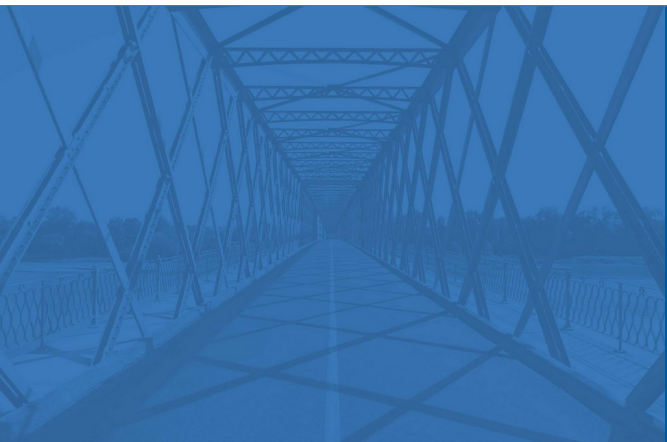




Chiffrement Transparent pour PostgreSQL

octobre 2026

Damien Clochard – Lead Dev

TDE, la dernière pièce manquante de Postgres ?

En 2025, DALIBO a lancé le projet [The Cryptic Elephant](#)

100 % open-source, pas de vendor lock-in

Mêmes Principes que PostgreSQL Anonymizer

Une longue histoire....

Notre Approche

Feuille de Route

Comment nous aider ?

Commençons par des définitions...

Le terme “TDE” est flou et recouvre plusieurs approches différentes:

TDE : Transparent Data Encryption

CFE : Cluster Files Encryption

TCE : Transparent Column Encryption

FDE : Full Disk Encryption

Une Longue Histoire

Beaucoup de tentatives, depuis plus de 10 ans

Pas d'implémentation dans le coeur de Postgres à moyen terme

Plusieurs Forks Propriétaires en compétition

Forks Propriétaires

Amazon Aurora

EDB TDE

Fujitsu Enterprise Postgres

Cybertec

Crunchy Hardened Postgres

Percona TDE

Problèmes

Coût des licences / services

Fortes Limitations (versions / upgrade / ...)

Solution Propriétaire - Pas d'audits de code possibles

Vendor Lock In

Incertitudes sur l'avenir de ces projets

TDE protège contre...

Le vol de volumes de stockage

Mise au rebut non destructive

TDE ne protège pas contre...

Attaque sur le système en fonctionnement

Compromission de l'application

Injection SQL

Élévation de privilèges sur le serveur

Administrateurs malveillants

Attaques réseau sur les connexions client-serveur

FDE plutôt que TDE...

FDE est simple et suffisant dans 90% des cas

Même principe que TDE (“chiffrement au repos”)

Mêmes protections que TDE

Beaucoup plus simple à mettre en place

⇒ La valeur ajoutée de TDE face à FDE est très limitée

Mais les Normes évoluent...

Les nouvelles réglementations (DORA / PCI DSS 4.0)
demandent 3 niveaux de chiffrement

data at rest + data in transit + data in use

Pour NIS2 le chiffrement “data in use”
n’est pas obligatoire mais recommandé

Storage



IO ops



Database



TCP / IP



Application



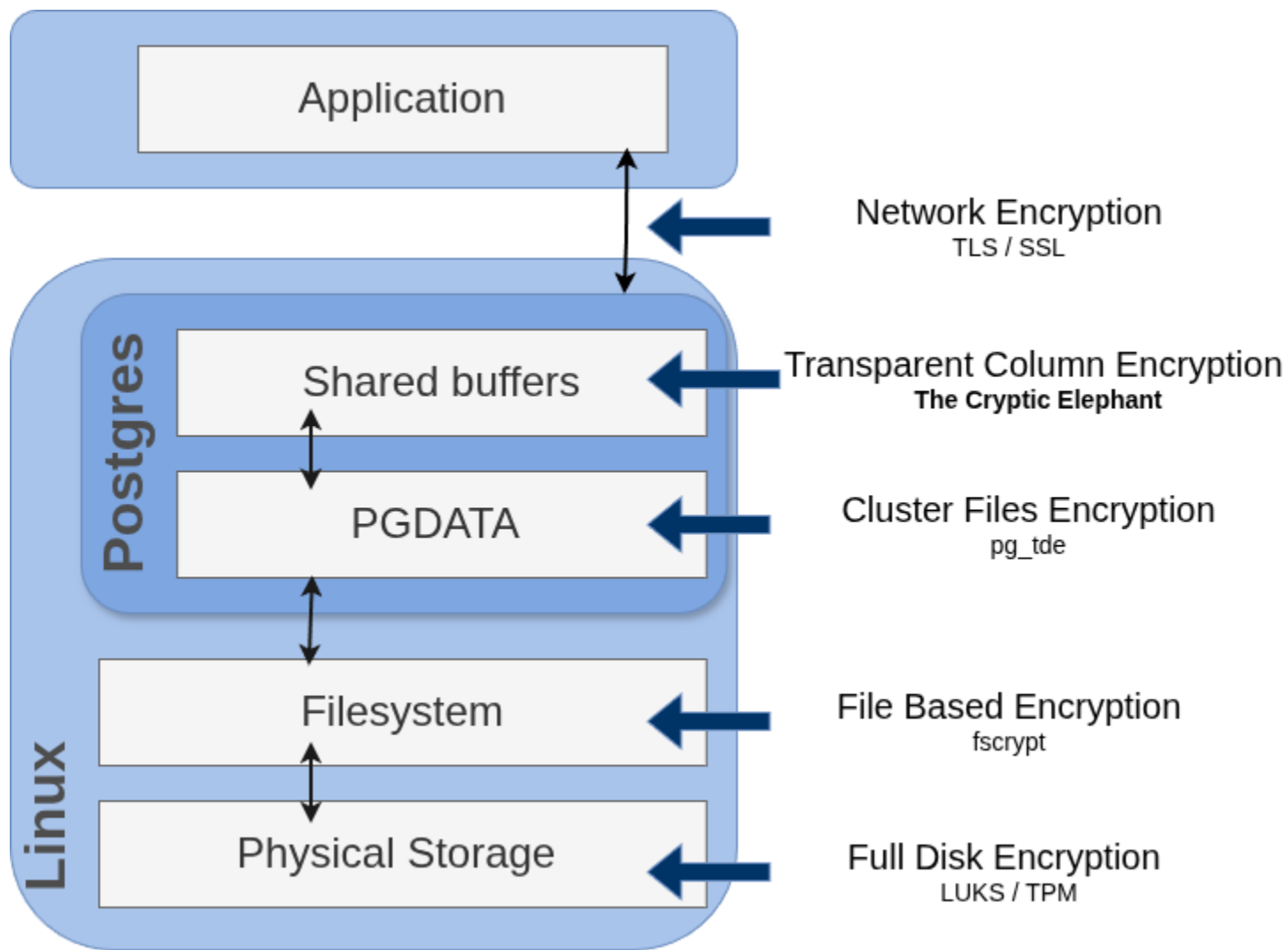
data at rest



data in use



data in transit



Notre Approche

TCE plutôt que TDE / CFE

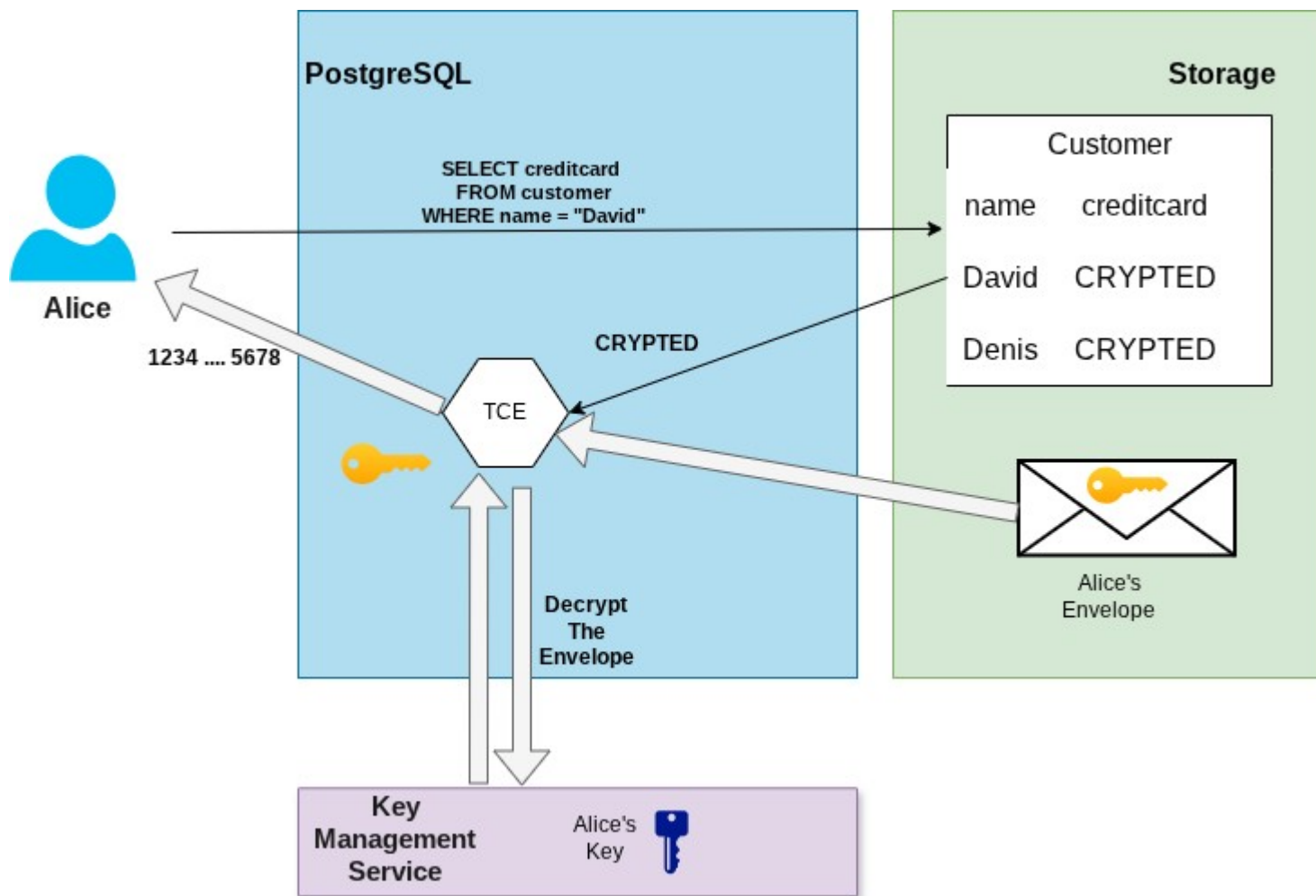
Une extension plutôt qu'un Fork

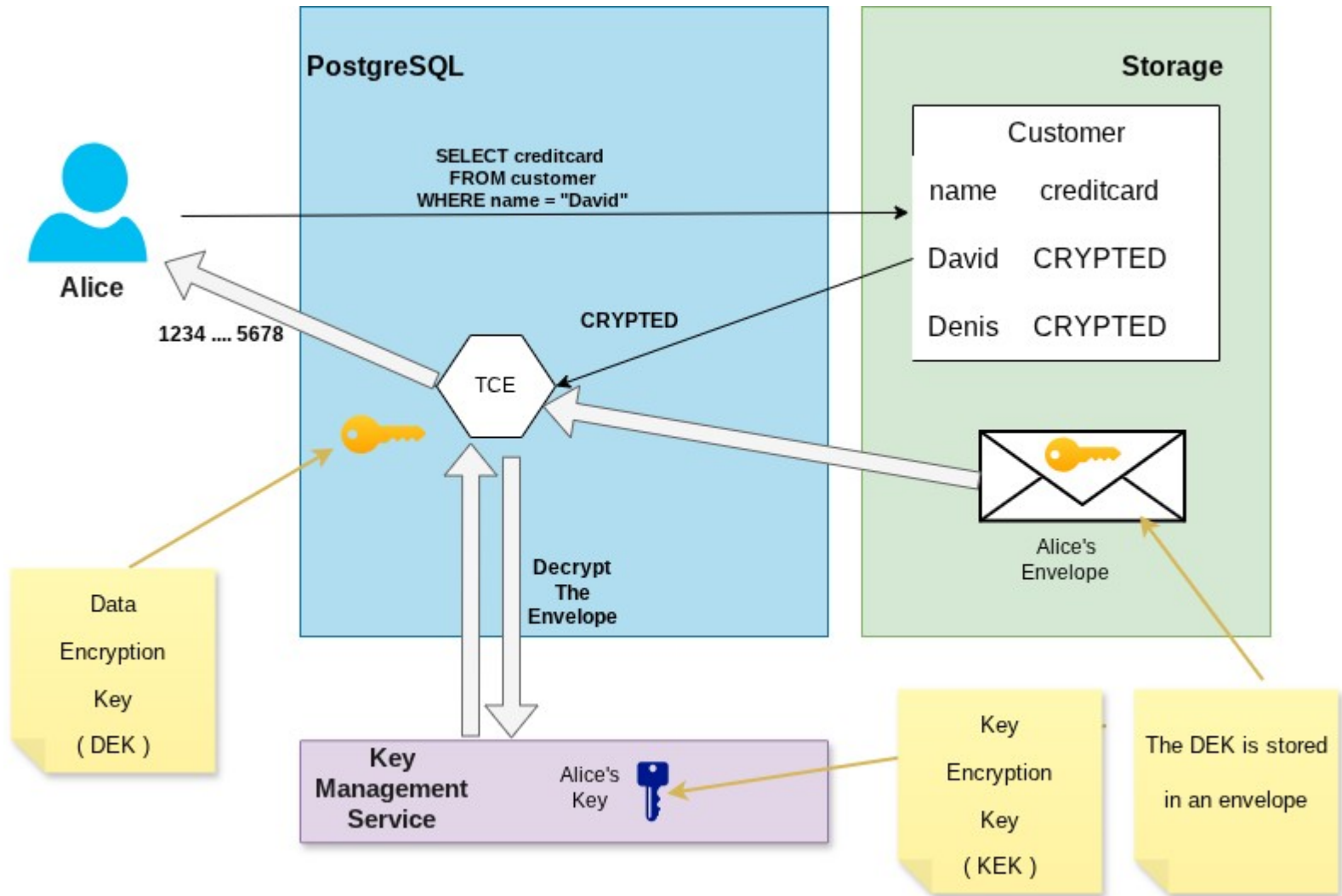
Compatible PostgreSQL 14 à 19

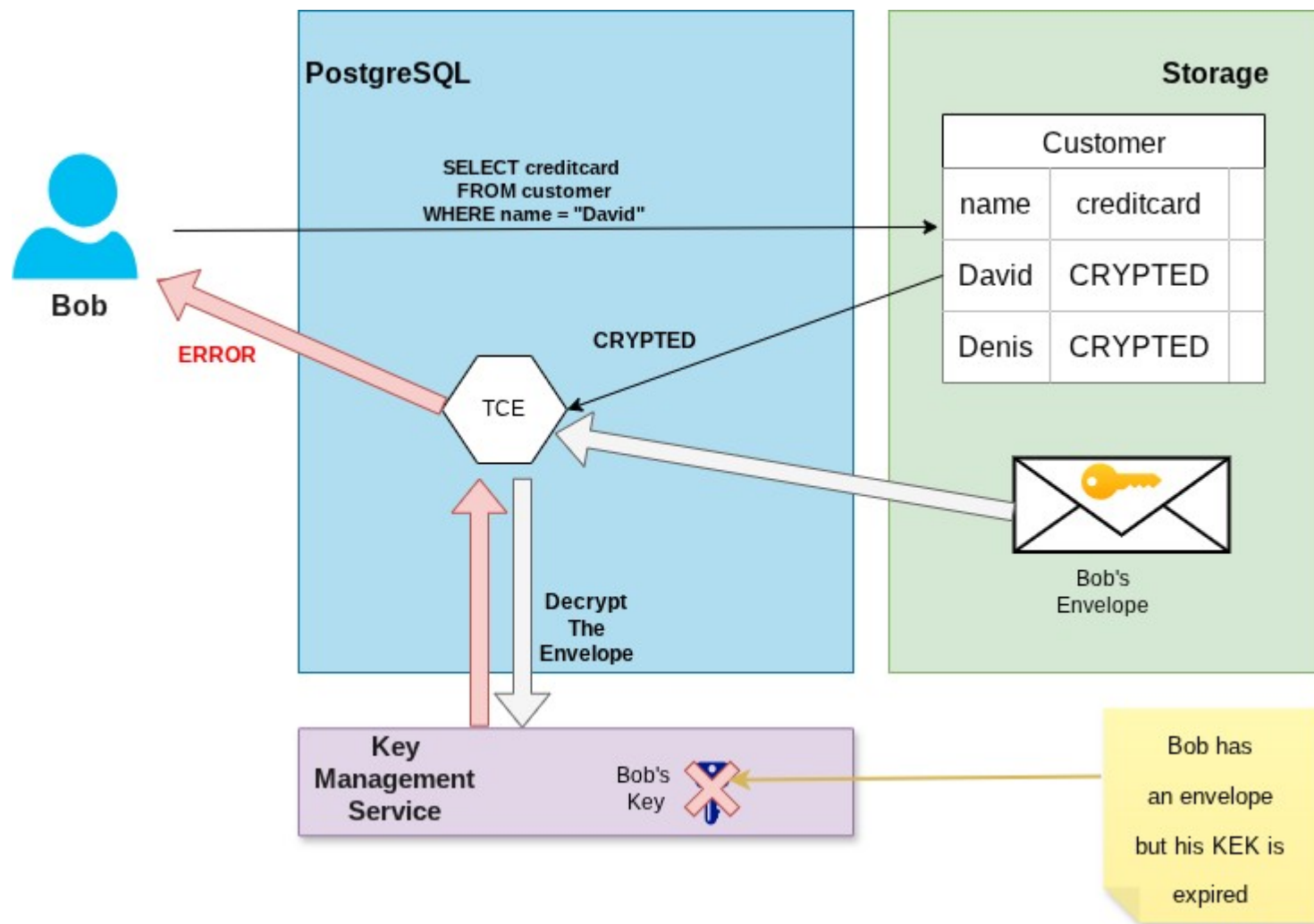
Licence PostgreSQL (pas de version open core)

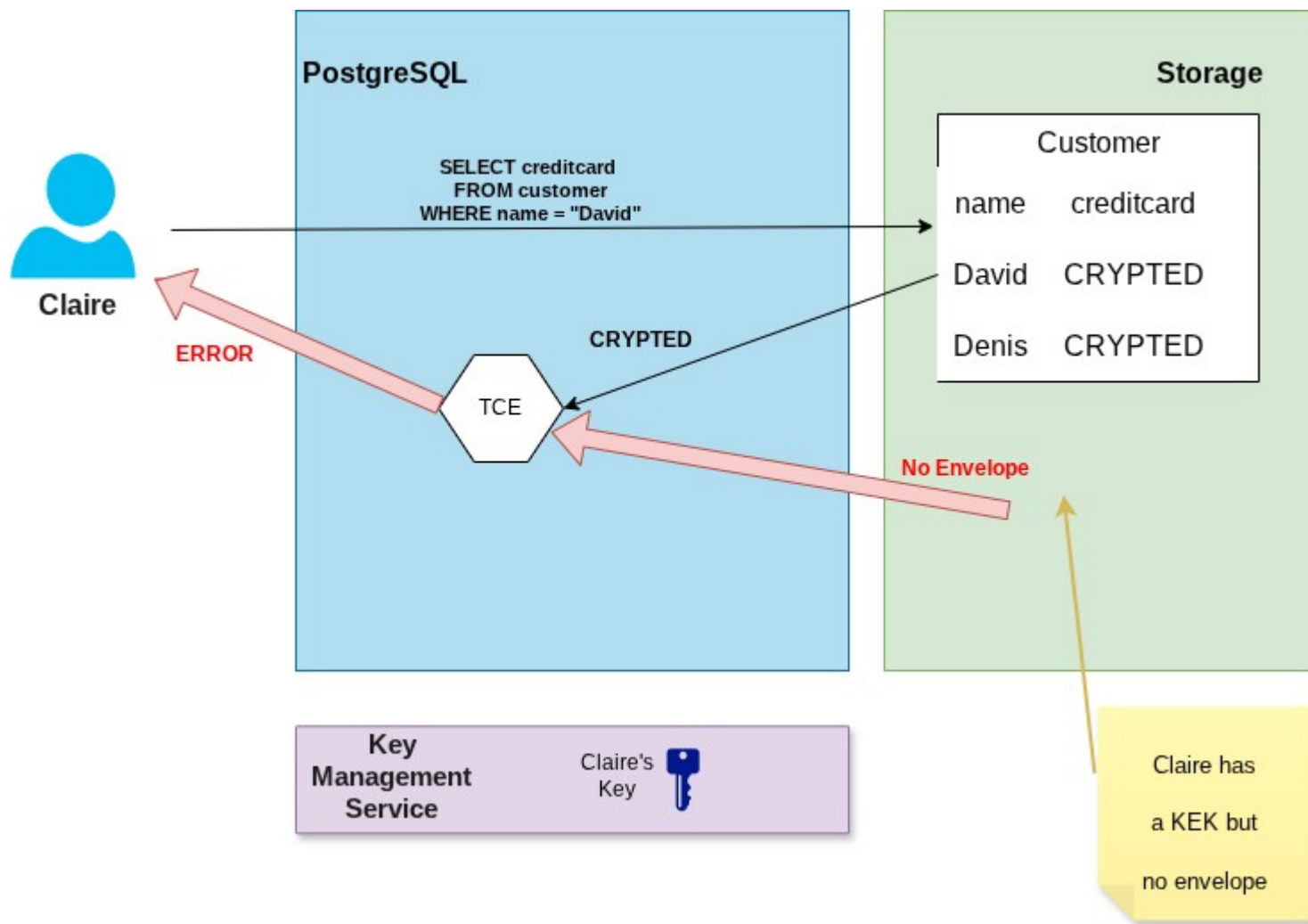
Intégration avec les KMS du marché
(Hashicorp Vault, Thalès CipherTrust, KMIP)

Implémentation en Rust avec des libs crypto auditées









Exemple

```
CREATE EXTENSION tce;
```

```
ALTER DATABASE demo SET tce.kms = aws;
```

```
CREATE ROLE alice LOGIN;
```

```
SECURITY LABEL FOR tce_user_key ON ROLE alice  
IS 'HOSTED BY KMS WITH ID = 98b797ed-e462-45ef-bc6a-4d1dc2434915';
```

Example

```
CREATE TABLE customer(  
    realname TEXT,  
    creditcard TEXT  
);  
  
INSERT INTO customer  
VALUES  
    ('James Bond', '0007 0007 0007 0007'),  
    ('Mata Hari', '1234 5678 1234 5678');
```

Exemple

```
ALTER TABLE customer  
  ALTER COLUMN creditcard  
    TYPE TCETEXT  
  USING creditcard::TCETEXT;
```

Exemple

```
SELECT *, raw(credit_card) as raw_value from person;
```

id	firstname	credit_card	raw_value
1	Bernard	1234567890	TCE:229:132:89:15:181:57:201:...
2	Bob	1234567890	TCE:53:68:180:197:138:99:200:...

Example

```
\connect demo alice
```

```
SELECT realname, creditcard FROM customer;
```

realname		creditcard
-----+-----		
James Bond		0007 0007 0007 0007
Mata Hari		1234 5678 1234 5678

Exemple

```
\connect demo bob
```

```
SELECT realname, creditcard FROM customer;
```

realname		creditcard
James Bond		TCE:229:132:89:15:...
Mata Hari		TCE:53:68:180:197:...

Feuille de Route

Développement actif en interne

Première version publique début 2026

Version bêta au second semestre 2026

Version 1.0 en 2027

Intégré dans le Socle Dalibo Postgres 19

Fonctionnalités pour la V1

- Chiffrement Chacha20Poly1350 + RSA
- Support de Amazon KMS
- Support de Hashicorp Vault / Open Bao
- Support de Thales CypherTrust
- Support de OVH KMS
- Support de Alterway KMS
- Protocole KMIP
- Chiffrement Post Quantum
- Enclave Mémoire
- Chiffrement des types: TEXT, INT, DATE , BYTEA

Comment nous aider ?

Nous finançons ce projet “en mode pur open source”

Pas de retour sur investissement direct

Vous pouvez contribuer via un programme de soutien !

Development Partner Program

Un programme de co-développement innovant

3 niveaux possibles : STARTER / STANDARD / PREMIUM

Des comités de pilotage réguliers

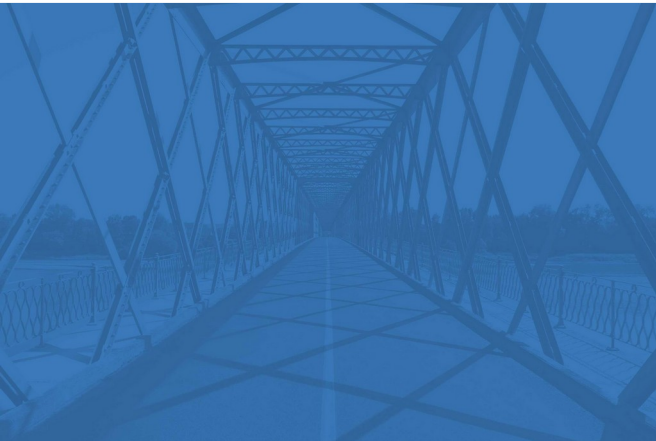
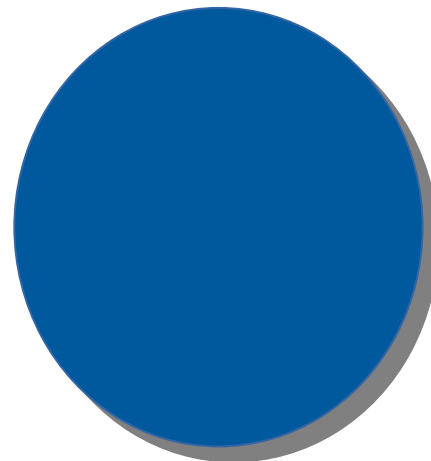
Des Cycles de Release très courts

Possibilité de tester au fil de l'eau

Copyright Partagé et Protégé

Damien CLOCHARD

Lead Developer
damien.clochard@dalibo.com



Notre prochain contact ?

FAQ : Et les index ?

Est-ce vraiment nécessaire ?

Chiffrement au choix : volatile ou immutable

Immutable est moins sécurisé

Volatile n'est pas indexable

FAQ : Chiffrer toute une table ?

C'est possible théoriquement

Est-ce un besoin réel ?

FAQ : Ce n'est pas tout à fait transparent

Oui

Il faut changer le type : TEXT > TCETEXT

C'est plus explicite

Mais pas toujours possible...

On y travaille !

FAQ : Pourquoi Rust ?

Les extensions en C sont plutôt dangereuses

L'écosystème Rust Crypto est très riche

Nous avons l'expérience concluante de PostgreSQL Anonymizer

FAQ : Performances ?

Probablement moins bonne que FDE

MAIS

On chiffre uniquement les données confidentielles
c'est donc beaucoup "économique"