Etebase

Your End-to-End Encrypted Backend

Building encrypted applications has never been easier

Tom Hacohen
tom@etebase.com
FOSDEM 2021

@TomHacohen
Some Background

- Creator and maintainer of Etebase & EteSync
- Etebase is an SDK for building end-to-end encrypted applications
- EteSync is a set of end-to-end encrypted apps built using Etebase
The problem:
Our data is exposed!
Solution #1

Self-host everything, but…

- Hosting at home is not always feasible (e.g. CGNAT)
- Hosting on a VPS is still someone else's server
- Requires constant security maintenance and backups
- Only accessible to techies
- The cloud is convenient and cheap
Solution #2

End-to-end encrypt everything!

With end-to-end encryption your data is safe
Common encryption misconceptions

My data is private, because:

- It's encrypted using 256bit TLS!
- It's encrypted at rest using AES!
- It's encrypted in transit and at rest!
But wait, encryption is hard…

- Easy to get wrong - partially solved by *libsodium*
- How do you implement sharing? Access control?
- How do you implement password changes?
- How do you ensure integrity? Conflict resolution?
- What about performance?
Solution: Etebase!

Securely encrypt and upload your data with only a few lines of code.

```javascript
// Setup encryption and login to server
const etebase = await Etebase.Account.login("username", "password");
const collectionManager = etebase.getCollectionManager();

// Create, encrypt and upload a new collection
const collection = await collectionManager.create(
    "collection.type",
    { name: "My data" },
    "My private data!"
);
await collectionManager.upload(collection);
```
Key features and capabilities

- Libraries for a variety of programming languages
- Zero cryptography knowledge needed
- A full revision history of all your data
- Automatic data de-duplication
- Easy collaboration (sharing)
- And more…
Used in projects such as…

 GNOME
 KDE
 Tasks.org
 EteSync
How does it work?
Key components

- Account - a user on the Etebase server
- Collection - a collection of items (e.g. a filesystem)
- Item - what holds the actual data (e.g. files)
- Revisions - a state of the item at a single point in time
- stoken - a token representing a point in time
Data structure
Account

- Main entry point for the Etebase user
- login, signup, logout, and etc.
- You only have one password

Diagram:
- Alice
  - Request Challenge
  - Return Challenge
  - Sign Challenge
  - Auth token
- Server
Collection

- A collection of items
- Have a unique UID
- Associated metadata e.g:
  - name
  - description
- Immutable CollectionType
  - Used to filter collections by usage
- Optional content
Item

- Almost all of the data in Etebase is stored in items
- Have a unique UID
- Also have associated metadata e.g:
  - name
  - description
- Optional content
- Optional revision history
stoken

- Represents a point in time of the data
- Used for efficient syncing (only sync changes)
- Used for integrity checks
Multiple accounts (sharing)
Structuring the data
As a full state sync protocol

- The easiest most common way
- Sync all of the data across devices
- Always fetch the whole data
  - Use sync tokens to only fetch changes
Hierarchical item structure

- When you don't want to sync all of the data
  - E.g. when syncing a large filesystem
- Fetch items by UIDs
Let's build a note taking app!
Well, it's a lightning talk, so just the Etebase parts...
Structuring the data

- Use the note specifications from the docs
- Collection is a notebook
  - Can be shared with other users
  - CollectionType: etebase.md.note
  - name: the name of the notebook
- Items are notes in Markdown:
  - type: null
  - name: the title of the note
Signup and login

Signup

```javascript
const etebase = await Etebase.Account.signup({
    username: "username",
    email: "email"
}, "password", serverUrl);
```

Login

```javascript
const etebase = await Etebase.Account.login("username", "password", serverUrl);
```
Create a notebook

```javascript
const collectionManager = etebase.getCollectionManager();

const collection = await collectionManager.create("etebase.md.note",
{
    name: "My Notes",
    mtime: (new Date()).getTime(),
},
"" // Empty content
);

// Upload the collection to server
await collectionManager.upload(collection);
```
Create a note

```javascript
// Using the collection from earlier
const itemManager = collectionManager.getItemManager(collection);

// Create, encrypt and upload a new item
const item = await itemManager.create(
  {
    name: "Shopping list",
    mtime: (new Date()).getTime(),
  },
  "- [X] Apples
- [ ] Oranges", // Comes from the user
);

// Batch upload of items (just one this time)
await itemManager.batch([item]);
```
// The stoken we got from a previous fetch
let stoken = localStorage.getItem("stoken");
let done = false;

while (!done) {
    const collections = await collectionManager.list("etebase.md.note", { stoken, limit: 30 });

    processChangedCollections(collections.data);

    stoken = collections.stoken;
    done = collections.done;
}

localStorage.setItem("stoken", stoken); // Persist stoken
// The stoken we got from a previous fetch
let stoken = localStorage.getItem(`stoken.${collection.uid}`);
let done = false;

while (!done) {
  const items = await itemManager.list({ stoken, limit: 30 });

  processChangedItems(items.data);

  stoken = items.stoken;
  done = items.done;
}

localStorage.setItem(`stoken.${collection.uid}`, stoken); // Persist stoken
const itemManager = collectionManager.getItemManager(collection);

const subscription = await itemManager.subscribeChanges((items) => {
  processChangedItems(items.data);
  localStorage.setItem(`stoken.${collection.uid}`, stoken); // Persist stoken
});
Caching notes locally

Collections

// The cache blob is just a Uint8Array that can be saved for later use
const cacheBlob = collectionManager.cacheSave(collection);

// Later on we can load the object back
const collection = collectionManager.cacheLoad(cacheBlob);

Items

// The cache blob is just a Uint8Array that can be saved for later use
const cacheBlob = itemManager.cacheSave(item);

// Later on we can load the object back
const item = itemManager.cacheLoad(cacheBlob);
And now it's time to logout...

```
await etebase.logout();
```
Closing words

Developer looking to secure user data?
Come chat with us!

Using apps that could benefit from Etebase?
Let us (and them) know!
Questions?

- Etebase: https://www.etebase.com
- Sources: https://github.com/etesync/
- Docs: https://docs.etebase.com
- Chat: https://www.etebase.com/community-chat/
- EteSync: https://www.etesync.com