Don’t trust us, trust the math behind **immudb**

Bartłomiej Święcki

February 5th, 2022, FOSDEM

Bartłomiej Święcki

Passionate about programming, open source, Linux, cryptography

Programmer, DevOp, family guy, runner....

Since June 2021
senior Codenotary team member and core immudb developer

Codenotary

Startup with focus on trusted CI/CD, DevSecOps

Secure Supply Chain and monitoring

Community Attestation Service
https://cas.codenotary.com/

Immudb - a solid technological foundation

Immudb

Open source database written in go
https://github.com/codenotary/immudb/

Built from ground-up around immutability and tamper resistance

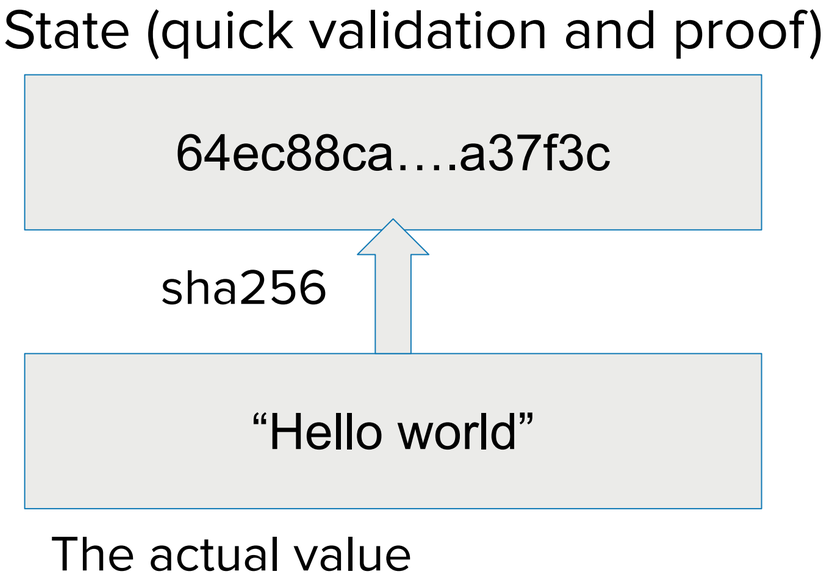
Unique features - data proofs, time travel, data expiration, direct s3 storage support and more

Solid, secure foundation for other solutions created at Codenotary

Simplified system with just one value

Cryptographic hash function:

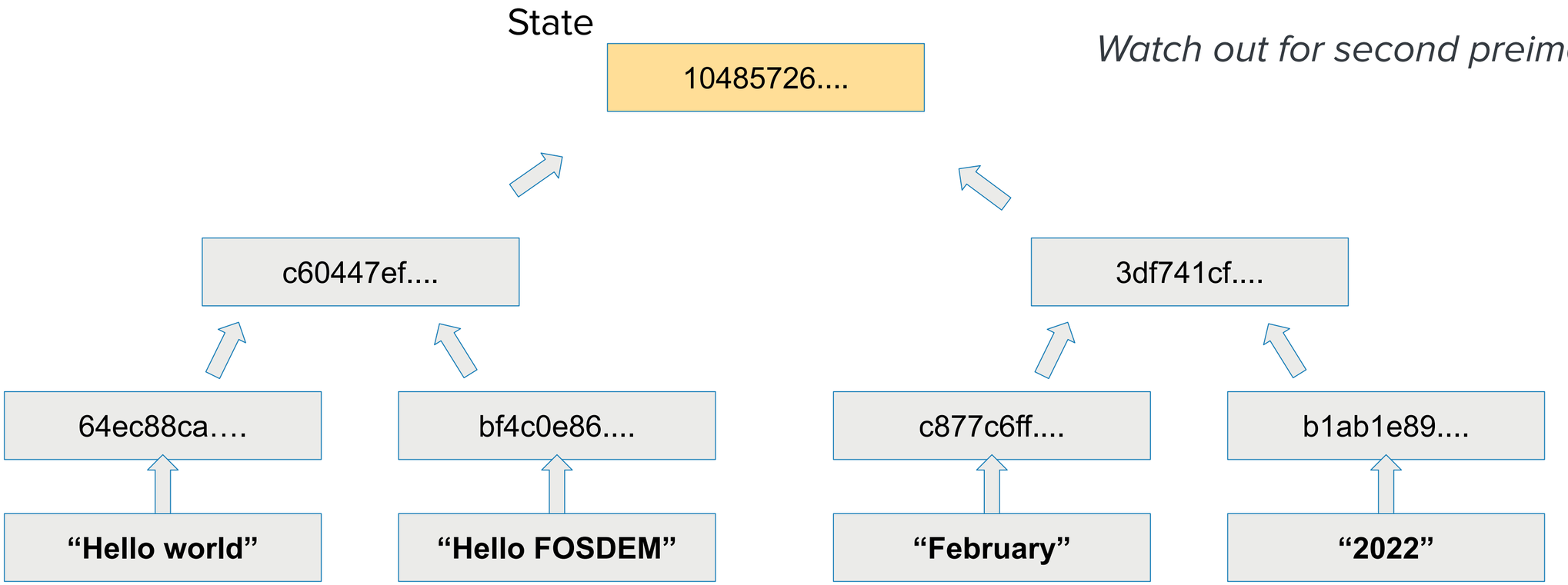
- Hash does not reveal any information about the original value
- Can not be reversed - can’t compute a value form only a hash



Let’s add more values to the picture

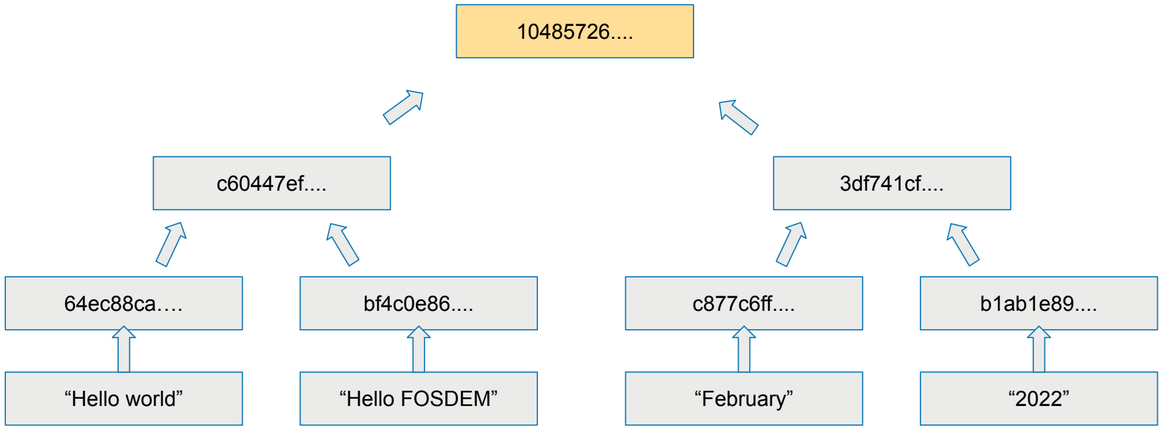
Merkle Tree

Watch out for second preimage attack



Merkle tree - interesting properties

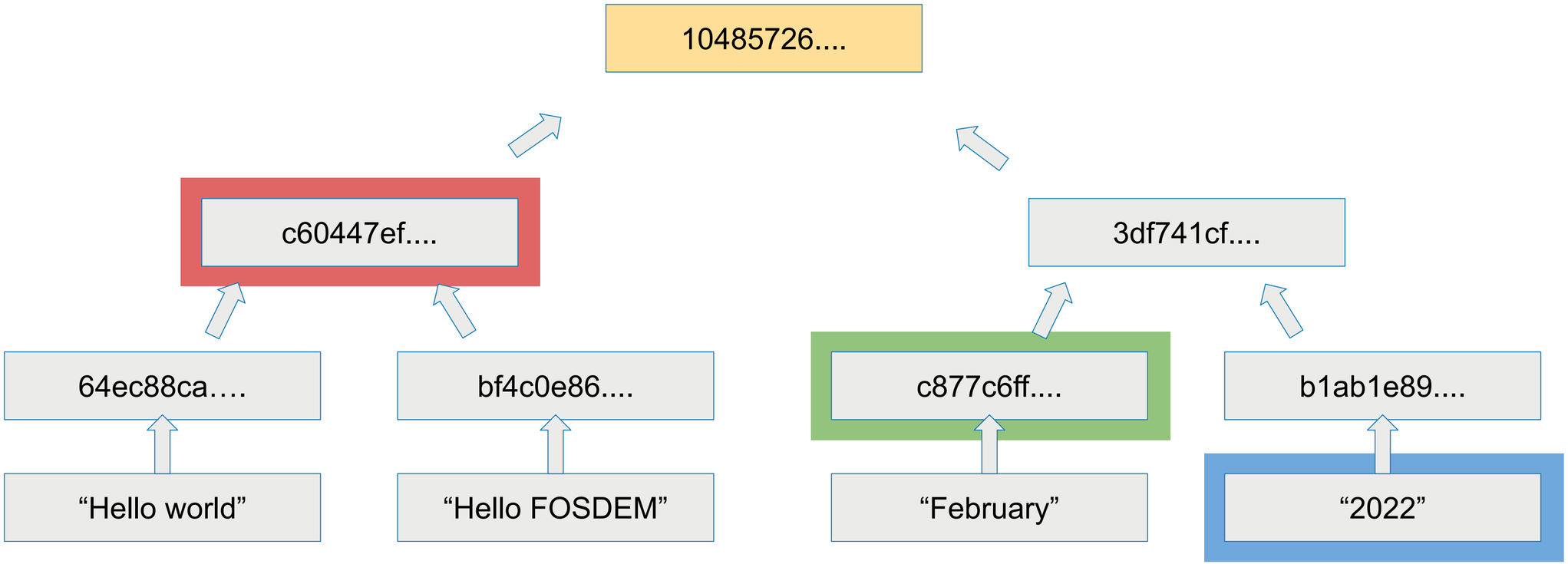
- Each value equally contributes to the root hash
- Values are ordered
- Changing any value will modify the root hash
- Can not come up with a fake tree with same root digest but using different leaf values*
- If a system claims given root hash, it can not deny existence of any of its leaf values



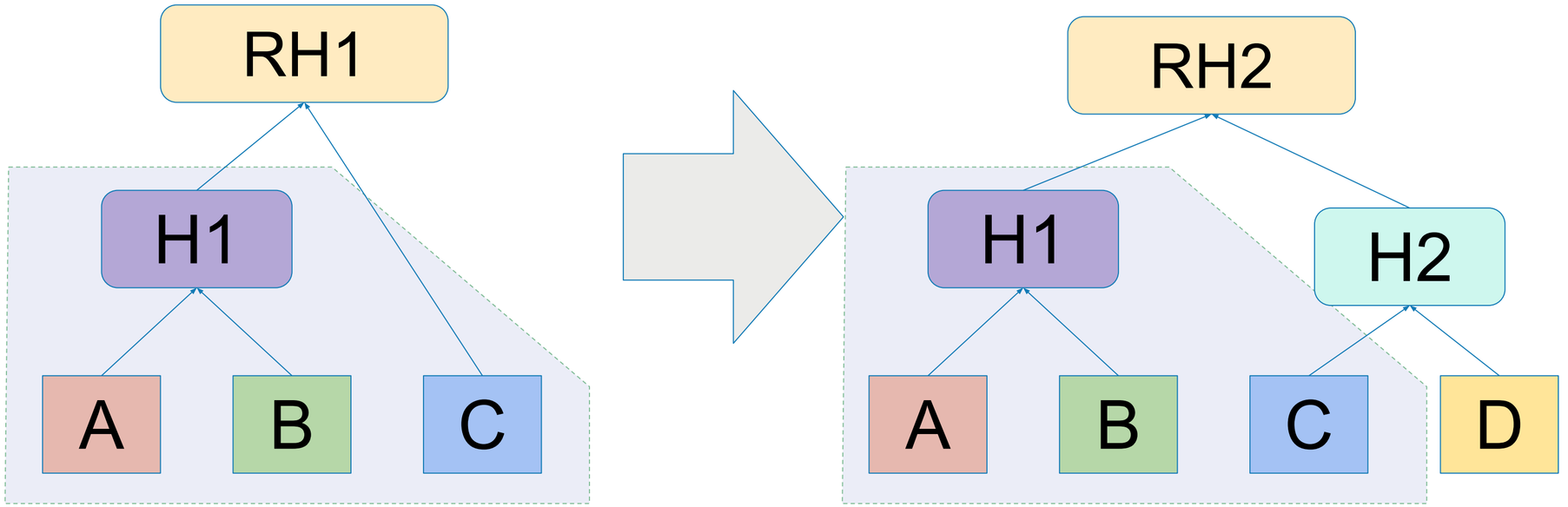
* if protection against second preimage attack was added

CRYPTOGRAPHIC HASH TREES

DB: It’s 2022
Me: prove it to me!
DB: sha256(`c60447ef....` + sha256(`c877c6ff....` + sha256(“2022”))) == `10485726...`
Me: where did you get those values from?
DB: my internal Merkle Tree, is there anything wrong with that?
Me: no, it’s fine.... this time...



AHT - we can only append values at the end of the tree



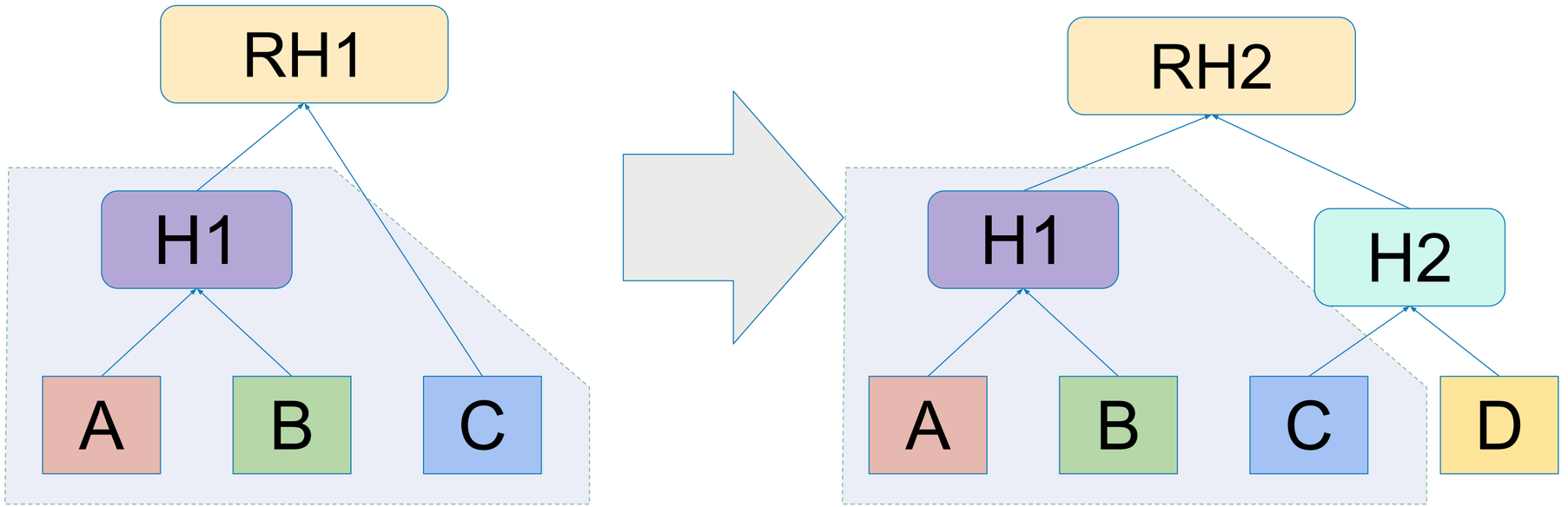
APPENDABLE HASH TREES

DB: Current state is RH2

Me: I know that your previous state was RH1, I need a proof that you did not discard old data

DB: sha256(H1 | C) == RH1, sha256(H1 | sha256(C | D)) == RH2

Me: wait, wait, need to think about this....



APPENDABLE HASH T

It's like calculating root hashes from scratch but reusing common subtrees

DB: Current state is RH2

Me sha256(H1 | C) == RH1, sha256(H1 | sha(C | D)) == RH2

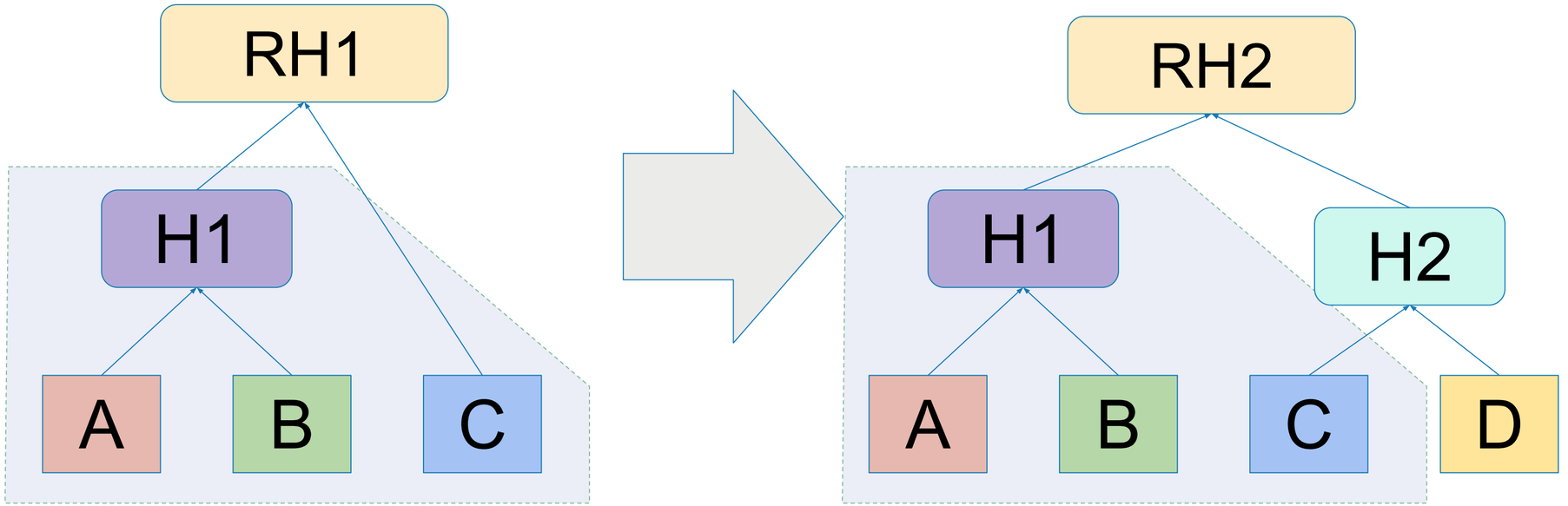
DB

Me

RH1 == sha256(sha256(A | B) | C) == sha256(H1 | C)

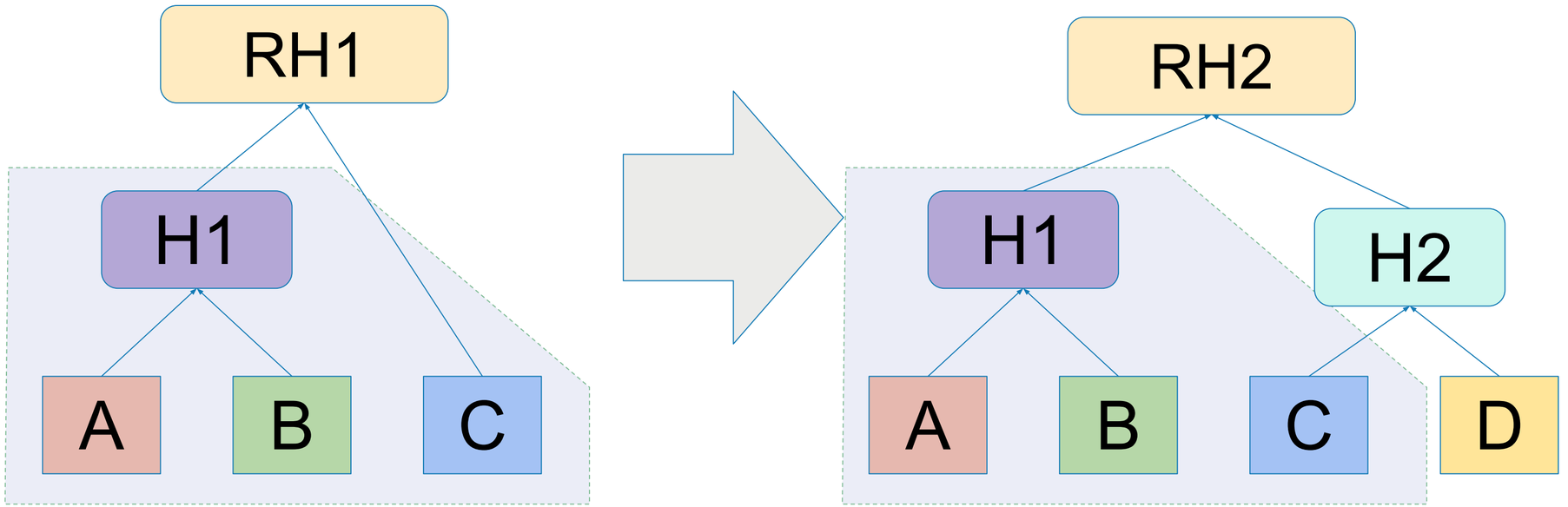
H1

RH2 == sha256(sha256(A | B) | sha256(C | D)) == sha256(H1 | sha(C | D))

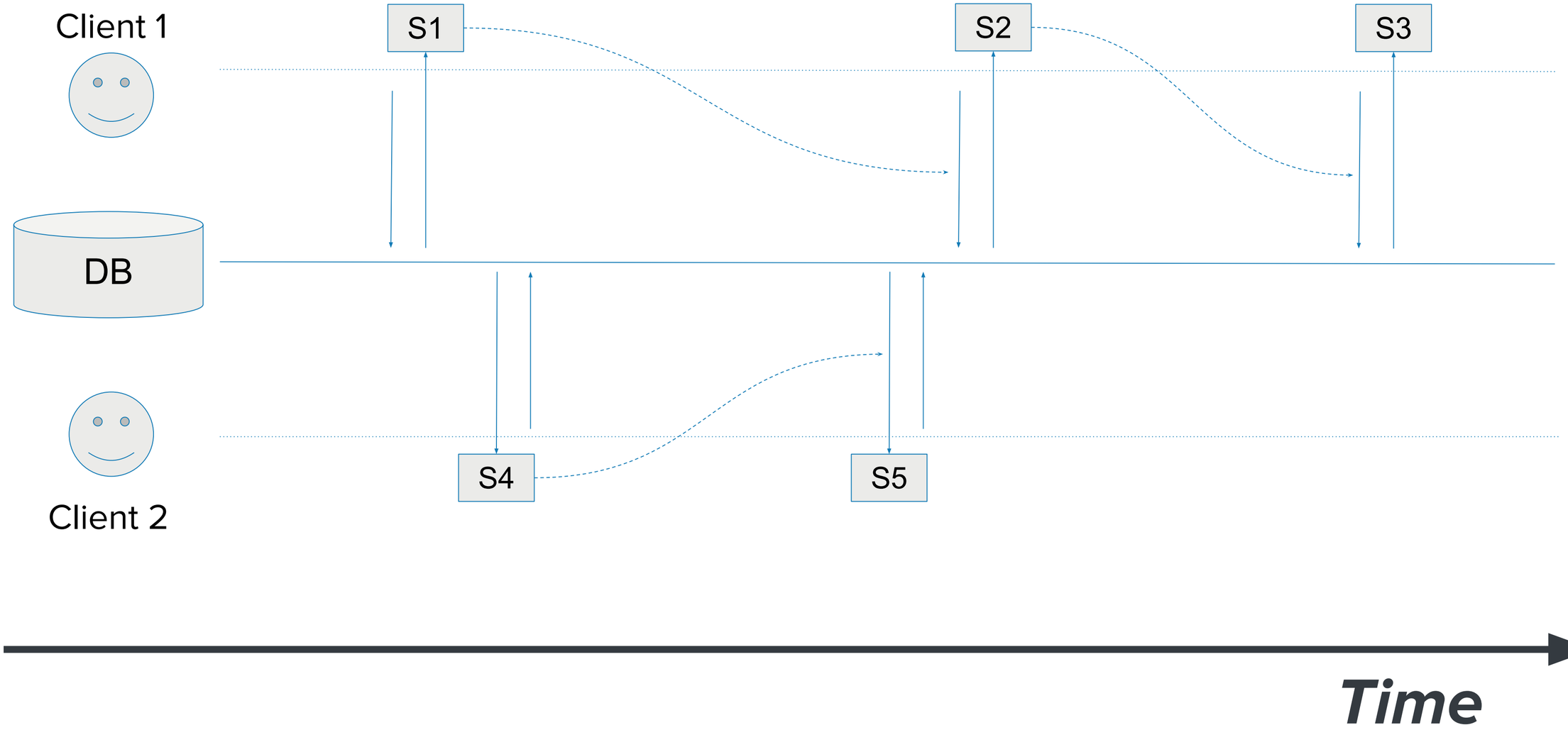


APPENDABLE HASH TREES

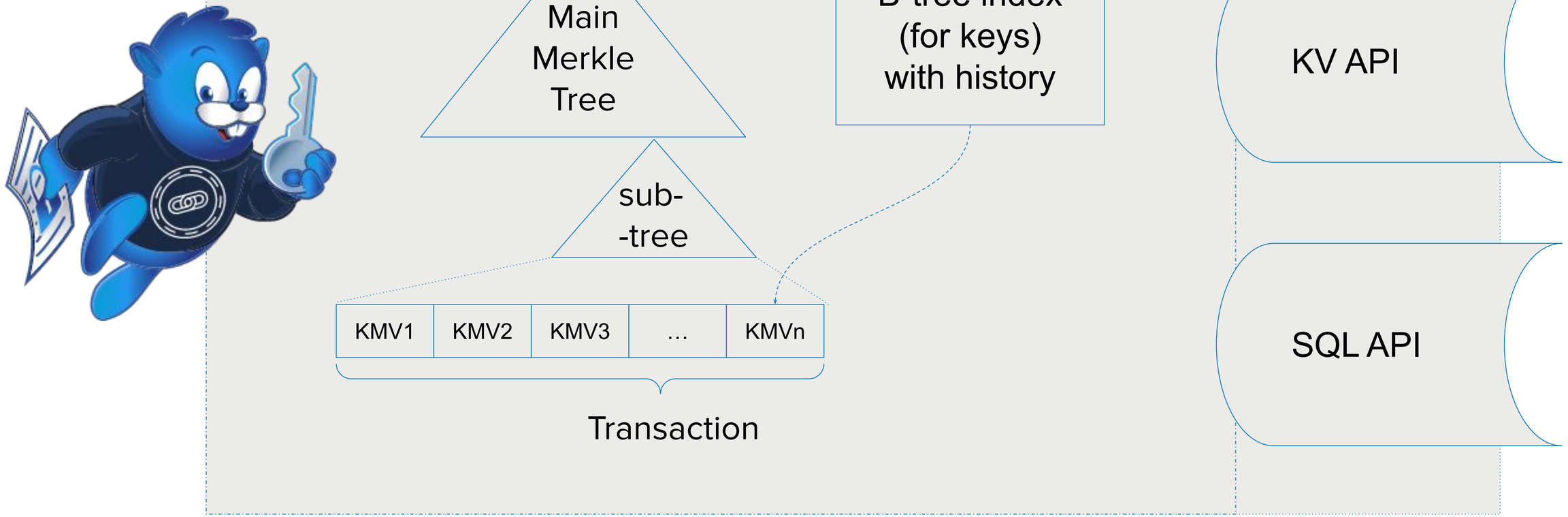
DB: Current state is RH2
Me: I know that your previous state was RH1, I need a proof that you did not discard old data
DB: sha(H1 | C) == RH1, sha(H1 | sha(C | D)) == RH2
Me: wait, wait, need to think about this….
Me: … ok, all clear now, now I know you're trustworthy, but I'll check once again the next time



DB WITH CRYPTOGRAPHIC PROOFS



immudb under the hood (a single database)



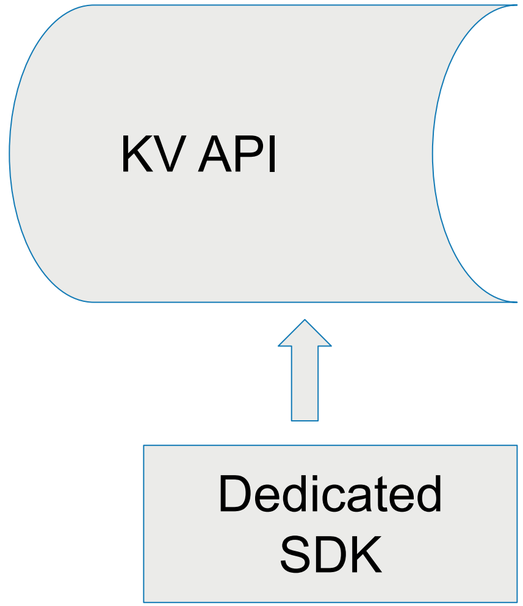
immudb - KV API

Low overhead

- Very fast (close to low-level disk ops)
- Stable
- Battle-tested (used at Codenotary projects, Fintech, Government, community projects)
- Low-level KV API may be a bit hard to work with

Functionalities

- Simple Get / Set / Scan operations
- References to other keys (including historical values)
- Sorted sets
- History of a key
- Verified operations



immudb - SQL API

Very robust

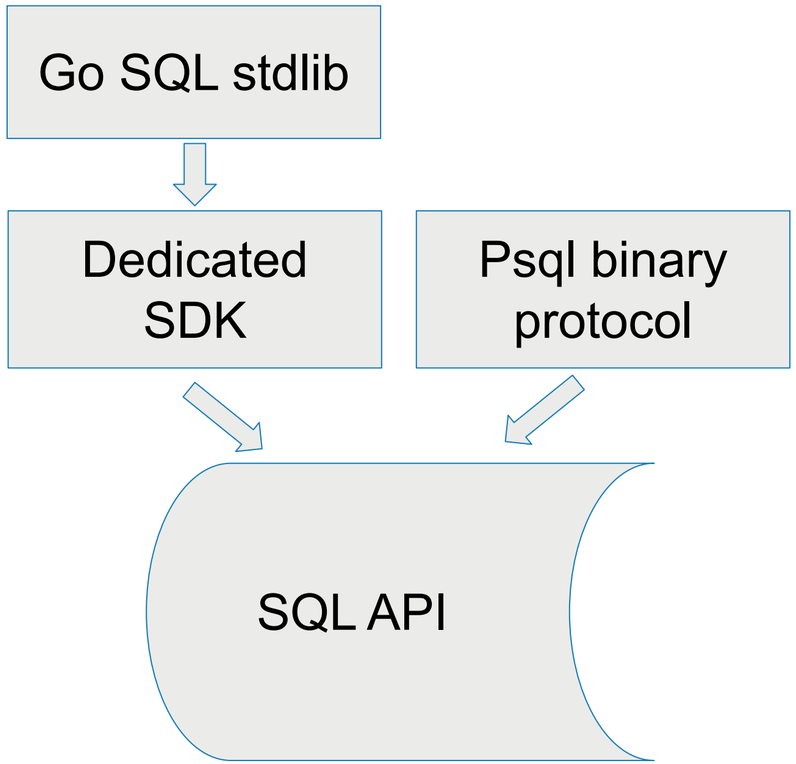
- tables
- indexes
- typed columns
- null values

Actively worked on

- May still be a bit unstable
- Some features missing (e.g. alter table, foreign keys)

SQL dialect support:

- INSERT
- UPDATE
- UPSERT
- DELETE
- SELECT (with JOINs)



Immudb - planned features

Physical deletion of data

- Free disk space once old data is no longer needed (without breaking proofs)

SQL improvements:

- ALTER TABLE
- CREATE INDEX on table with data
- More data types
- expose DB schema in SQL
- support common SQL/ORM libraries (e.g. gorm)
- Improved psql compatibility

Improved transactions:

- Improve transaction API on KV level
- Allow working with both KV and SQL within a single transaction

Performance and scalability improvements

Goal: use immudb as a primary database

*DEMO**

*almost live ;)

immudb - where to use?

Best as a content-specific (e.g. as a document storage) or secondary database protecting most crucial parts of the system

- Important data that must not be tampered with (verification)
- Data that needs full history of all changes (time travel)
- Reliable ledgers and audit trails

May not be good for

- Frequently changing data such as counters (each update of a value is in fact a new entry)
- Temporary data (no physical deletion of records yet, planned)
- Changing SQL schemas (no schema changes yet, in progress)

immudb - how to use?

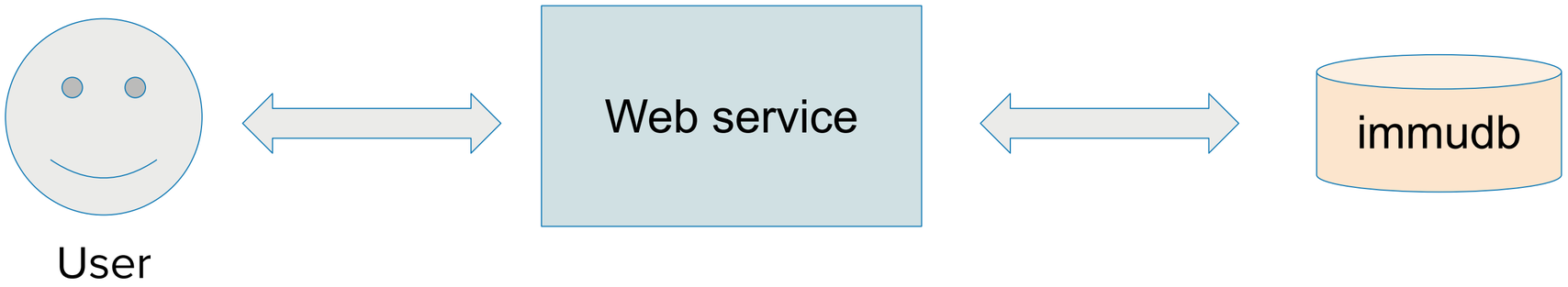
Internal state of immudb must be validated by some external entity

Validation is done when performing a verified operation through SDK or by running dedicated auditor process (immudb in audit mode)

Depending on required security level, it can be implemented in multiple ways

immudb - how to use?

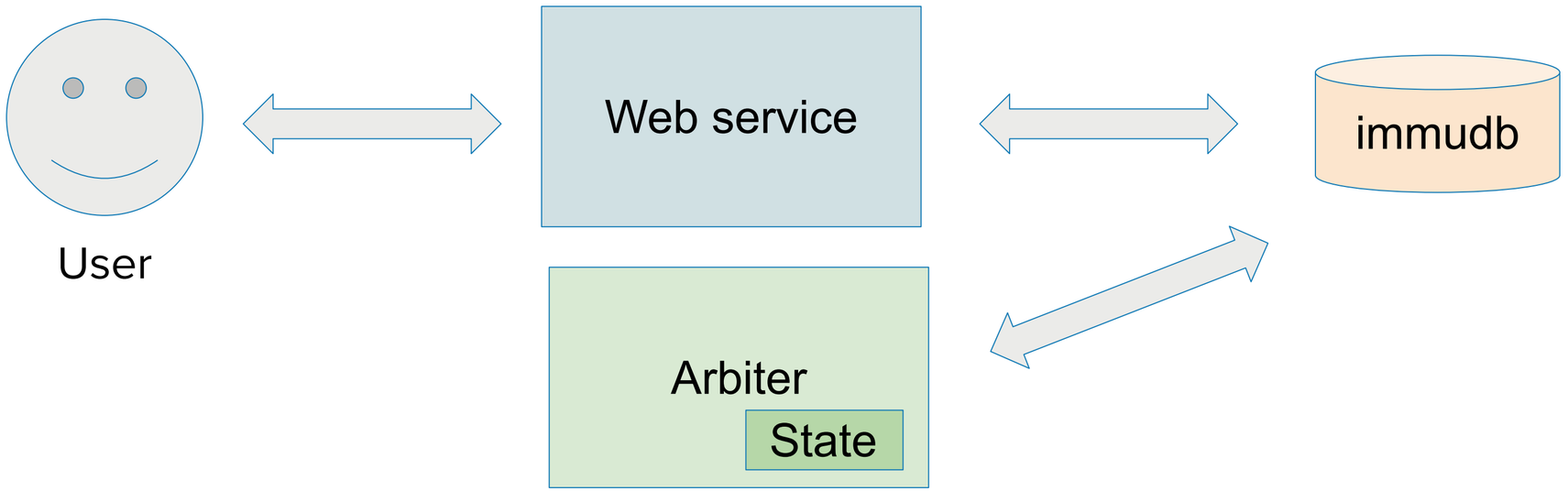
1. No verification - just use immudb as a normal database



Even though the state is not validated, a separated immudb installation will still be harder to attack than other databases

immudb - how to use?

2. Additional arbiter process



Independent arbiter processes check if immudb state moves forward (consistency proof).
Independently deployed arbiter is harder to attack.

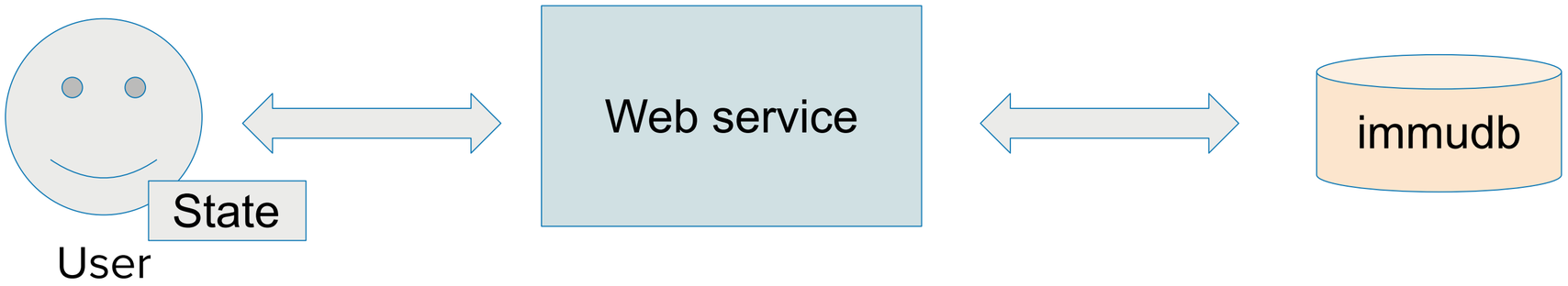
immudb - how to use?

3. State validation in the intermediate web service

Intermediate web service validates database
Compromised database will be detected, but a large attack on whole backend may still get unnoticed

immudb - how to use?

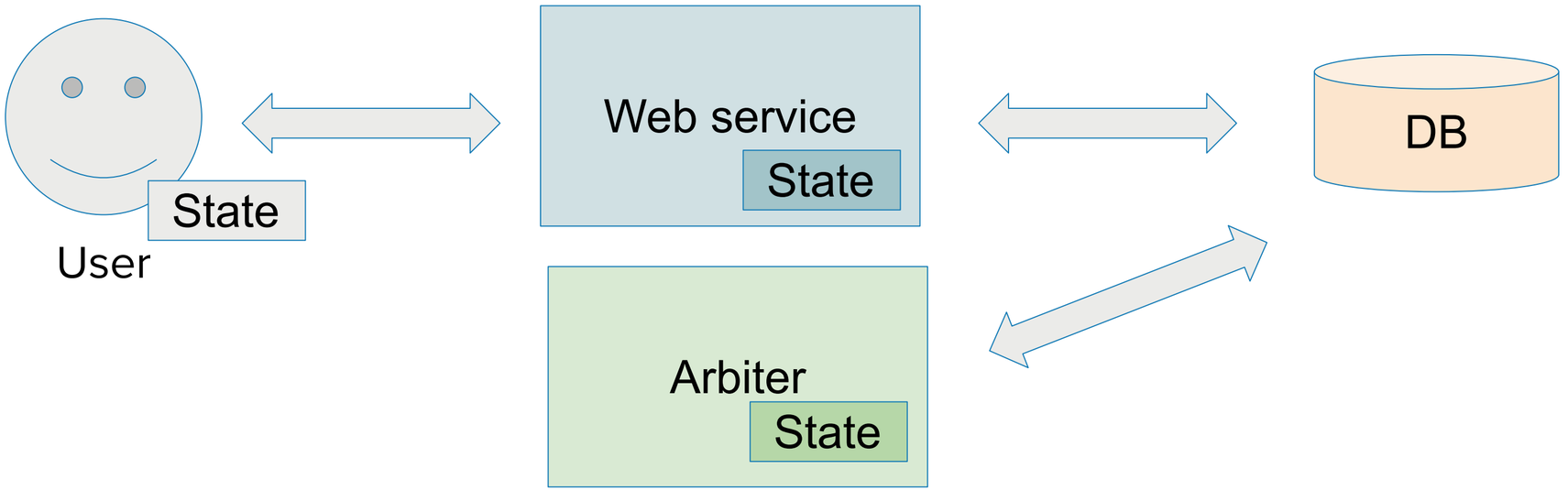
4. State validation done by the end user



Hard to attack - each user acts as an additional arbiter, malicious player must to attack the backend and all users. Requires frequent validations made by users.

immudb - how to use?

Paranoid mode - check everything everywhere



Trust No1

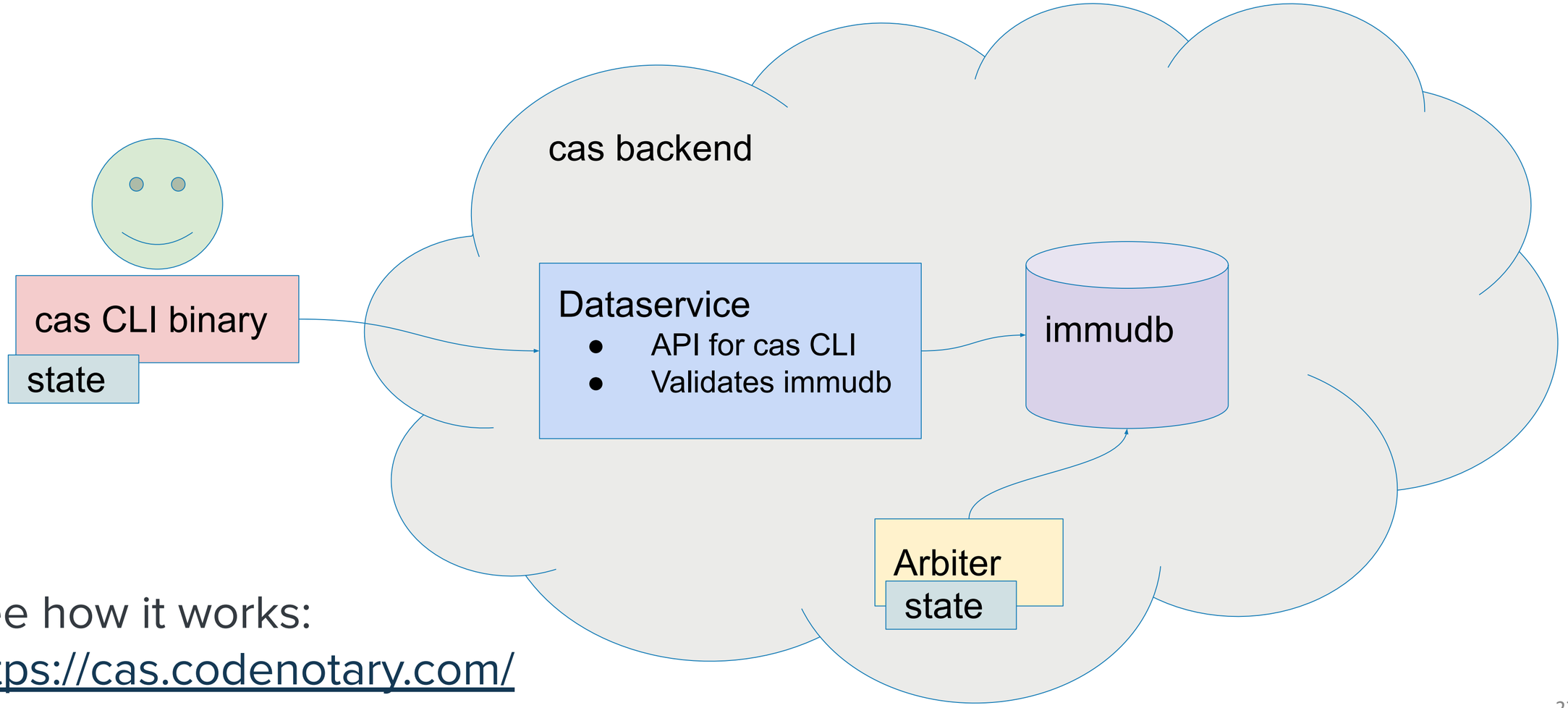
immudb - how to use?

The more validation methods the better

Depends on the runtime environment - e.g. user-side validation may not be possible if there’s no storage available

More distributed system is harder to attack - e.g. dedicated arbiters could be installed on different datacenters or cloud providers

CAS - Practical example of immudb deployment



See how it works:
https://cas.codenotary.com/

Codenotary

THANK YOU

https://immudb.io/
https://github.com/codenotary/immudb/
https://docs.immudb.io/
https://discord.gg/ThSJxNEHhZ

https://codenotary.com/
https://cas.codenotary.com/

bart@codenotary.com

Bonus slides

Code examples - setup

```
$ go mod init fosdem-demo  # Initialize go module
go: creating new go.mod: module fosdem-demo
go: to add module requirements and sums:
        go mod tidy
$ go get github.com/codenotary/immudb@v1.2.2 && go mod tidy  # Get immudb go sdk
go get: added github.com/Masterminds/goutils v1.1.1
go get: added github.com/Masterminds/semver v1.5.0
go get: added github.com/Masterminds/sprig v2.22.0+incompatible
...
$ ./immudb  # Run immudb

(_)                               | | |
 _ _ __ ___  _ __ ___  _   _  __| | |__
| | '  `   \| '  `   \| | | |/  ` | '  \
| | | | | | | | | | | |_| | (_| | |_) |
|_|_| |_| |_|_| |_| |_|\__,_|\__,_|_.__/

immudb 1.2.1
Commit  : f2a471e4b79811a61af7c2e3411a42740154e687
Built by: bart@codenotary.com
Built at: Mon, 17 Jan 2022 15:21:05 CET
================ Config ================
Data dir         : ./data
Address          : 0.0.0.0:3322
Metrics address  : 0.0.0.0:9497/metrics
Config file      : configs/immudb.toml
...
```

Code examples - KV

Set KV entry

```go
package main

import (
    "context"
    "fmt"

    immudb "github.com/codenotary/immudb/pkg/client"
)

func errCheck(err error) {
    if err != nil {
        panic(err)
    }
}

func main() {
    ctx := context.Background()
    client := immudb.NewClient()
    err := client.OpenSession(
        ctx,
        []byte("immudb"), []byte("immudb"),
        "defaultdb",
    )
    errCheck(err)
    defer client.CloseSession(ctx)

    fmt.Println("Connected")

    key := []byte("fosdem")
    value := []byte("2022")

    tx, err := client.Set(ctx, key, value)
    errCheck(err)
    fmt.Println("TX:", tx.Id)
}
```

Read KV entry with verification

```go
package main

import (
    "context"
    "fmt"

    immudb "github.com/codenotary/immudb/pkg/client"
)

func errCheck(err error) {
    if err != nil {
        panic(err)
    }
}

func main() {
    ctx := context.Background()
    client := immudb.NewClient()
    err := client.OpenSession(
        ctx,
        []byte("immudb"), []byte("immudb"),
        "defaultdb",
    )
    errCheck(err)
    defer client.CloseSession(ctx)

    fmt.Println("Connected")

    key := []byte("fosdem")

    val, err := client.VerifiedGet(ctx, key)
    errCheck(err)
    fmt.Println("Value is:", string(val.Value))
}
```

Code examples - SQL (native SDK)

```go
package main

import (
    "context"
    "fmt"

    immudb "github.com/codenotary/immudb/pkg/client"
)

func errCheck(err error) {
    if err != nil {
        panic(err)
    }
}

func main() {
    ctx := context.Background()
    client := immudb.NewClient()
    err := client.OpenSession(
        ctx,
        []byte("immudb"), []byte("immudb"),
        "defaultdb",
    )
    errCheck(err)
    defer client.CloseSession(ctx)

    fmt.Println("Connected")
```

```go
    _, err = client.SQLExec(ctx, `
        CREATE TABLE IF NOT EXISTS demo(
            id INTEGER AUTO_INCREMENT,
            event VARCHAR,
            PRIMARY KEY(id)
        )
    `, nil)
    errCheck(err)

    _, err = client.SQLExec(ctx, `
        INSERT INTO demo(event) VALUES('fosdem 2022')
    `, nil)
    errCheck(err)

    res, err := client.SQLQuery(ctx, `
        SELECT id, event
        FROM demo
        ORDER BY id DESC
        LIMIT 1
    `, nil, true)
    errCheck(err)

    fmt.Printf("Event %d: %s\n",
        res.Rows[0].Values[0].GetN(),
        res.Rows[0].Values[1].GetS(),
    )
}
```

Code examples - SQL (stdlib)

```go
package main

import (
    "database/sql"
    "fmt"

    _ "github.com/codenotary/immudb/pkg/stdlib"
)

func errCheck(err error) {
    if err != nil {
        panic(err)
    }
}

func main() {
    db, err := sql.Open("immudb",
       "immudb://immudb:immudb@127.0.0.1:3322/defaultdb?sslmode=disable" )
    errCheck(err)
    defer db.Close()

    _, err = db.Exec(`
        CREATE TABLE IF NOT EXISTS demo(
            id INTEGER AUTO_INCREMENT,
            event VARCHAR,
            PRIMARY KEY(id)
        )
    `, nil)
    errCheck(err)

    _, err = db.Exec(`
        INSERT INTO demo(event) VALUES('fosdem 2022')
    `, nil)
    errCheck(err)

    rows, err := db.Query(`
        SELECT id, event
        FROM demo
        ORDER BY id DESC
        LIMIT 1
    `, nil)
    errCheck(err)
    defer rows.Close()
    rows.Next()

    var id int64
    var event string
    err = rows.Scan(&id, &event)
    errCheck(err)

    fmt.Printf("Event %d: %s\n", id, event)
}
```

More related links if you're interested...

Description of Merkle Trees and proofs in Certificate Transparency logs:
https://datatracker.ietf.org/doc/html/rfc6962#section-2.1

Alternative KV DB using same techniques (part of sigstore):
https://github.com/sigstore/rekor

Go mod proxy also uses merkle trees and proofs:
https://youtu.be/KqTySYYhPUE?t=1344