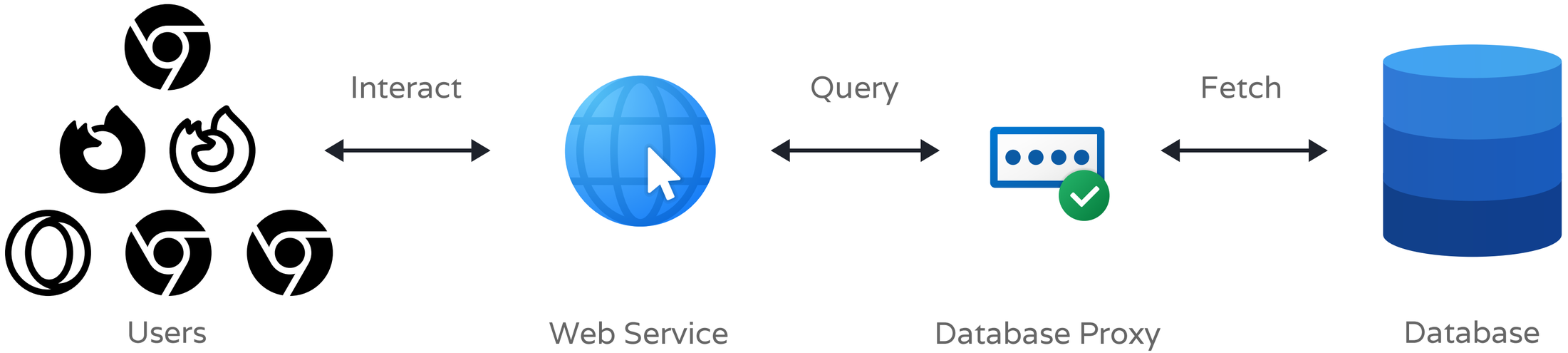
CryptDB

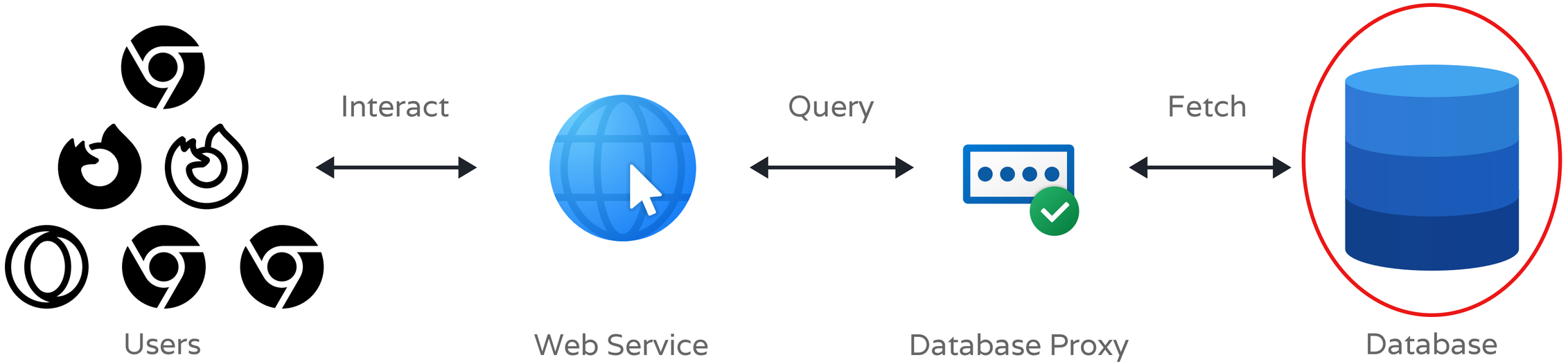
Can we secure user privacy at the database layer?

Presenter

AKSHAT MAHAJAN

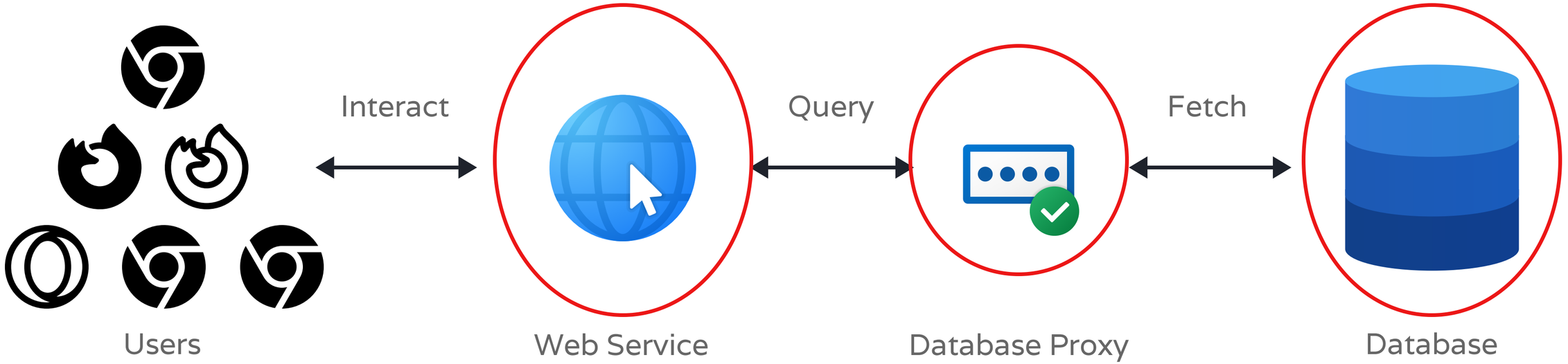


A web app under the hood



Threat 1: Stealing directly from the database

Gaining direct access to the database means you can run any query you like



Threat 2: Compromising everything

An attacker who sees all and knows all can get away with anything (deletion, reading, etc.)

Let’s encrypt the database, and continue as normal!

(insufficient for our threat models - can you articulate why?)

Encryption brings problems

INCOMPATIBLE WITH TRADITIONAL TECH

Database queries don’t work over ciphertext

INEFFICIENT FOR END USERS

Performing decryption over ciphertext slows queries down

MEANINGLESS IF KEYS ARE UNPROTECTED

If you can steal encryption keys, no encryption can save you

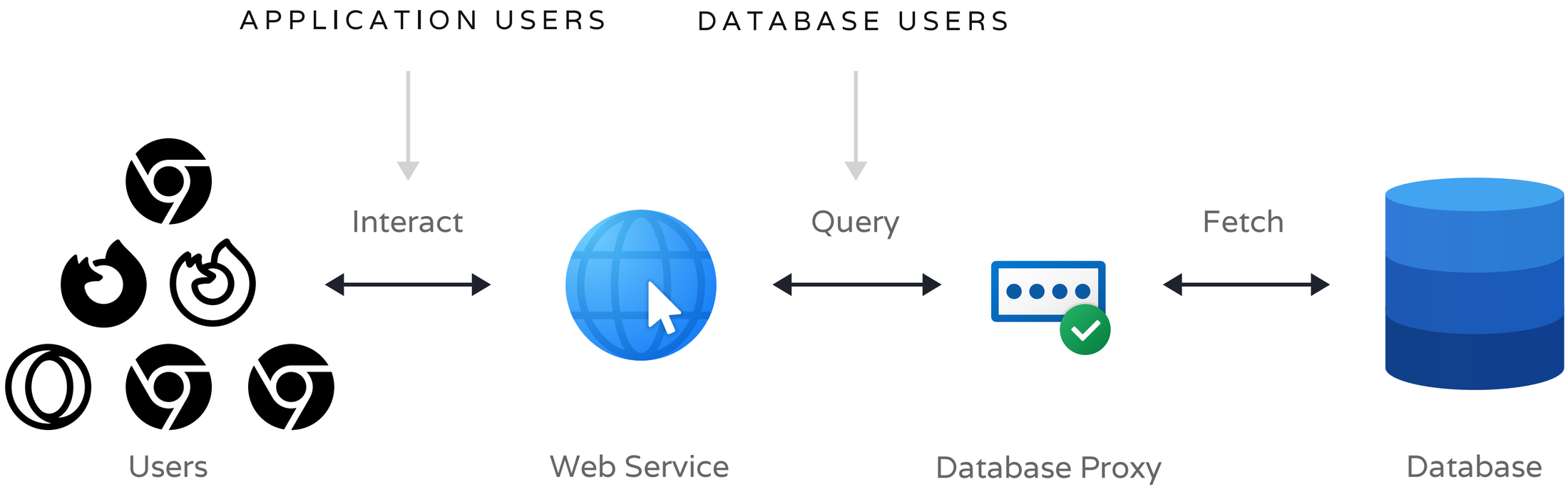
Let’s make ciphertext accessible to queries

(more relevant to performance)

Let’s use keys only the end user can access

(more relevant to privacy)

Distinguishing Between Users (*Principals*)



Database users connect directly to the proxy; application users rely on the application to connect for them.
An application logs in as a database user on behalf of application users.

Simplified Definition of Logged-In

Active Users Table

Alice (database user)	<decryption key encrypted with Alice's password>
Bob (application user through Alice)	<<decryption key encrypted with Bob's password> with Alice's password>

Permissions column for another table with Bob's data:	<value encrypted with Bob's decryption key>

Database users are automatically added to active users on new connection, but must add / remove application users manually. Decryption keys are renegotiated every new connection. When user is not active, permissions are encrypted with a master key.

Exercise

Let's attack this system so far:

- **Would being a man-in-the-middle help?**
- **How much data can you access with just database access?**
- **How much can you access with complete control over application, proxy and database?**

```
PRINCTYPE physical_user EXTERNAL;
PRINCTYPE user, group, forum_post, forum_name;

CREATE TABLE users       ( userid int, username varchar(255),
  (username physical_user) SPEAKS_FOR (userid user) );

CREATE TABLE usergroup ( userid int, groupid int,
  (userid user) SPEAKS_FOR (groupid group) );

CREATE TABLE aclgroups ( groupid int, forumid int, optionid int,
  (groupid group) SPEAKS_FOR (forumid forum_post)
        IF optionid=20,
  (groupid group) SPEAKS_FOR (forumid forum_name)
        IF optionid=14);

CREATE TABLE posts       ( postid int, forumid int,
  post text ENC_FOR (forumid forum_post) );

CREATE TABLE forum       ( forumid int,
  name varchar(255) ENC_FOR (forumid forum_name) );
```

```
PRINCTYPE physical_user EXTERNAL;
PRINCTYPE contact, review;

CREATE TABLE ContactInfo    ( contactId int, email varchar(120),
  (email physical_user) SPEAKS_FOR (contactId contact) );

CREATE TABLE PCMember     ( contactId int );
CREATE TABLE PaperConflict ( paperId int, contactId int );
CREATE TABLE PaperReview  (
  paperId          int,
  reviewerId       int  ENC_FOR (paperId review),
  commentsToPC text ENC_FOR (paperId review),
  (PCMember.contactId contact) SPEAKS_FOR
        (paperId review) IF NoConflict(paperId, contactId) );

NoConflict (paperId, contactId):     /* Define a SQL function */
  (SELECT COUNT(*) FROM PaperConflict c WHERE
      c.paperId = paperId AND c.contactId = contactId) = 0;
```

Policy Annotations

A USER CAN INHERIT PERMISSIONS FROM A GROUP

CONDITIONS CAN BE PLACED ON ANY USER'S ACCESS

Revealing little while being quick: partially homomorphic encryption

RND

No special properties

DET

Equality

HOM

Addition

OPE

Sorting

SEARCH

Substring matching

OPE-JOIN

Range-based table joins

JOIN-ADJ

Equality-based table joins

EVOLUTION OF ADAPTIVE ENCRYPTION OVER QUERIES

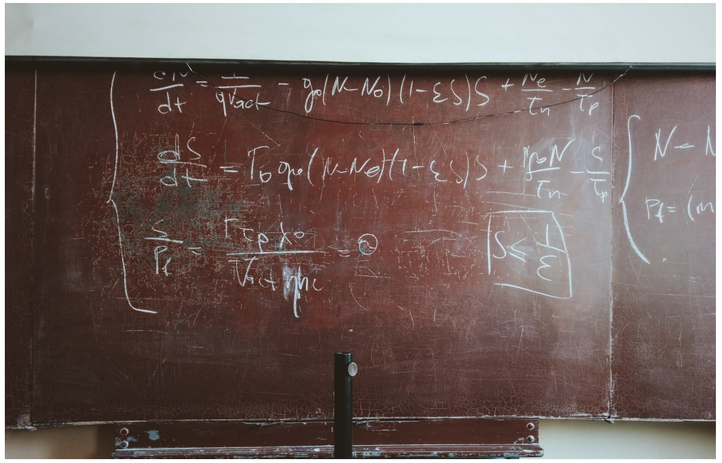
<ARE YOU READY, KIDS?>

DEMO TIME!

WHERE DO ENCRPYTED DATABASES GO FROM HERE?

Onwards

WHAT NEEDS TO HAPPEN BEFORE ENCRYPTED DATABASES ARE NORMAL?



Homomorphic Encryption

Support arbitrary computation over data, not just specific kinds

Production Readiness

Support more real-world data types with partially homomorphic algorithms

Ensuring k-anonymity

Ensuring data is robust against inference

SQL Server offers RND and DET, but...

- Columns using one of the following data types: `xml`, `timestamp`/`rowversion`, `image`, `ntext`, `text`, `sql_variant`, `hierarchyid`, `geography`, `geometry`, alias, user defined-types.
- `FILESTREAM` columns
- Columns with the `IDENTITY` property.
- Columns with `ROWGUIDCOL` property.
- String (`varchar`, `char`, etc.) columns with non-bin2 collations.
- Columns that are keys for clustered and nonclustered indices when using randomized encryption (deterministic encryption is supported).
- Columns included in full-text indexes (Always Encrypted does not support Full Text Search).
- Computed columns.
- Columns referenced by computed columns (when the expression does unsupported operations for Always Encrypted).
- Sparse column set.
- Columns that are referenced by statistics when using randomized encryption (deterministic encryption is supported).
- Columns using alias types.
- Partitioning columns.
- Columns with default constraints.
- Columns referenced by unique constraints when using randomized encryption (deterministic encryption is supported).
- Primary key columns when using randomized encryption (deterministic encryption is supported).
- Referencing columns in foreign key constraints when using randomized encryption or when using deterministic encryption, if the referenced and referencing columns use different keys or algorithms.

A PARTIAL LIST OF COLUMN TYPES IT CANNOT SUPPORT

Inference Attacks (Naveed et. al., 2015)

Concrete attacks. We study the effectiveness of four different attacks. Two are well-known and two are new:

- *frequency analysis:* is a well-known attack that decrypts DTE-encrypted columns given an auxiliary dataset that is "well-correlated" with the plaintext column. The extent of the correlation needed, however, is not significant and many publicly-available datasets can be used to attack various kinds of encrypted columns with this attack.
- ℓ_p*-optimization:* is a new *family* of attacks we introduce that decrypts DTE-encrypted columns. The family is parameterized by the ℓ_p-norms and is based on combinatorial optimization techniques.
- *sorting attack:* is an attack that decrypts OPE-encrypted columns. This folklore attack is very simple but, as we show, very powerful in practice. It is applicable to columns that are "dense" in the sense that every element of the message space appears in the encrypted column. While this may seem like a relatively strong assumption, we show that it holds for many real-world datasets.
- *cumulative attack:* is a new attack we introduce that decrypts OPE-encrypted columns. This attack is applicable even to low-density columns and also makes use of combinatorial optimization techniques.

- ℓ_2*-optimization* (vs. DTE-encrypted columns): the attack recovered the mortality risk and patient death attributes for 100% of the patients for at least 99% of the 200 largest hospitals. It recovered the disease severity for 100% of the patients for at least 51% of those same hospitals.
- *frequency analysis* (vs. DTE-encrypted columns): the attack had the same results as ℓ_2-optimization.
- *sorting attack* (vs. OPE-encrypted columns): the attack recovered the admission month and mortality risk of 100% of patients for at least 90% of the 200 largest hospitals.
- *cumulative attack* (vs. OPE-encrypted columns): the attack recovered disease severity, mortality risk, age, length of stay, admission month, and admission type of at least 80% of the patients for at least 95% of the largest 200 hospitals. For 200 small hospitals, the attack recovered admission month, disease severity, and mortality risk for 100% of the patients for at least 99.5% of the hospitals.

MILESTONES IN FULLY HOMOMORPHIC ENCRYPTION

2008

IBM’s Craig Gentry provides the first working algorithm

2011

Second-generation FHE algorithms

2013

Third-generation FHE algorithms

2016

TFHE, fastest FHE library, released to the public

CryptDB: The End

Can we secure user privacy at the database layer? Maybe.

Discussion Questions

CAN INFORMATION FLOW CONTROL WORK WELL WITH RESIN? E.G. RESIN TOGETHER WITH CRYPTDB? - MARY

DOES CRYPTDB-SUPPORTED AGGREGATION POSE A PRIVACY LEAK BY CONVEYING IMPLICIT INFORMATION? - SAM THOMAS

WHAT ASPECTS OF THE GDPR WOULD CRYPTDB BE A POOR FIT FOR?

CAN ONE BUILD AN ENCRYPTED DATABASE THAT GUARDS AGAINST LOSS OF DATA INTEGRITY? (E.G. ACCIDENTAL DATA DELETES)