.**
  2. Newly added centers cannot participate in establishing TLS connections.

Main Idea

- Portunus : uses a variant of traditional public key cryptography called **ciphertext-policy attribute-based encryption (CP-ABE)**

1. Key Distribution
2. Encrypting customer keys
3. Accessing customer keys
4. Key Rotation
5. Attribute Changes



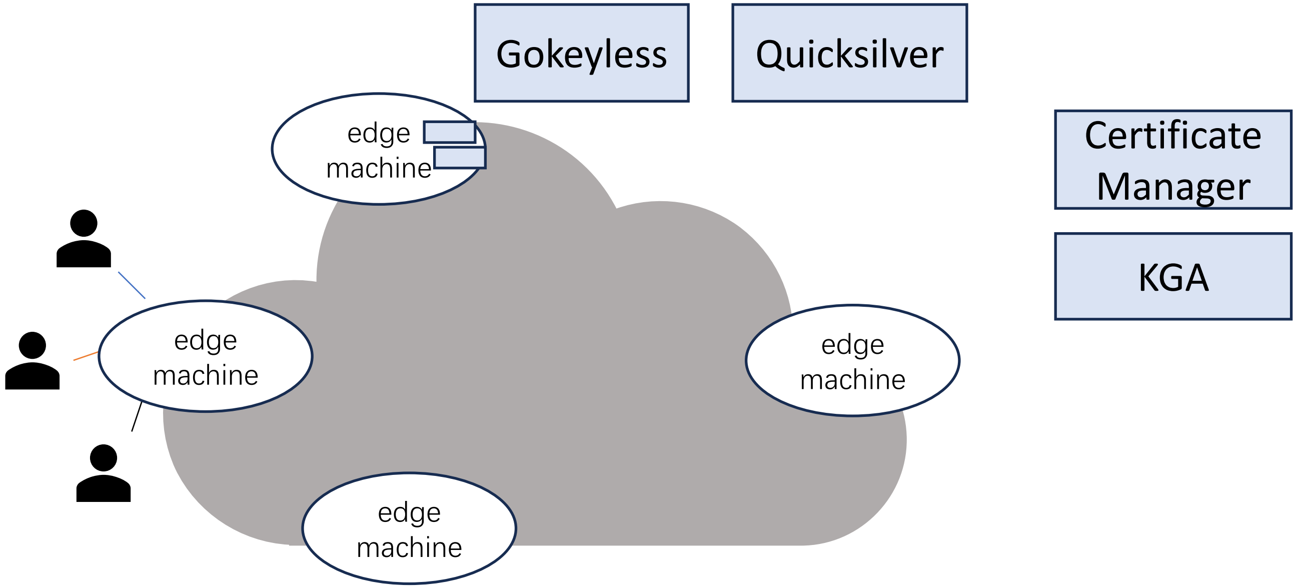
country: US or region: EU

- $\mathrm{Setup}(\lambda) \rightarrow (\mathrm{MPK}, \mathrm{MSK})$
- $\mathrm{KeyGen}(\mathrm{MSK}, S) \rightarrow \mathrm{SK}_S$
- $\mathrm{Encrypt}(\mathrm{MPK}, \mathbb{A}, M) \rightarrow \mathrm{CT}_{\mathbb{A}}$
- $\mathrm{Decrypt}(\mathrm{SK}_S, \mathrm{CT}_{\mathbb{A}}) \rightarrow M'$

S : a set of attributes
A: an access policy

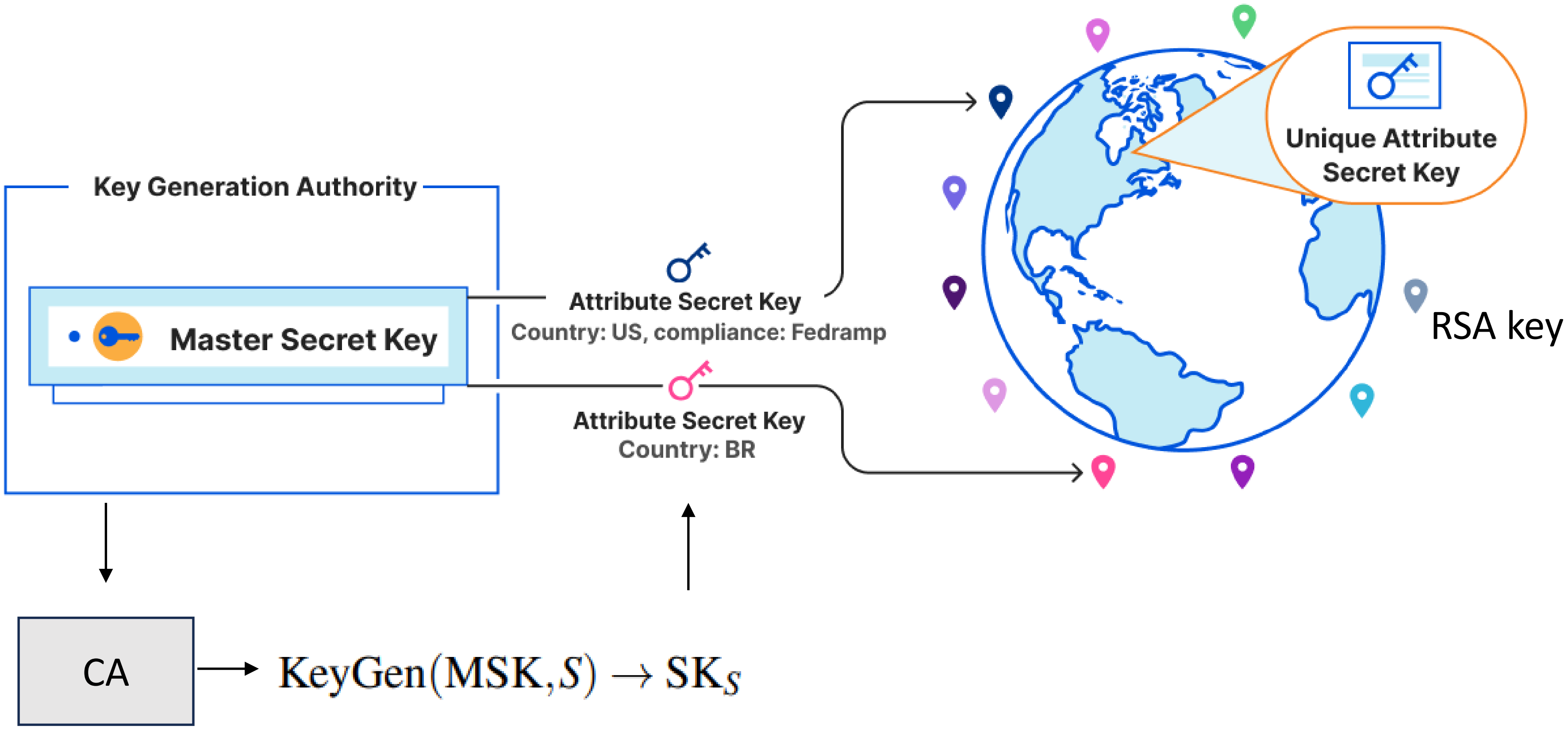
Architecture

- **Cloudflare** logically has four components.
  (Portunus has deployed across Cloudflare's 400+ global data centers)
  1. Edge machines
  2. A set of services in the control-plane
  3. Key Generation Authority (**KGA**)
  4. A globally synchronized key-value store, **Quicksilver**



Key Distribution

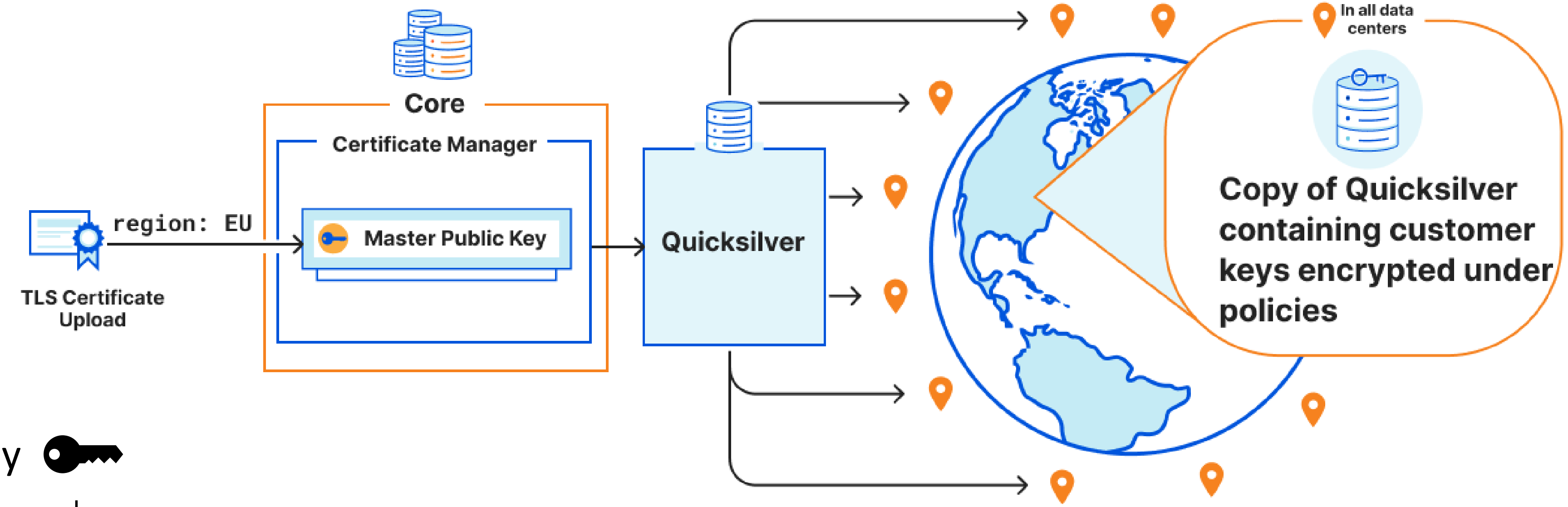
- $\text{Setup}(\lambda) \to (\text{MPK}, \text{MSK})$
- $\text{KeyGen}(\text{MSK}, S) \to \text{SK}_S$
- $\text{Encrypt}(\text{MPK}, \mathbb{A}, M) \to \text{CT}_{\mathbb{A}}$
- $\text{Decrypt}(\text{SK}_S, \text{CT}_{\mathbb{A}}) \to M'$

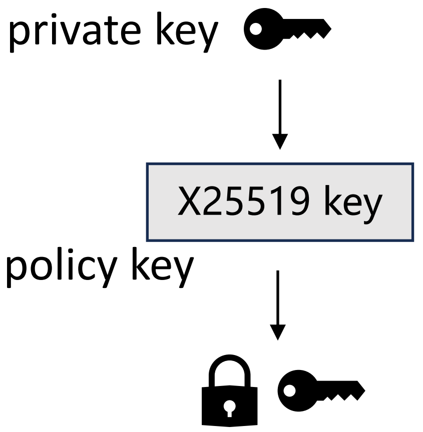


$\text{KeyGen}(\text{MSK}, S) \to \text{SK}_S$

Encrypting customer keys

- $\mathrm{Setup}(\lambda) \to (\mathrm{MPK}, \mathrm{MSK})$
- $\mathrm{KeyGen}(\mathrm{MSK}, S) \to \mathrm{SK}_S$
- $\boxed{\mathrm{Encrypt}(\mathrm{MPK}, \mathbb{A}, M) \to \mathrm{CT}_{\mathbb{A}}}$
- $\mathrm{Decrypt}(\mathrm{SK}_S, \mathrm{CT}_{\mathbb{A}}) \to M'$

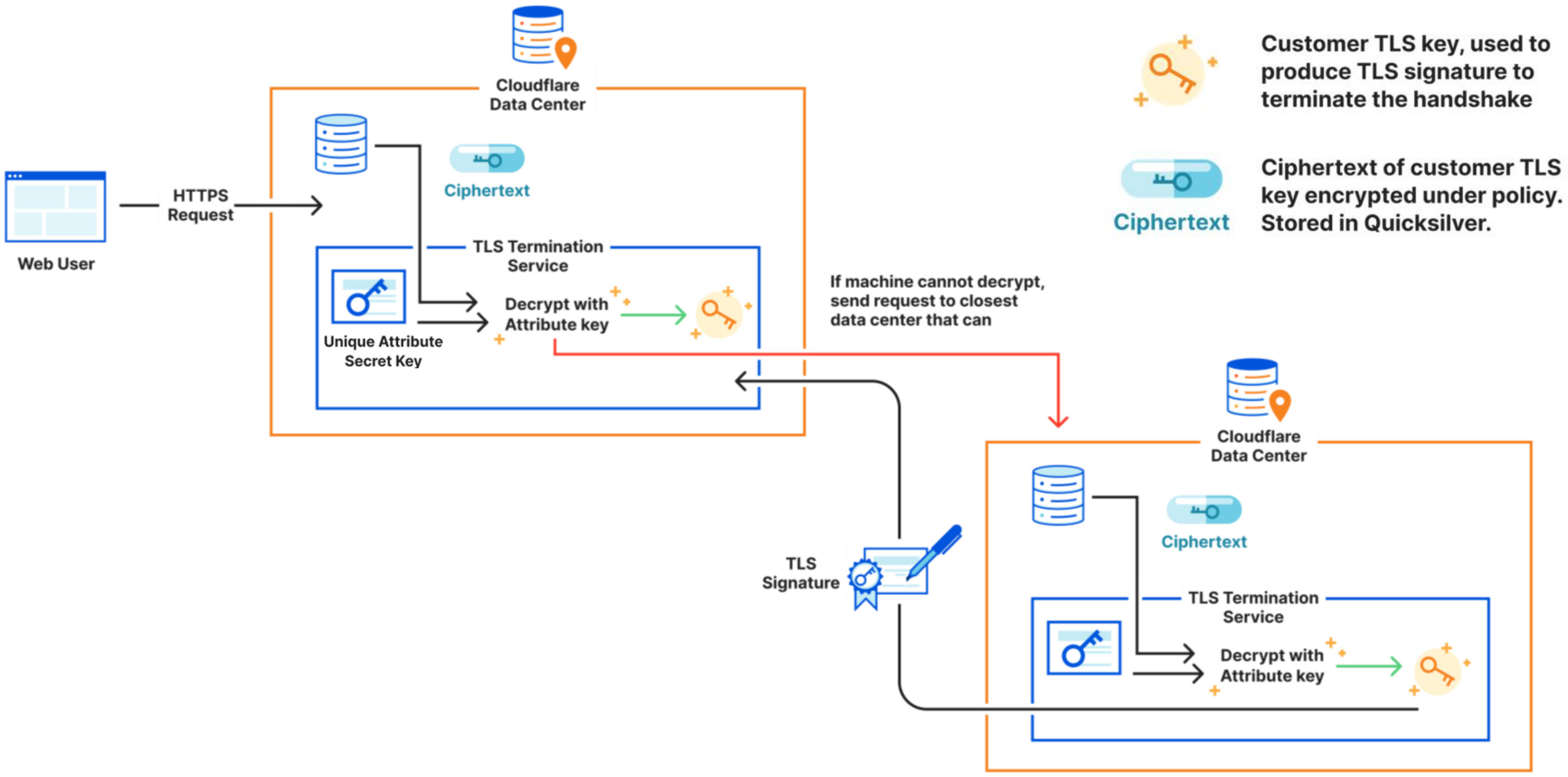




$\longrightarrow \mathrm{Encrypt}(\mathrm{MPK}, \mathbb{A}, M) \to \mathrm{CT}_{\mathbb{A}}$

Accessing customer keys

- $\mathsf{Setup}(\lambda) \rightarrow (\mathsf{MPK}, \mathsf{MSK})$
- $\mathsf{KeyGen}(\mathsf{MSK}, S) \rightarrow \mathsf{SK}_S$
- $\mathsf{Encrypt}(\mathsf{MPK}, \mathbb{A}, M) \rightarrow \mathsf{CT}_{\mathbb{A}}$
- $\boxed{\mathsf{Decrypt}(\mathsf{SK}_S, \mathsf{CT}_{\mathbb{A}}) \rightarrow M'}$



Customer TLS key, used to produce TLS signature to terminate the handshake

Ciphertext of customer TLS key encrypted under policy. Stored in Quicksilver.

- $\mathrm{Setup}(\lambda) \rightarrow (\mathrm{MPK}, \mathrm{MSK})$
- $\mathrm{KeyGen}(\mathrm{MSK}, S) \rightarrow \mathrm{SK}_S$
- $\mathrm{Encrypt}(\mathrm{MPK}, \mathbb{A}, M) \rightarrow \mathrm{CT}_{\mathbb{A}}$
- $\mathrm{Decrypt}(\mathrm{SK}_S, \mathrm{CT}_{\mathbb{A}}) \rightarrow M'$

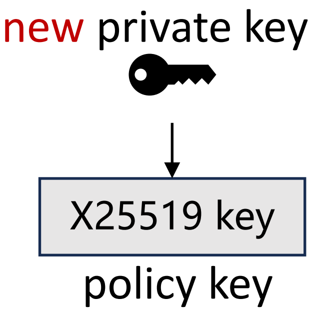
Key Rotation

- When attackers know **MPK**, **MSK** and ***M***, they can infer the private key.
- The lifetime of a customer certificate can extend beyond a rotation period

$$\mathrm{Setup}(\lambda) \rightarrow \overset{\text{new}}{(\mathrm{MPK}, \mathrm{MSK})}$$

$$\mathrm{KeyGen}(\mathrm{MSK}, S) \rightarrow \overset{\text{new}}{\mathrm{SK}_S}$$

$$\underset{\text{old}}{\mathrm{SK}_S}$$

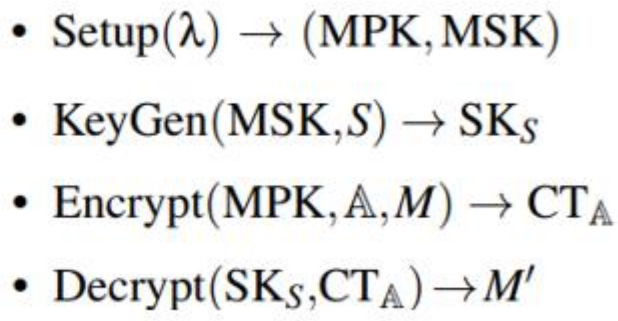


new private key

X25519 key

policy key

Attribute Changes



- $\mathrm{Setup}(\lambda) \rightarrow (\mathrm{MPK}, \mathrm{MSK})$
- $\mathrm{KeyGen}(\mathrm{MSK}, S) \rightarrow \mathrm{SK}_S$
- $\mathrm{Encrypt}(\mathrm{MPK}, \mathbb{A}, M) \rightarrow \mathrm{CT}_{\mathbb{A}}$
- $\mathrm{Decrypt}(\mathrm{SK}_S, \mathrm{CT}_{\mathbb{A}}) \rightarrow M'$

- Introduce new label : the data center is almost unaffected

- Change existing attributes: need a transition

  1. The affected label is removed from the forwarding information**.**
  2. the key(SKs) is re-issued with the new attribute.
  3. the new attribute is re-added to the forwarding information

Evaluation

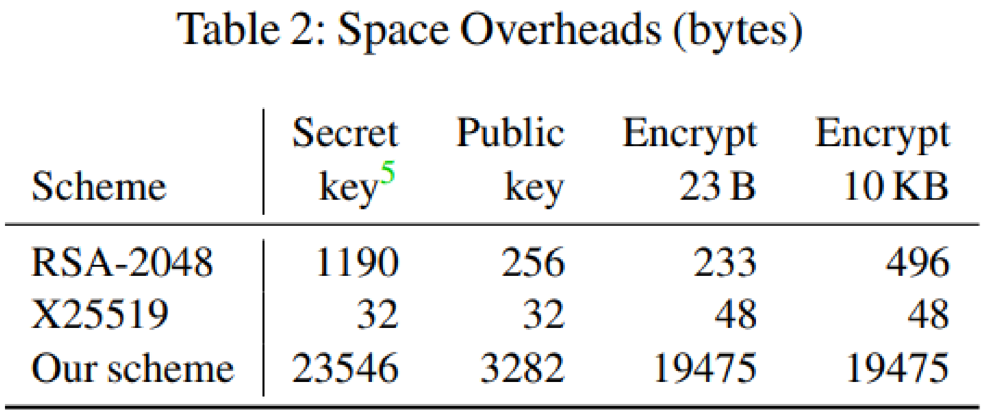


Table 2: Space Overheads (bytes)

Scheme	Secret key[5]	Public key	Encrypt 23 B	Encrypt 10 KB
RSA-2048	1190	256	233	496
X25519	32	32	48	48
Our scheme	23546	3282	19475	19475

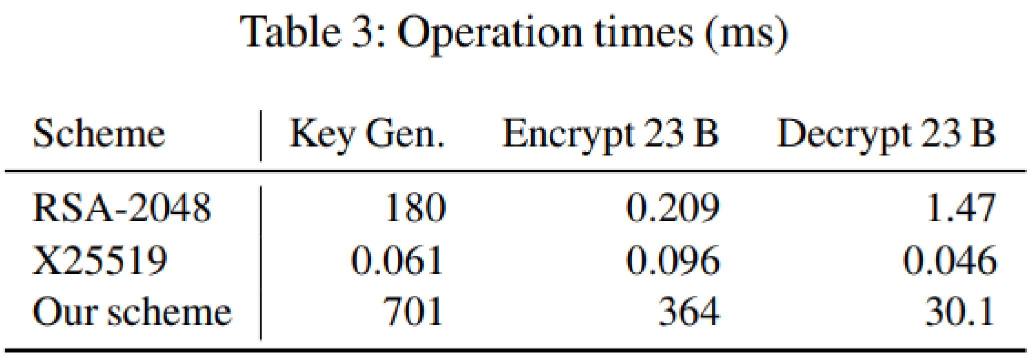


Table 3: Operation times (ms)

Scheme	Key Gen.	Encrypt 23 B	Decrypt 23 B
RSA-2048	180	0.209	1.47
X25519	0.061	0.096	0.046
Our scheme	701	364	30.1

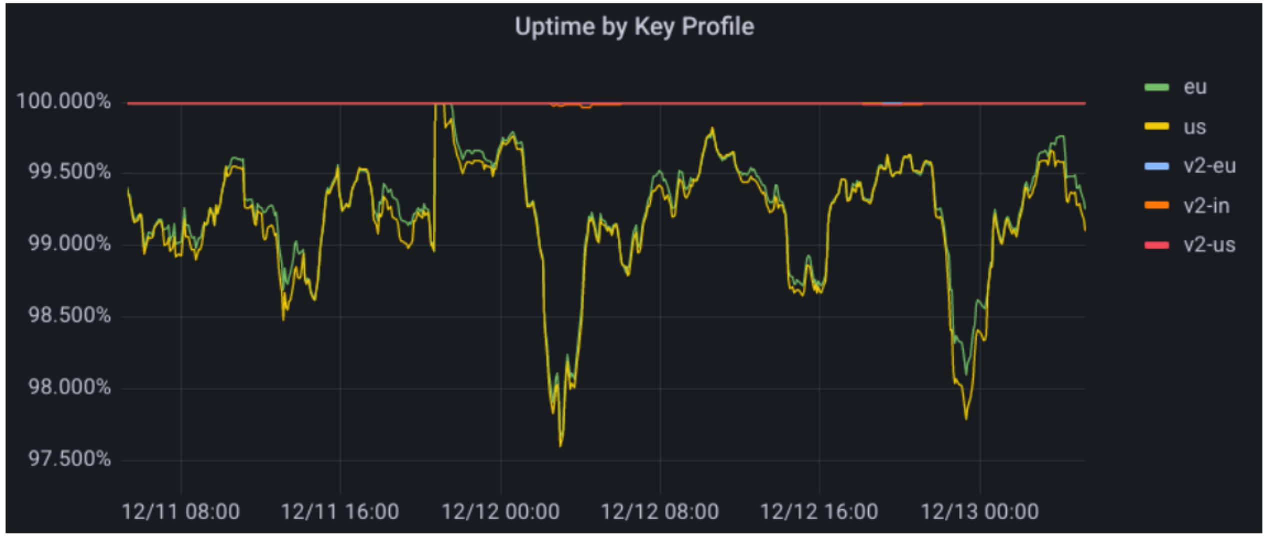
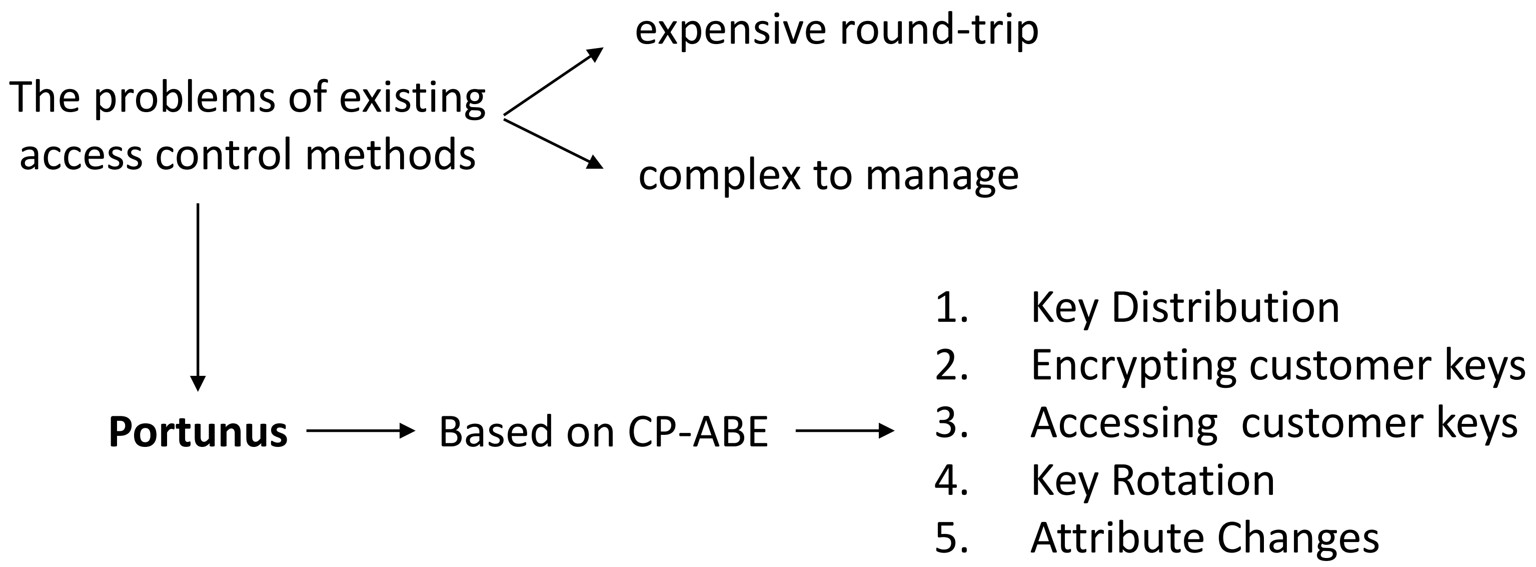


Figure 3: Uptime by policy; this shows that Portunus (v2) has consistently better uptime than Geo Key Manager (v1)

Summary



The problems of existing access control methods

expensive round-trip

complex to manage

Portunus

Based on CP-ABE

1. Key Distribution
2. Encrypting customer keys
3. Accessing customer keys
4. Key Rotation
5. Attribute Changes