VAULTER- Your Secure API Key Manager

USER DOCUMENTATION

Have you ever found it difficult to locate or manage your API keys across various websites and apps? We understand how frustrating it can be - juggling numerous keys, ensuring they remain safe, and avoiding accidental leaks can be overwhelming.

Introducing **VAULTER** - your personal, ultra-secure vault for managing API keys. Designed with security and simplicity at its core, Vaulter protects your sensitive credentials using top-tier encryption standards. With a clean and user-friendly interface, you can organize, store, and access all your API keys with ease all in one secure place. The core promise of our service is simple: **Your API keys are never stored in plaintext**.

- **Uncompromising Security:** We use military-grade **AES-256** encryption to protect your keys. From the moment you enter a key, it's encrypted on our backend and stored securely in our database. Only you can decrypt and view your keys.
- **Simple and Secure Login:** We use Clerk for authentication, allowing you to log in quickly and securely using trusted providers like Google, GitHub, or your email. All access to your data is protected and verified.
- **Intuitive Dashboard:** Your dashboard gives you a clear overview of all your keys. You can see statistics like how many keys stored, last modified and how many times they have been used, filter your keys by tags, and find what you need in seconds.
- **Safe Management:**
    - **Add Keys Easily:** A simple form lets you add new keys.
    - **Masked by Default:** Your keys are always hidden behind a "masked" view (e.g., sk-....Abc1) for your safety.
    - **Reveal and Copy:** Securely reveal a key when you need it and copy it to your clipboard with a single click.
    - **Track Usage:** Keep track of when your keys were last used and how often.

How is Your Security Guaranteed?

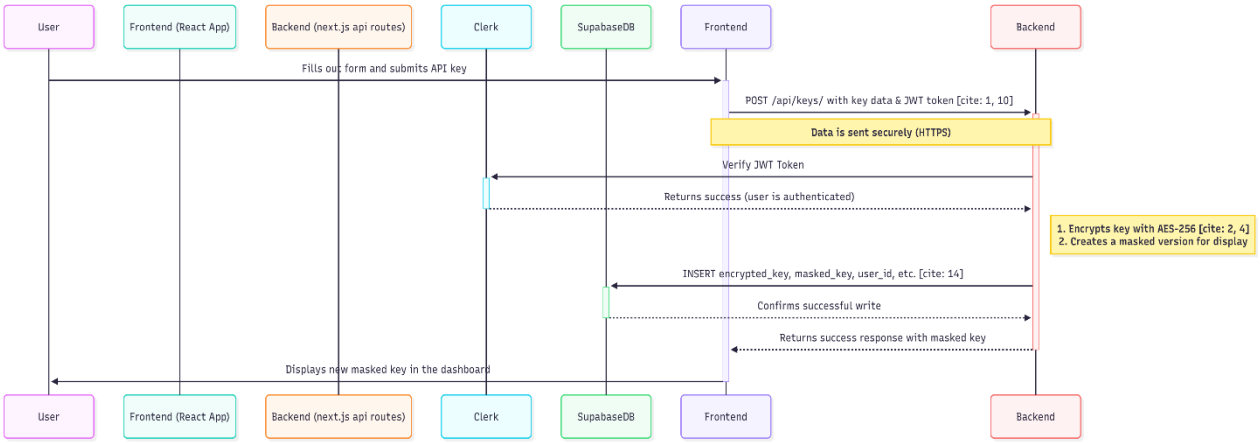
Understanding how your data is handled is key to trusting the system. Here is the journey your API key takes to ensure its security:

1. User **to Frontend:** You start by entering the key's details into the web interface.

2. Frontend **to Backend:** The React application sends this information, along with your secure authentication token (JWT), to the Next.js API routes server.
3. Backend **to Clerk:** The server first checks with Clerk to verify your token, ensuring the request is coming from a logged-in, authenticated user.
4. Backend **Processing:** Once you are authenticated, the server performs the critical security step: it encrypts your API key using AES-256 so it's no longer in plaintext.
5. Backend **to SupabaseDB:** The server stores only the **encrypted** version of your key in the Supabase database. The plaintext key is never saved.
6. Backend **to Frontend:** The server confirms the operation was successful and sends back the safe, masked version of the key to be displayed.
7. Frontend **to User:** Your dashboard updates to show the newly added key in its masked, secure format.

Visual Workflow: How Adding a Key Works

This diagram illustrates the secure journey your API key takes from your browser to our encrypted database.



Database Schema Explanation

Database Schema: api_keys Table

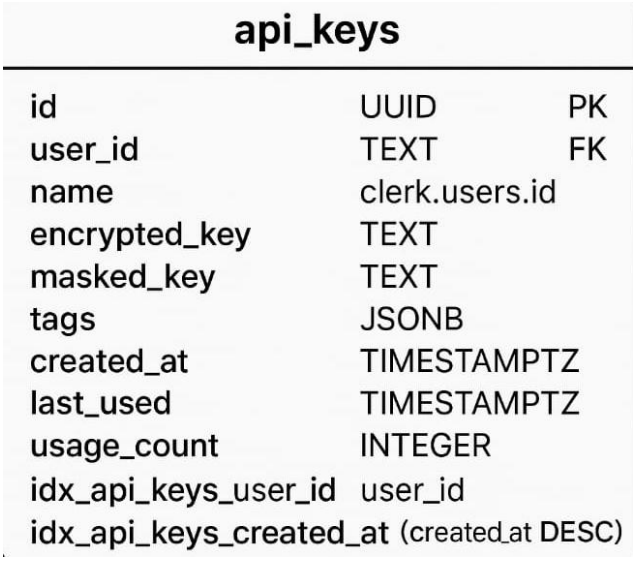
The API key information is stored in a structured format. Below is a description of the database table that holds your data. Notice that we only ever store the encrypted version of your key.

Column Name	Data Type	Description
id	UUID	A unique, randomly generated ID for each key entry.
user_id	TEXT	The unique identifier that links this key directly to your user account.
name	TEXT	A user-friendly name you give to the key (e.g., "OpenAI Key").
encrypted_key	TEXT	The API key after it has been encrypted with AES-256. This is not readable.
masked_key	TEXT	A safe, partially hidden version of the key for display purposes (e.g., sk-....1234).
tags	JSONB	Custom tags you can add to organize and filter your keys.
created_at	TIMESTAMP	The date and time of when the key was added to the vault.
last_used	TIMESTAMP	The date and time the key was last accessed or used.
usage_count	INTEGER	A counter for how many times the key has been used.

Code :

```
CREATE TABLE IF NOT EXISTS api_keys (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id TEXT NOT NULL,
  name TEXT NOT NULL,
  encrypted_key TEXT NOT NULL,
  masked_key TEXT NOT NULL,
```

```
  tags JSONB DEFAULT '[]'::jsonb,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  last_used TIMESTAMP WITH TIME ZONE,
  usage_count INTEGER DEFAULT 0
);
CREATE INDEX IF NOT EXISTS idx_api_keys_user_id ON api_keys(user_id);
CREATE INDEX IF NOT EXISTS idx_api_keys_created_at ON api_keys(created_at DESC)
```



How To Use Vaulter?

Upon accessing the application, you will be redirected to the Login Page. You can either sign up or log in using your email, GitHub, or Google account credentials.

Dashboard Overview

- Once logged in successfully, you will be greeted with a clean and aesthetically designed dashboard.
- This interface provides key metrics such as:
  - Total Keys
  - Recently Used Keys
  - Unique Tags

- These metrics offer a quick overview of your stored API keys and their usage.

Adding a New API Key

- To store a new API key in Vaulter, click on the "+ Add Key" button on the dashboard.
- A modal window will appear, prompting you to enter the following details:
    - Key Name
    - API Key
    - Tags
- Ensure that all entered information is accurate before proceeding.
- Once all fields are filled, click "Add Key" to save the entry.

Viewing Stored Keys

- After successfully adding an API key, it will appear on your dashboard with the following details:
    - Name Assigned
    - Date Created
    - Number of Times Used
    - Last Used Date
- For privacy and security reasons, the API key will be displayed in a masked format.

Copying and Using Your API Key

- Users can easily copy their API keys directly from the dashboard whenever needed.
- This feature ensures seamless integration and quick access while maintaining data confidentiality.

Deleting Your Stored API Keys

- There is an option to delete your API key at any point in time. Just click on the "delete" icon.
- This ensures that the key is deleted from the database and will not be available to the user anymore.

You can now securely store and retrieve your API keys anytime, all from one intuitive dashboard.

TECHNICAL DOCUMENTATION

The Secure API Key Manager is a full-stack application designed for safely managing, encrypting, and tracking API keys. It ensures that no API key is ever stored in plaintext, offering robust AES-256 encryption and JWT-based authentication via Clerk. The system integrates a Next.js API routes backend, Supabase for database management, and a modern React + TailwindCSS frontend for a smooth, glassmorphic user interface.

Backend (Next.js API routes)

Authentication: Clerk JWT verification middleware

Encryption: AES-256 encryption using the Cryptography library

Database: Supabase integration with encrypted storage

Security: Keys are encrypted before storage; never stored in plaintext

Deployment: Vercel

API Endpoints

1 POST /api/keys/ - Create encrypted API key

2 GET /api/keys/ - List all keys (masked)

3 GET /api/keys/{id} - Get decrypted key

4 DELETE /api/keys/{id} - Delete key

5 POST /api/usage/{id} - Log usage event

Clerk Authentication Integration

Clerk provides secure user authentication and management for the platform. Users can log in using multiple identity providers (Google, GitHub, Email, etc.) and manage sessions seamlessly. Clerk JWT tokens are verified in the Next.js API routes backend via middleware to protect all API routes.

Clerk also allows account linking and management features — meaning users can connect multiple accounts or API credentials under a single profile. Each request to protected routes includes a verified JWT for secure **access** control.

Supabase Integration

Supabase serves as the encrypted database for storing user data and API keys. The database never stores plaintext values; instead, encrypted values generated via AES-256 are stored. Supabase API keys are securely managed through environment variables and never exposed in client-side code. Supabase also provides real-time capabilities and secure row-level access policies, ensuring that each user can only access their own records.

Frontend (React + Vite + TailwindCSS)

- Authentication: Clerk integration with protected routes
- UI: Glassmorphic design with backdrop blur and Framer Motion animations
- Dashboard with stats: total keys, recently used keys, and tag filters
- Add Key modal with validation and masked display for sensitive data
- Copy-to-clipboard and reveal/hide functionalities
- Search and filter by tags

Conclusion

This system combines strong encryption, modern authentication, and a clean UI to create a secure and elegant API key management platform. Its architecture is scalable, privacy-focused, and designed to meet production standards.