A Secure Password Wallet based on the SEcube™ framework

Walter Gallego Gómez

Department of control and computer engineering
Politecnico di Torino

July 23, 2018

Motivation

The need for a hardware-based password manager is justified answering these three questions:

Motivation

The need for a hardware-based password manager is justified answering these three questions:

Are passwords still relevant?

Motivation

The need for a hardware-based password manager is justified answering these three questions:

Are passwords still relevant?

Yes, they are the dominant form of authentication.

Motivation

The need for a hardware-based password manager is justified answering these three questions:

Are passwords still relevant?

Yes, they are the dominant form of authentication.

Why should people use password managers?

Motivation

The need for a hardware-based password manager is justified answering these three questions:

Are passwords still relevant?

Yes, they are the dominant form of authentication.

Why should people use password managers?

So they can use unique strong passwords.

Motivation

The need for a hardware-based password manager is justified answering these three questions:

Are passwords still relevant?

Yes, they are the dominant form of authentication.

Why should people use password managers?

So they can use unique strong passwords.

Why are hardware-based approaches more reliable?

Motivation

The need for a hardware-based password manager is justified answering these three questions:

Are passwords still relevant?

Yes, they are the dominant form of authentication.

Why should people use password managers?

So they can use unique strong passwords.

Why are hardware-based approaches more reliable?

To authenticate, Master password + Device are required

Outline

Outline

1. Introduction
2. Technologies used
   - Software libraries
   - The SEcube™ Framework
3. Design and implementation
   - Basic SEcube™ Operation
   - General Architecture
   - Implementation details
4. Demos
5. Conclusions
6. Future Work

Introduction

Exploit the capabilities of the SEcube™ (Secure Environment cube) hardware and software framework to store and protect a set of passwords (Wallet).

The desktop application, named **SEcubeWallet**, was written in C/C++and Qt, and it interacts with a SEcube™ device, requesting services like authentication and encryption/decryption of data.

Outline

1. Introduction
2. **Technologies used**
   - ▶ Software libraries
   - ▶ The SEcube™ Framework
3. Design and implementation
   - ▶ Basic SEcube™ Operation
   - ▶ General Architecture
   - ▶ Implementation details
4. Demos
5. Conclusions
6. Future Work

Software Libraries

The following open source libraries were used:

Software Libraries

The following open source libraries were used:

Qt: Graphical User Interface

Software Libraries

The following open source libraries were used:

Qt: Graphical User Interface

C++ library, cross-platform, variety of display elements

Software Libraries

The following open source libraries were used:

Qt: Graphical User Interface

C++ library, cross-platform, variety of display elements

SQLite: DataBase management

Software Libraries

The following open source libraries were used:

Qt: Graphical User Interface

C++ library, cross-platform, variety of display elements

SQLite: DataBase management

Self-contained, written in C, Transactional

Software Libraries

The following open source libraries were used:

Qt: Graphical User Interface

C++ library, cross-platform, variety of display elements

SQLite: DataBase management

Self-contained, written in C, Transactional

PwGen: Password generator

Software Libraries

The following open source libraries were used:

Qt: Graphical User Interface

C++ library, cross-platform, variety of display elements

SQLite: DataBase management

Self-contained, written in C, Transactional

PwGen: Password generator

Configurable, random or readable

Software Libraries

The following open source libraries were used:

Qt: Graphical User Interface

C++ library, cross-platform, variety of display elements

SQLite: DataBase management

Self-contained, written in C, Transactional

PwGen: Password generator

Configurable, random or readable

zxcvbn: Password strength estimator

Software Libraries

The following open source libraries were used:

Qt: Graphical User Interface

C++ library, cross-platform, variety of display elements

SQLite: DataBase management

Self-contained, written in C, Transactional

PwGen: Password generator

Configurable, random or readable

zxcvbn: Password strength estimator

Dictionaries, keyboard patterns, sequences, years

The SEcube™ Open Security Platform

Hardware

Software

The SEcube™ Open Security Platform

Hardware

Developed by the Blu5 Group

Software

The SEcube™ Open Security Platform

Hardware

Developed by the Blu5 Group

Family

- SEcube™ Chip
- SEcube™ DevKit
- USEcube™ Stick

Software

The SEcube™ Open Security Platform

Hardware

Developed by the Blu5 Group

Family

- SEcube™ Chip
- SEcube™ DevKit
- USEcube™ Stick

SEcube™ Chip

- **MCU:** STM32F4 (STM)
- **FPGA:** MachXO2-7000 (Lattice)
- **Smart Card:** SLJ52G (infineon)

Software

The SEcube™ Open Security Platform

Hardware

Developed by the Blu5 Group

Family

- SEcube™ Chip
- SEcube™ DevKit
- USEcube™ Stick

SEcube™ Chip

- **MCU:** STM32F4 (STM)
- **FPGA:** MachXO2-7000 (Lattice)
- **Smart Card:** SLJ52G (infineon)

Software

Developed by European research institutions. Written in C using the Eclipse IDE.

The SEcube™ Open Security Platform

Hardware

Developed by the Blu5 Group

Family

- SEcube™ Chip
- SEcube™ DevKit
- USEcube™ Stick

SEcube™ Chip

- **MCU:** STM32F4 (STM)
- **FPGA:** MachXO2-7000 (Lattice)
- **Smart Card:** SLJ52G (infineon)

Software

Developed by European research institutions. Written in C using the Eclipse IDE.

Firmware: Developers can customize the firmware to their needs, and load the updated version to the SEcube™ chip.

The SEcube™ Open Security Platform

Hardware

Developed by the Blu5 Group

Family

- SEcube™ Chip
- SEcube™ DevKit
- USEcube™ Stick

SEcube™ Chip

- **MCU:** STM32F4 (STM)
- **FPGA:** MachXO2-7000 (Lattice)
- **Smart Card:** SLJ52G (infineon)

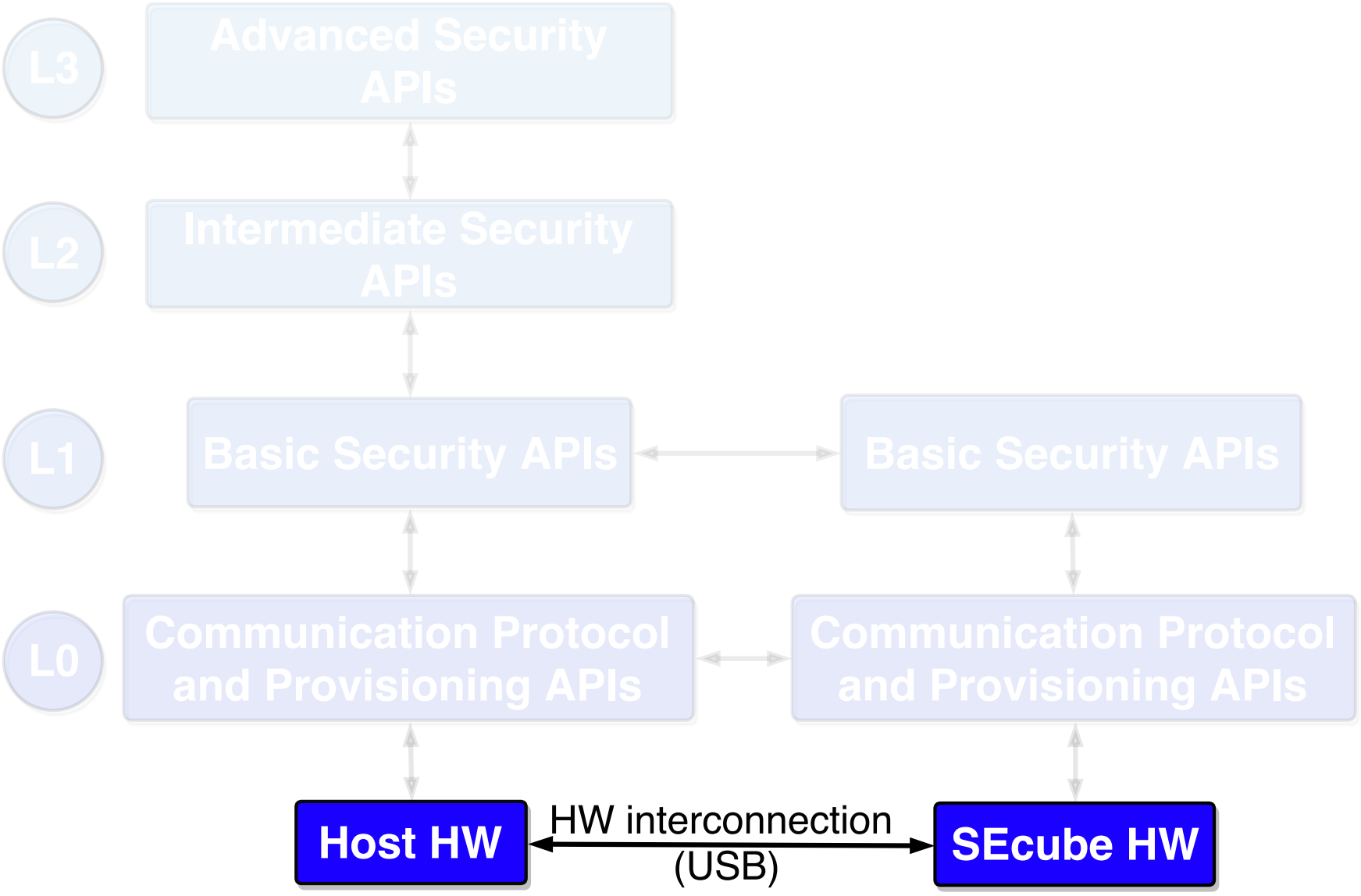
Software

Developed by European research institutions. Written in C using the Eclipse IDE.

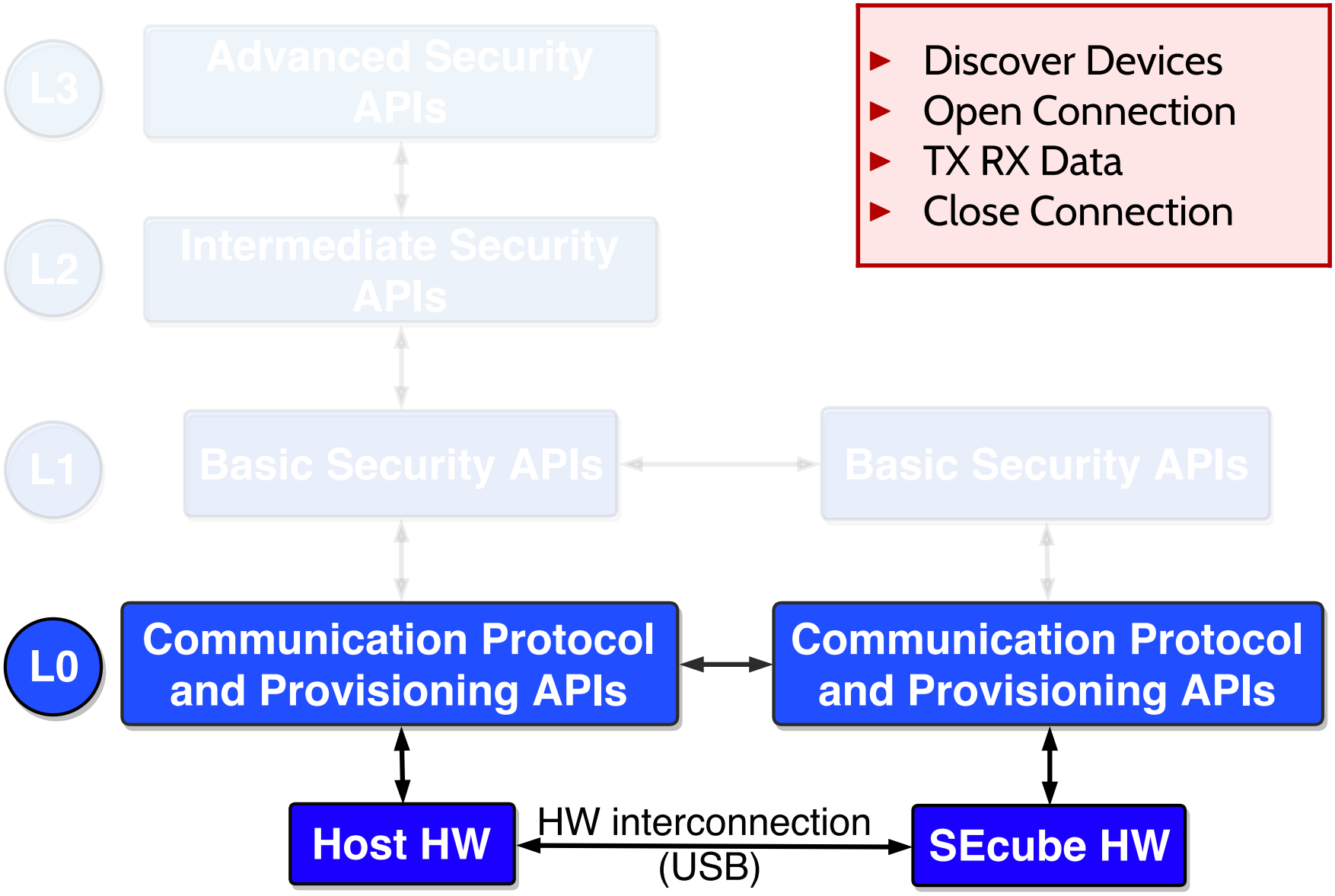
Firmware: Developers can customize the firmware to their needs, and load the updated version to the SEcube™ chip.

Host libraries: Allow to experience the platform as a high-security black box.

SEcube™ APIs hierarchy

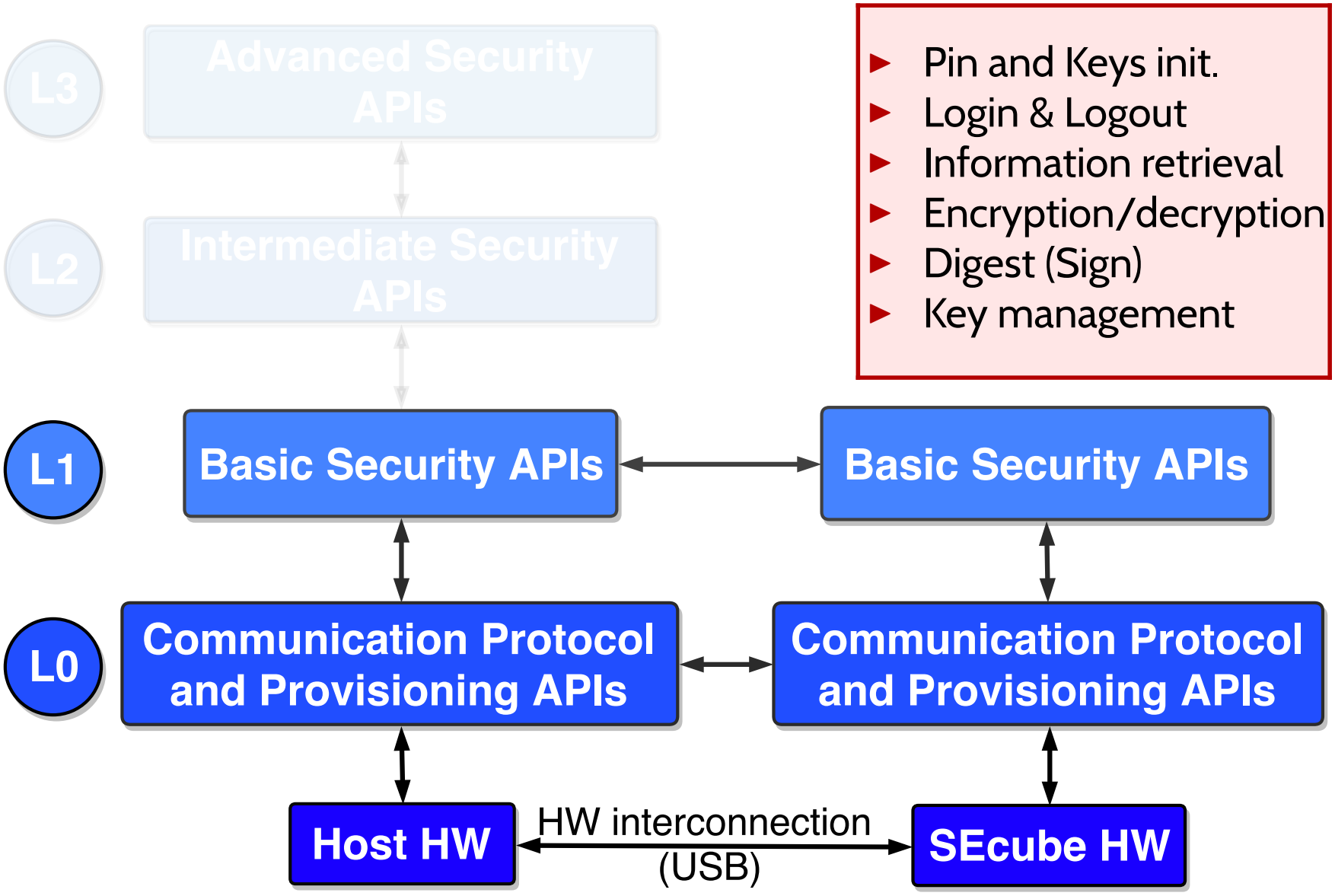


SEcube™ APIs hierarchy

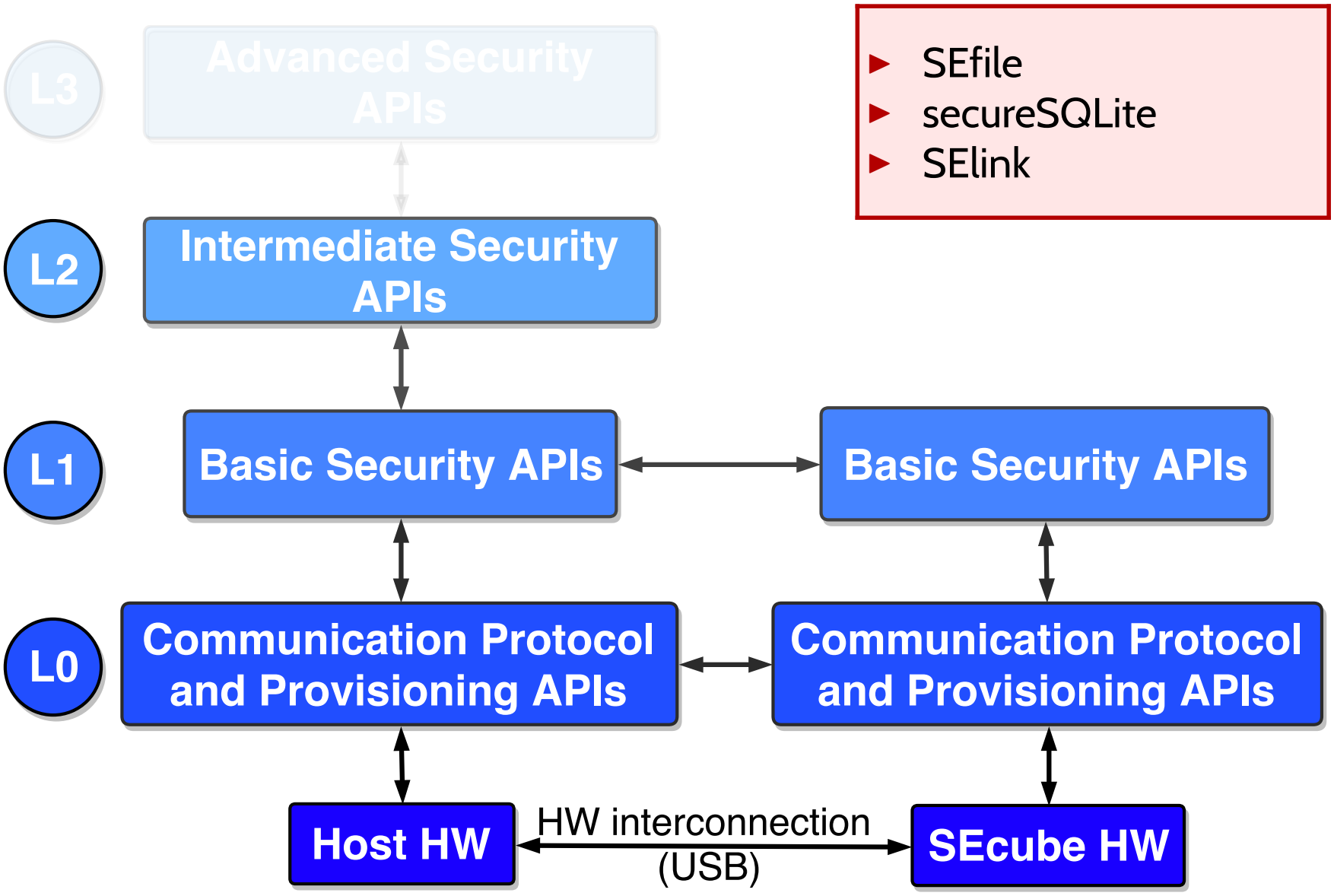


- Discover Devices
- Open Connection
- TX RX Data
- Close Connection

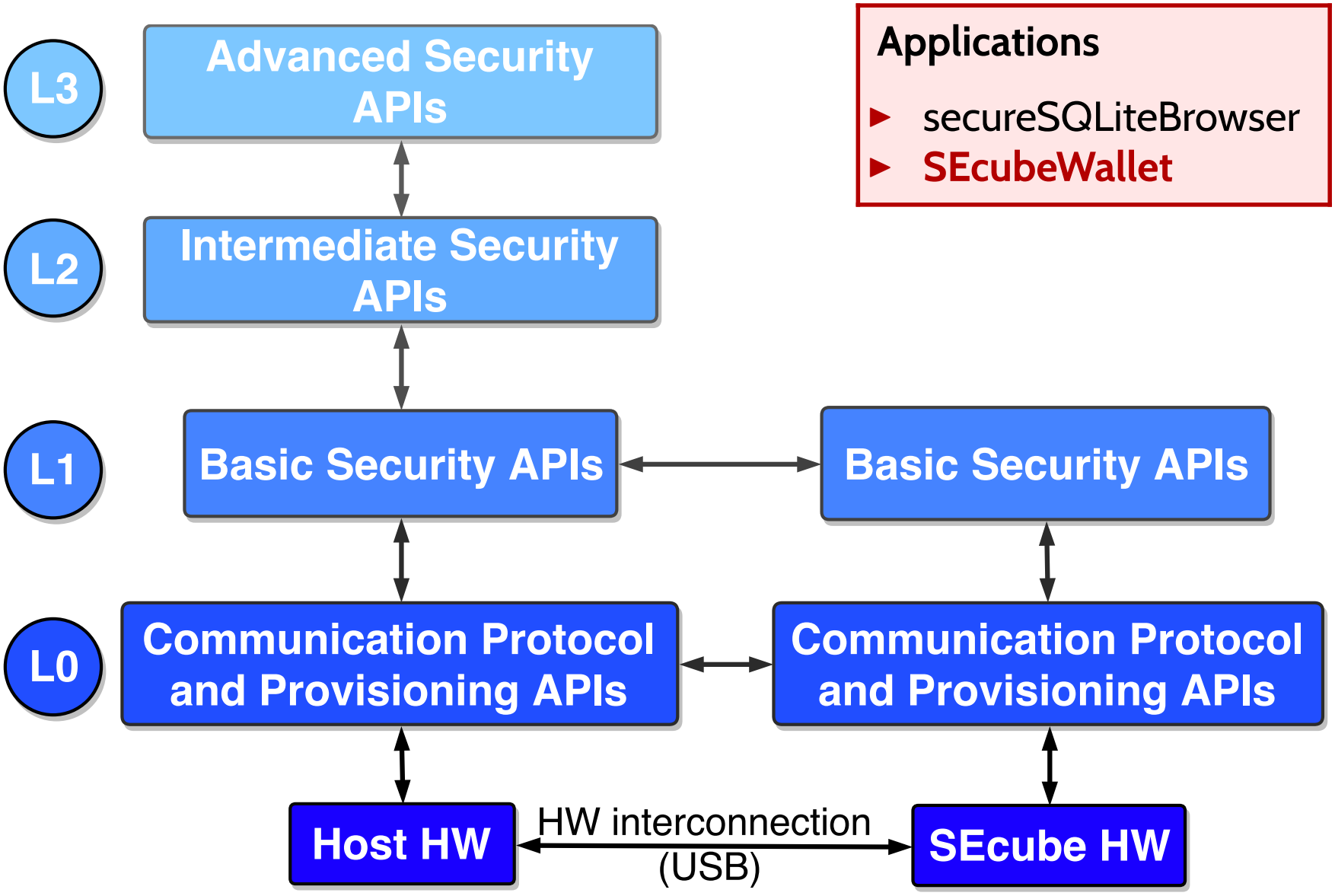
SEcube™ APIs hierarchy



SEcube™ APIs hierarchy



SEcube™ APIs hierarchy



Outline

Basic SEcube™ Operation

Basic SEcube™ Operation

- At factory initialization, an admin/developer writes to the SEcube™ flash memory:
  - Admin pin
  - User pin
  - User Keys

Basic SEcube™ Operation

- At factory initialization, an admin/developer writes to the SEcube™ flash memory:
  - Admin pin
  - User pin
  - User Keys
- User logins with its pin and a challenge-based authentication.

Basic SEcube™ Operation

- At factory initialization, an admin/developer writes to the SEcube™ flash memory:
  - Admin pin
  - User pin
  - User Keys
- User logins with its pin and a challenge-based authentication.
- User chooses which of the keys to use to encrypt/decrypt.

Basic SEcube™ Operation

- At factory initialization, an admin/developer writes to the SEcube™ flash memory:
  - Admin pin
  - User pin
  - User Keys
- User logins with its pin and a challenge-based authentication.
- User chooses which of the keys to use to encrypt/decrypt.
- The user request encryption/decryption to the device.

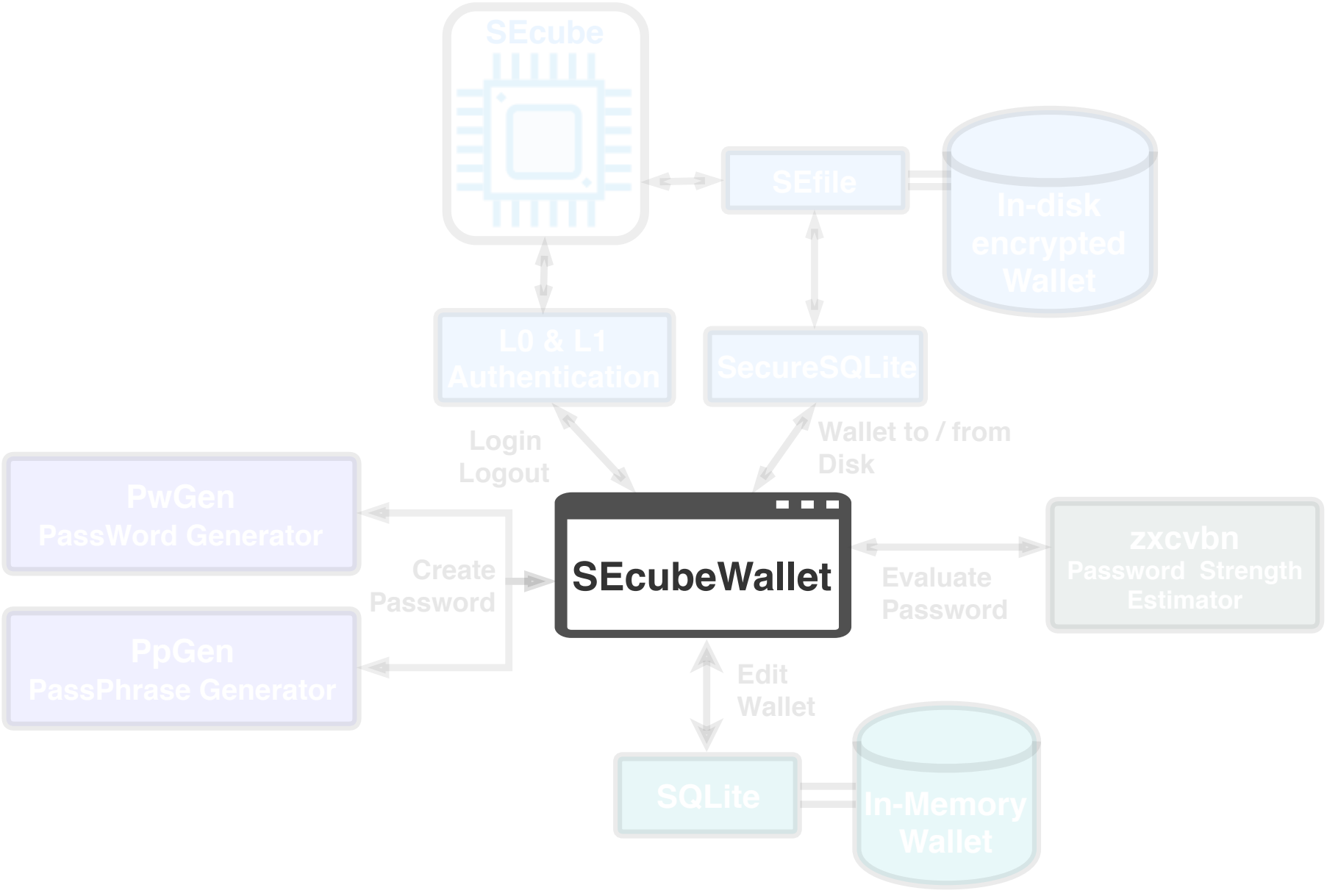
Basic SEcube™ Operation

- At factory initialization, an admin/developer writes to the SEcube™ flash memory:
  - Admin pin
  - User pin
  - User Keys
- User logins with its pin and a challenge-based authentication.
- User chooses which of the keys to use to encrypt/decrypt.
- The user request encryption/decryption to the device.

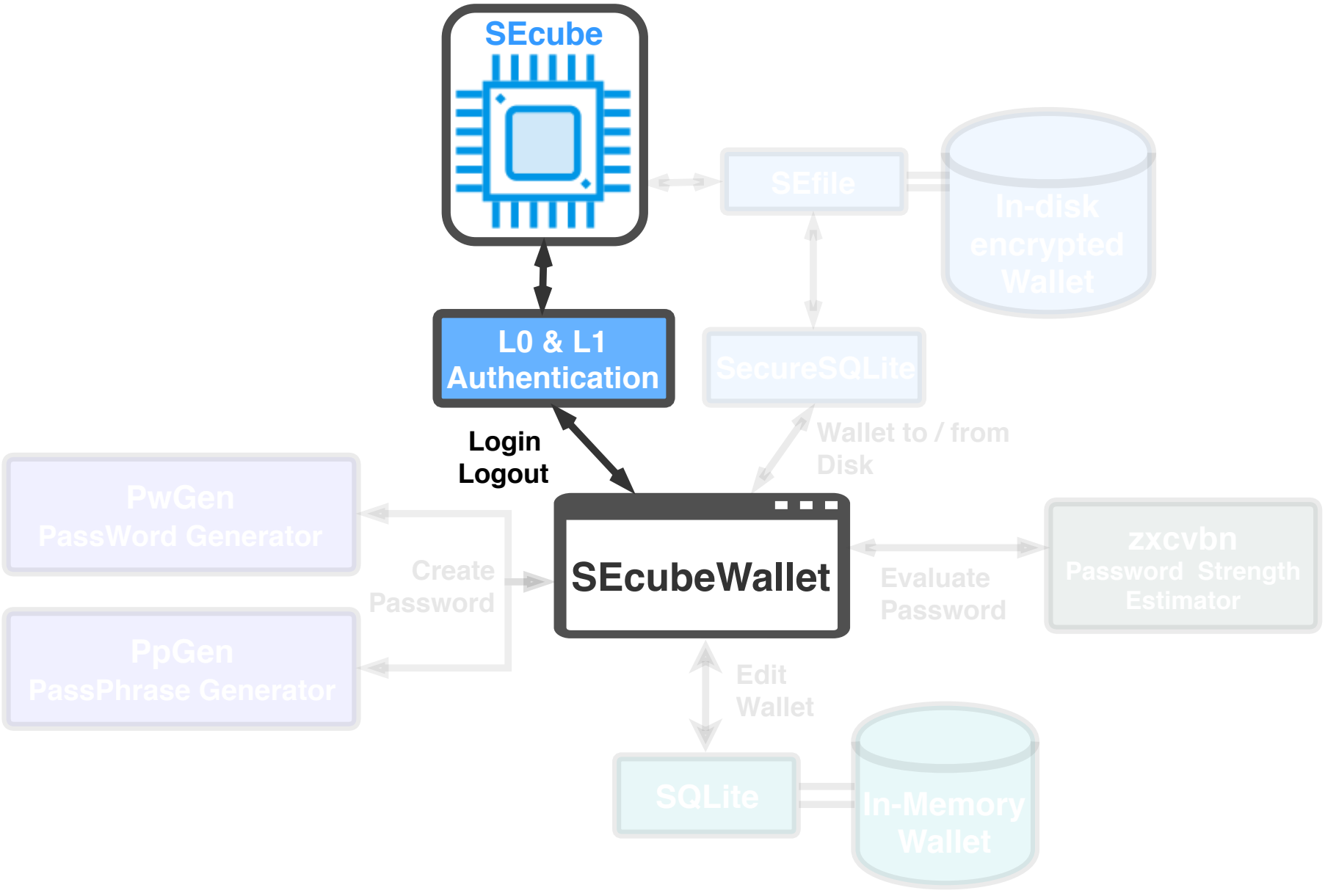
The data (passwords) can only be accessed if:

- SEcube™ device is connected
- Login pin is the correct one
- Key inside the device is the correct one.

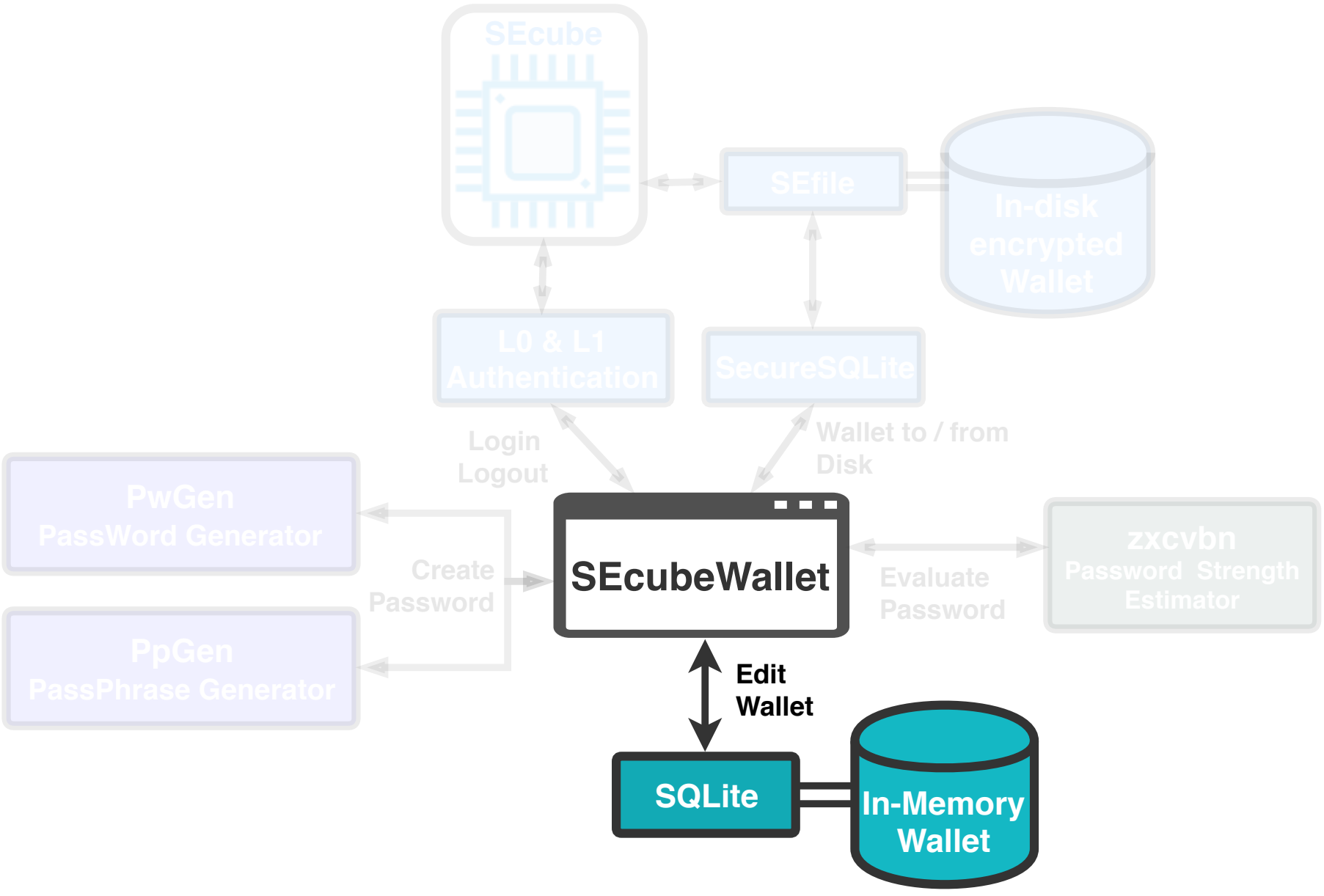
SEcubeWallet Application



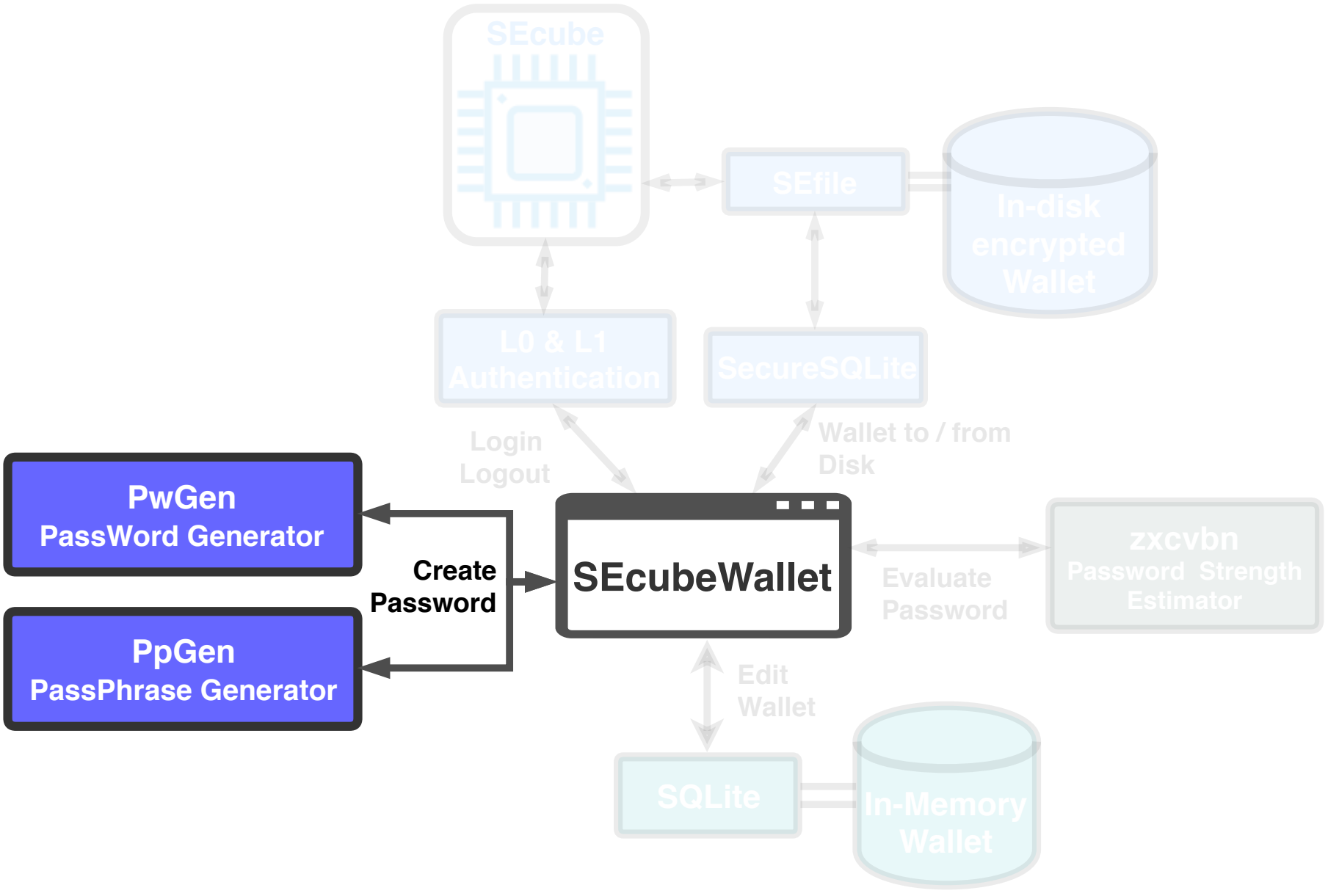
Open device and authenticate



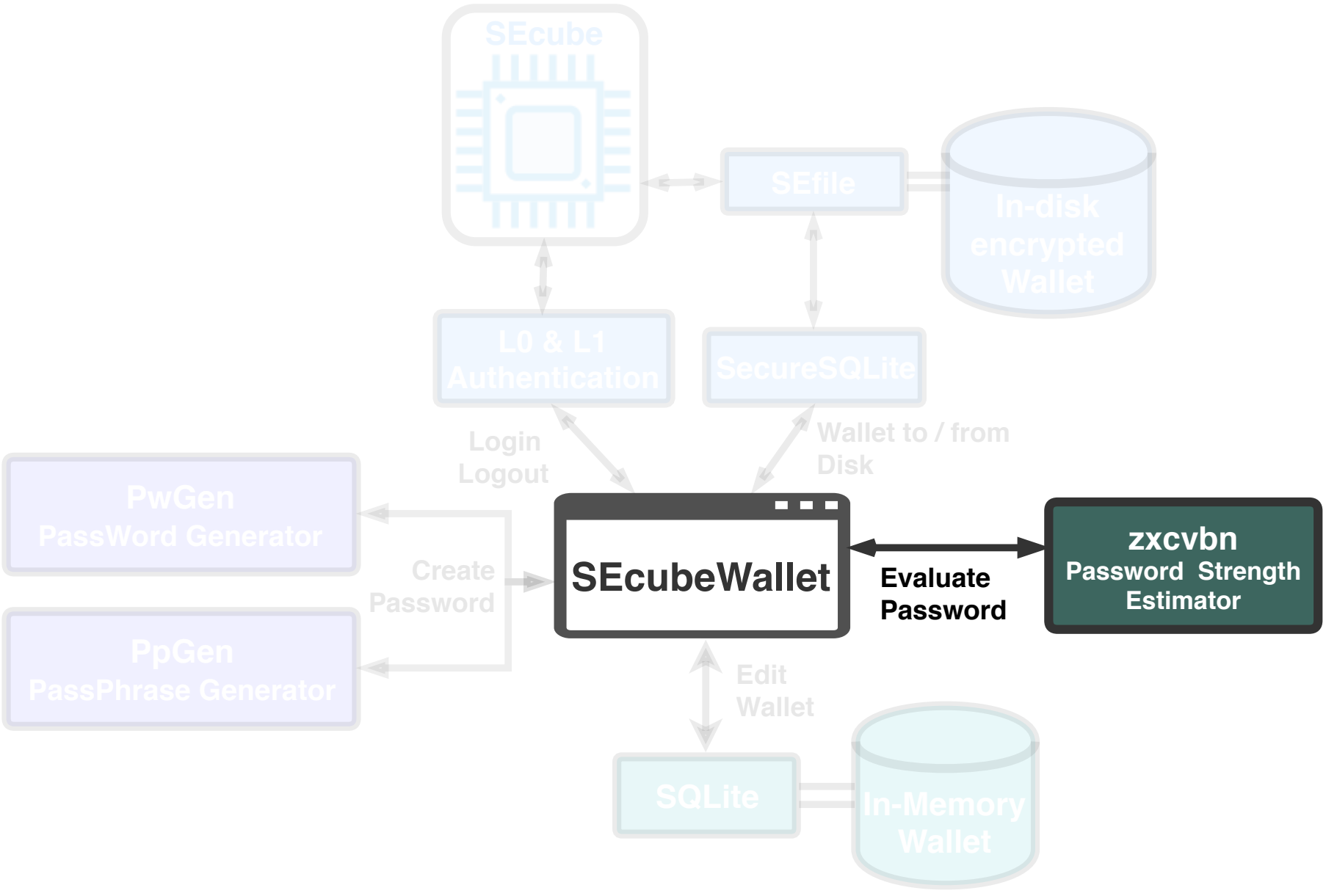
Create In-memory Wallet



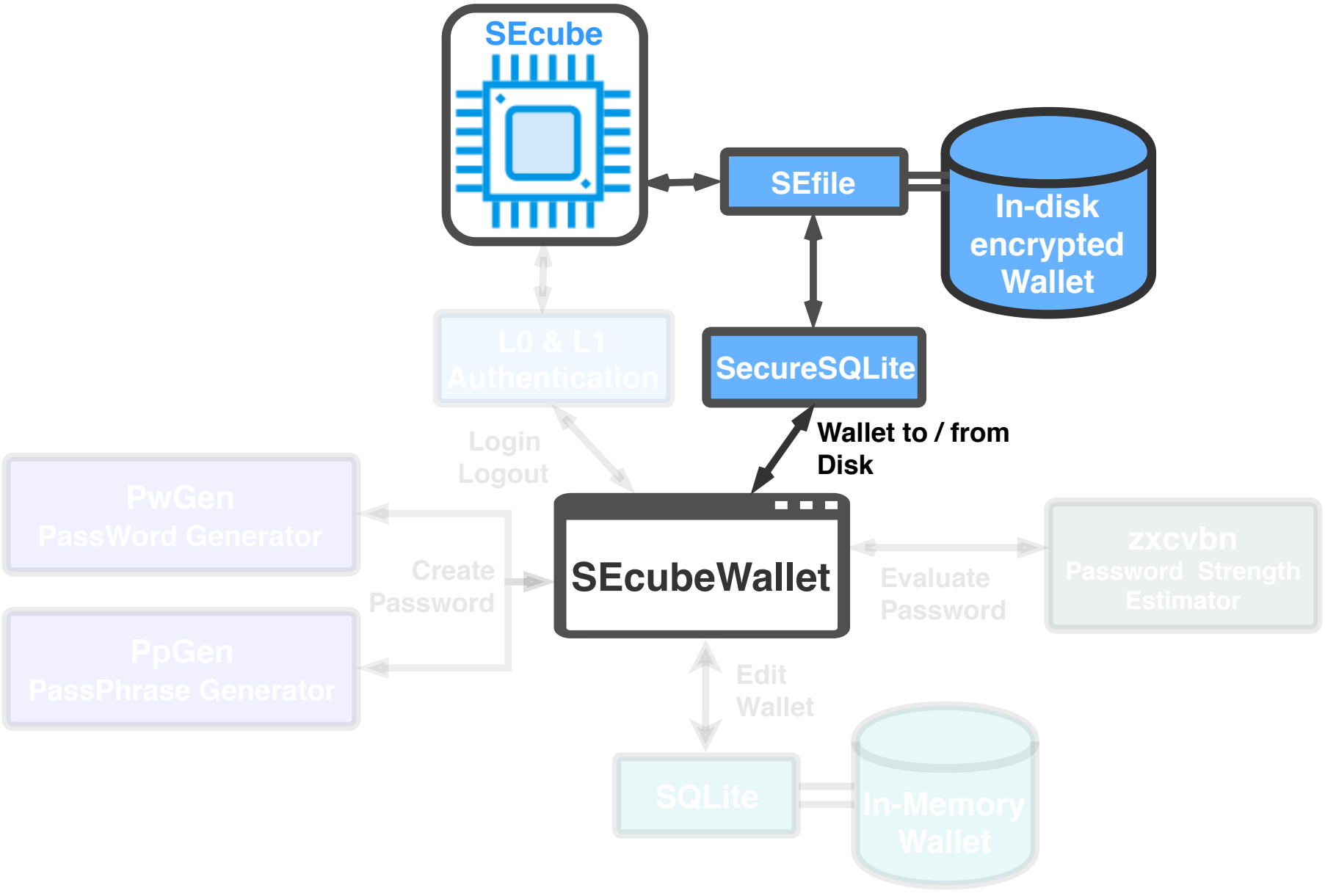
Generate Password/Passphrase



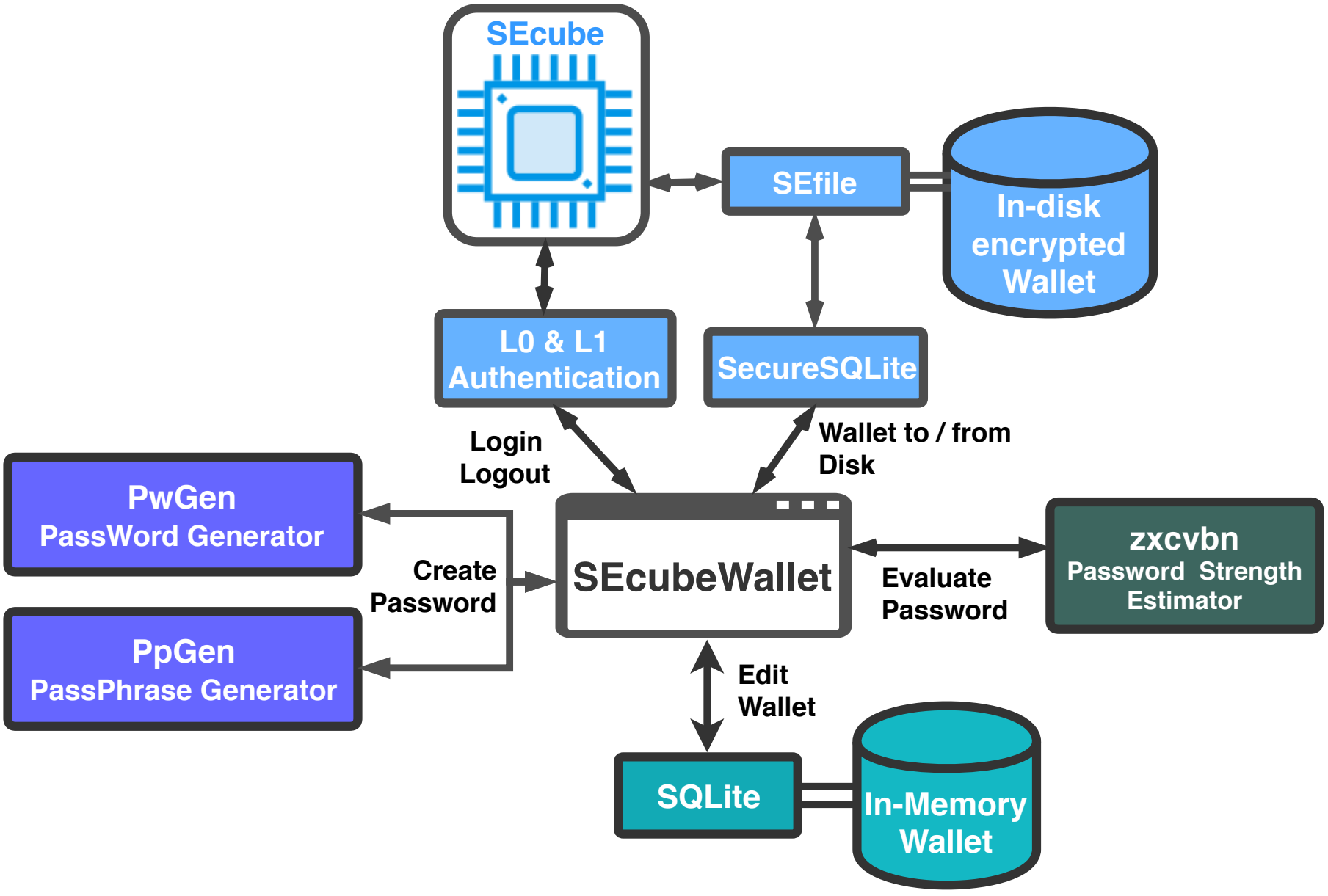
Evaluate Strength



Encrypt and Save Wallet to disk



General Architecture



In-Memory and In-Disk DBs

We want to achieve both security and efficiency:

In-Memory and In-Disk DBs

We want to achieve both security and efficiency:

- **New Wallet:** An In-memory SQLite DB is created.

In-Memory and In-Disk DBs

We want to achieve both security and efficiency:

- **New Wallet:** An In-memory SQLite DB is created.
- **Save Wallet:** An In-disk encrypted secureSQLite DB is created and populated with the In-memory DB contents

In-Memory and In-Disk DBs

We want to achieve both security and efficiency:

- **New Wallet:** An In-memory SQLite DB is created.
- **Save Wallet:** An In-disk encrypted secureSQLite DB is created and populated with the In-memory DB contents
- **Open Wallet:** The selected In-disk DB is decrypted and its contents are dumped into an In-memory DB

In-Memory and In-Disk DBs

We want to achieve both security and efficiency:

- **New Wallet:** An In-memory SQLite DB is created.
- **Save Wallet:** An In-disk encrypted secureSQLite DB is created and populated with the In-memory DB contents
- **Open Wallet:** The selected In-disk DB is decrypted and its contents are dumped into an In-memory DB
- **Delete Wallet:** Both the In-memory DB and the In-disk encrypted file are deleted.

Windows and display elements

- Main Window

Windows and display elements

- Main Window
  - **Table View:** for displaying the wallet entries

Username	Domain	Password	Date	Description

Windows and display elements

- Main Window
  - **Table View:** for displaying the wallet entries

Username	Domain	Password	Date	Description

  - **Filters:** So the user can search in each of the table's columns.

Windows and display elements

- Main Window
  - **Table View:** for displaying the wallet entries

Username	Domain	Password	Date	Description

  - **Filters:** So the user can search in each of the table's columns.
  - **Tool Bars:** Open Wallet, Save Wallet, Add entry etc.

Windows and display elements

- Main Window
  - **Table View:** for displaying the wallet entries

Username	Domain	Password	Date	Description

  - **Filters:** So the user can search in each of the table's columns.
  - **Tool Bars:** Open Wallet, Save Wallet, Add entry etc.
  - **Status Bar:** Used to display messages and the wallet's name

Windows and display elements

- Main Window
  - **Table View:** for displaying the wallet entries

Username	Domain	Password	Date	Description

  - **Filters:** So the user can search in each of the table's columns.
  - **Tool Bars:** Open Wallet, Save Wallet, Add entry etc.
  - **Status Bar:** Used to display messages and the wallet's name
- Preference Window
  - Password Generators Configuration.
  - zxcvbn Configuration.

Windows and display elements

- Main Window
  - **Table View:** for displaying the wallet entries

Username	Domain	Password	Date	Description

  - **Filters:** So the user can search in each of the table's columns.
  - **Tool Bars:** Open Wallet, Save Wallet, Add entry etc.
  - **Status Bar:** Used to display messages and the wallet's name
- Preference Window
  - Password Generators Configuration.
  - zxcvbn Configuration.
- Help Window

Windows and display elements

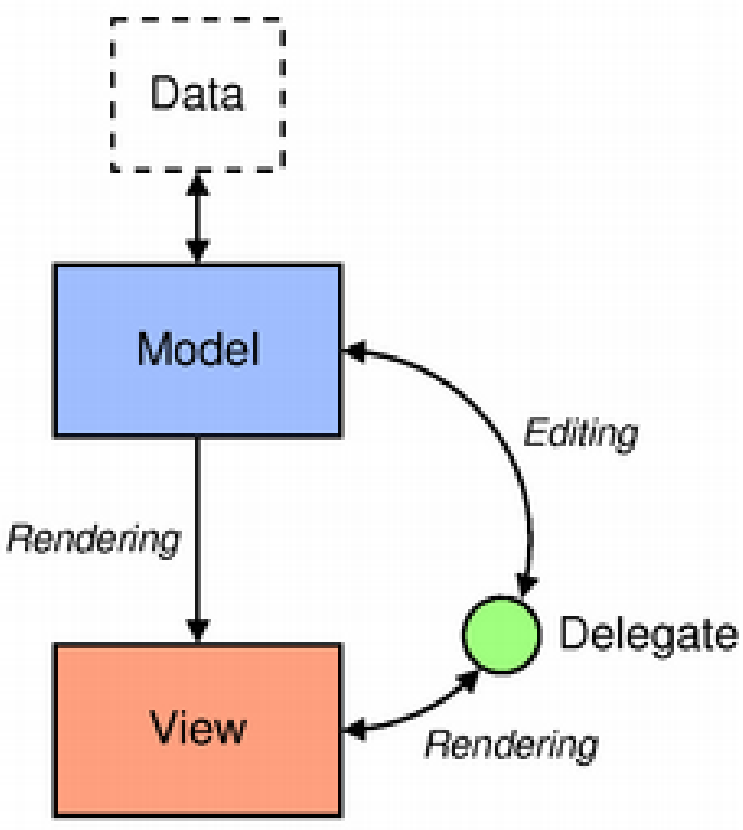
- Main Window
  - **Table View:** for displaying the wallet entries

Username	Domain	Password	Date	Description

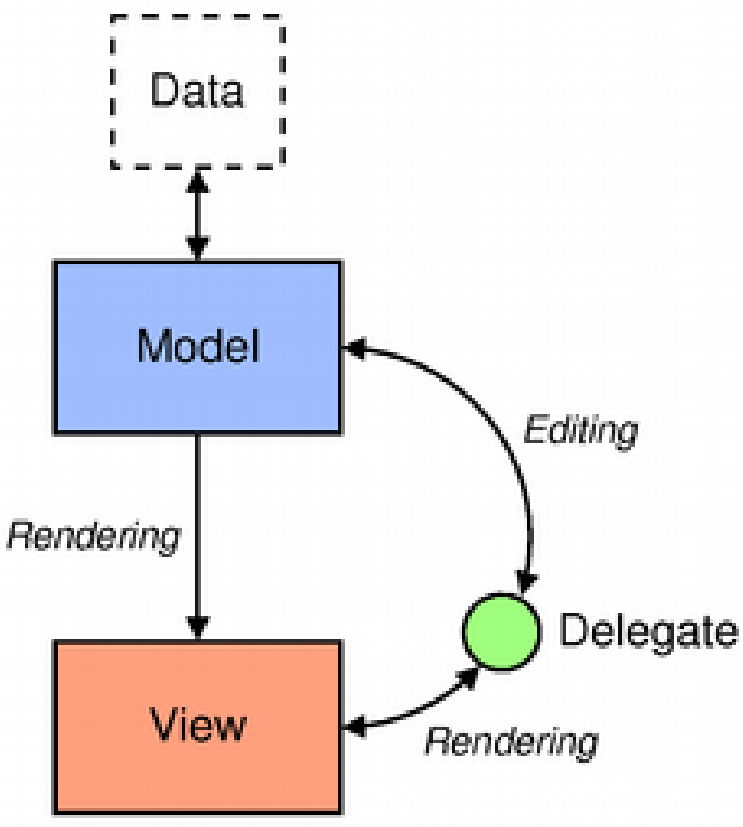
  - **Filters:** So the user can search in each of the table's columns.
  - **Tool Bars:** Open Wallet, Save Wallet, Add entry etc.
  - **Status Bar:** Used to display messages and the wallet's name
- Preference Window
  - Password Generators Configuration.
  - zxcvbn Configuration.
- Help Window
- Environment Window

Data Display: Model/View architecture

Data Display: Model/View architecture

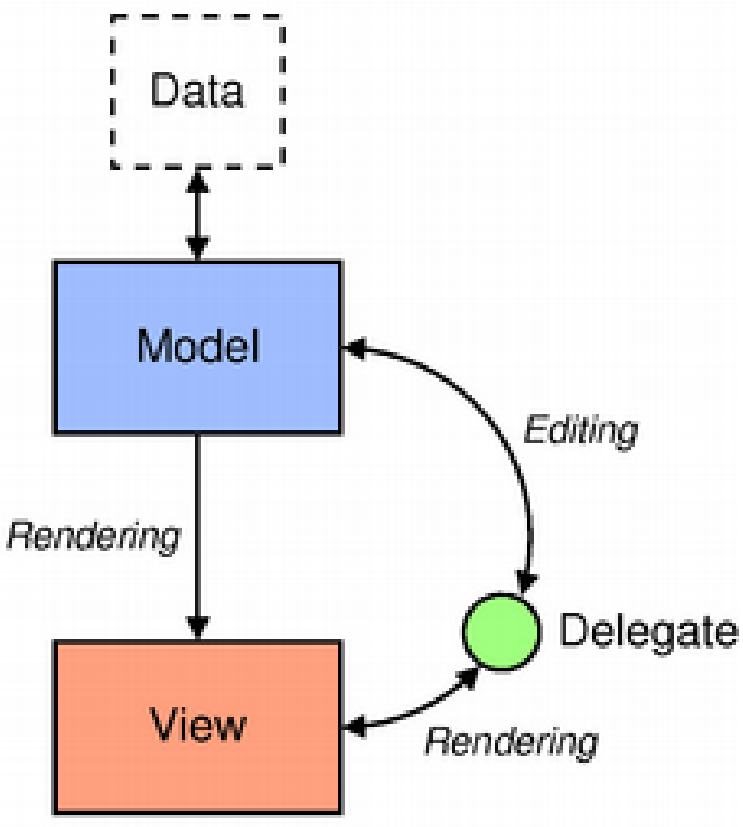


Data Display: Model/View architecture



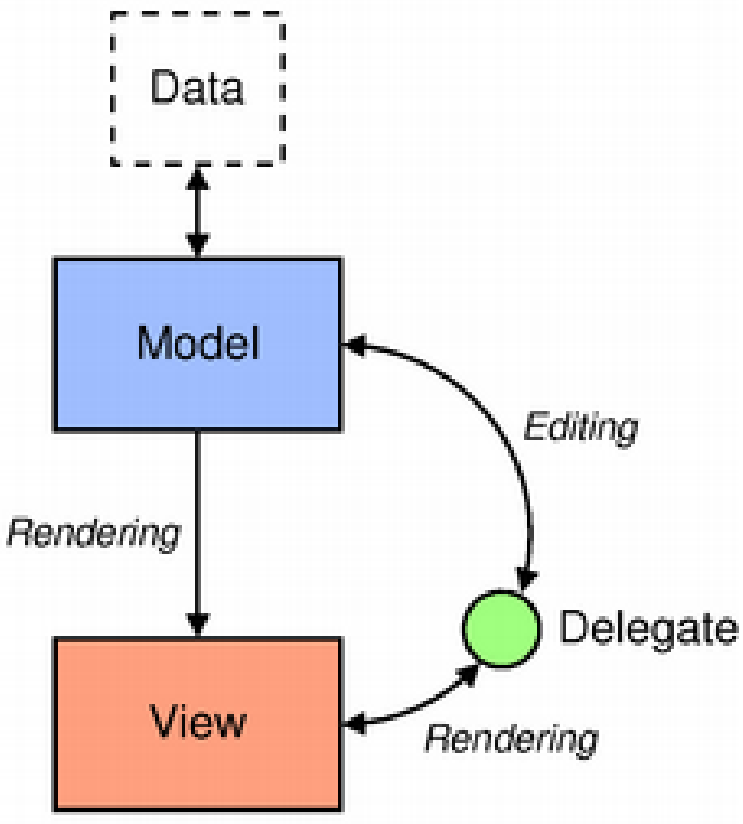
- **Data:** Entries in a table from the In-memory DB.

Data Display: Model/View architecture



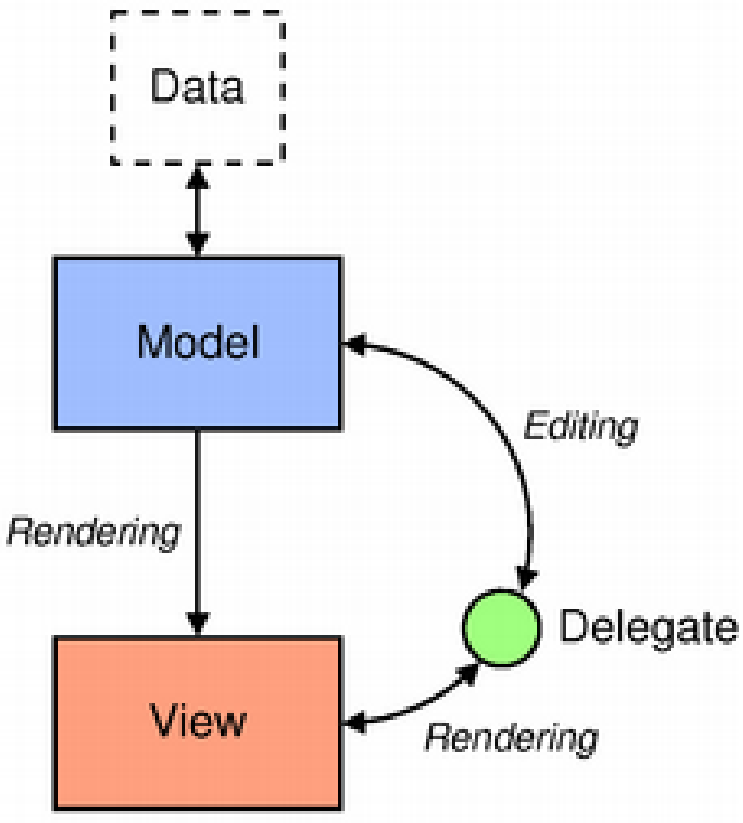
- **Data:** Entries in a table from the In-memory DB.
- **Model**: Wrapper for handling SQLite DBs easily.

Data Display: Model/View architecture



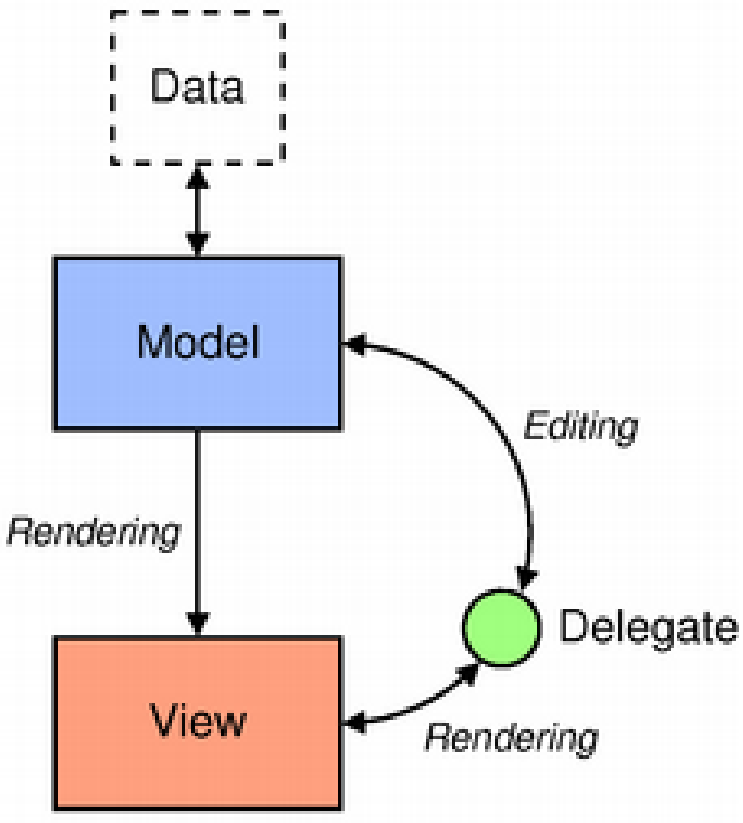
- **Data:** Entries in a table from the In-memory DB.
- **Model**: Wrapper for handling SQLite DBs easily.
- **View**: Data is displayed as a table.

Data Display: Model/View architecture



- **Data:** Entries in a table from the In-memory DB.
- **Model**: Wrapper for handling SQLite DBs easily.
- **Proxy Model**: Custom filtering for each column.
- **View**: Data is displayed as a table.

Data Display: Model/View architecture



- **Data:** Entries in a table from the In-memory DB.
- **Model**: Wrapper for handling SQLite DBs easily.
- **Proxy Model**: Custom filtering for each column.
- **View**: Data is displayed as a table.
- **Delegate**: Used to Show/Hide the passwords.

SW Libraries

SW Libraries

zxcvbn: Password strength estimator

- C/C++ open sources.
- Huge dictionary files need to be compiled into the sources
- Included in the project as a **Dynamic Library**

SW Libraries

zxcvbn: Password strength estimator

- C/C++ open sources.
- Huge dictionary files need to be compiled into the sources
- Included in the project as a **Dynamic Library**

PwGen: Pronounceable Passwords Generator

The sources were slightly simplified and included in the application.

SW Libraries

zxcvbn: Password strength estimator

- C/C++ open sources.
- Huge dictionary files need to be compiled into the sources
- Included in the project as a **Dynamic Library**

PwGen: Pronounceable Passwords Generator

The sources were slightly simplified and included in the application.

PassPhrase Generator

- Implemented as a C++/Qt function.
- Works by extracting Random words from dictionaries.
- HorseBatteryShoeStudying

Other functionalities

Other functionalities

- **Search for expired passwords**
  Using custom filter for the date column

Other functionalities

- **Search for expired passwords**
  Using custom filter for the date column

- **l33t converter**
  From PenicuikCiting to 9en1cu1kC1t1ng.

Other functionalities

- **Search for expired passwords**
  Using custom filter for the date column

- **l33t converter**
  From PenicuikCiting to 9en1cu1kC1t1ng.

- **Launch entry's domain**
  In default OS web browser

Outline

Login and Open a Wallet

Generate and evaluate password

Outline

Conclusions

- The SEcube™ is perfect for this type of application as it offers both reliability and simplicity to use.

Conclusions

- The SEcube™ is perfect for this type of application as it offers both reliability and simplicity to use.
- In any password manager it is important to suggest random passwords and to check their strength

Conclusions

- The SEcube™ is perfect for this type of application as it offers both reliability and simplicity to use.
- In any password manager it is important to suggest random passwords and to check their strength
- All the used libraries in this project are open source, proving it is possible to achieve a high level of security with the use of open software and hardware tools.

Conclusions

- The SEcube™ is perfect for this type of application as it offers both reliability and simplicity to use.
- In any password manager it is important to suggest random passwords and to check their strength
- All the used libraries in this project are open source, proving it is possible to achieve a high level of security with the use of open software and hardware tools.
- The developed application still lacks some features in order to be considered a truly commercial product.

Outline

1. Introduction
2. Technologies used
   - Software libraries
   - The SEcube™ Framework
3. Design and implementation
   - Basic SEcube™ Operation
   - General Architecture
   - Implementation details
4. Demos
5. Conclusions
6. **Future Work**

Future Work

Future Work

Web Browse Integration

- Develop a web browser complement.
- Use keyboard emulation to Auto Fill forms

Future Work

Web Browse Integration

- Develop a web browser complement.
- Use keyboard emulation to Auto Fill forms

More than static passwords

- OTP (One Time Passwords)
- U2F (Universal second factor)

Future Work

Web Browse Integration

- Develop a web browser complement.
- Use keyboard emulation to Auto Fill forms

More than static passwords

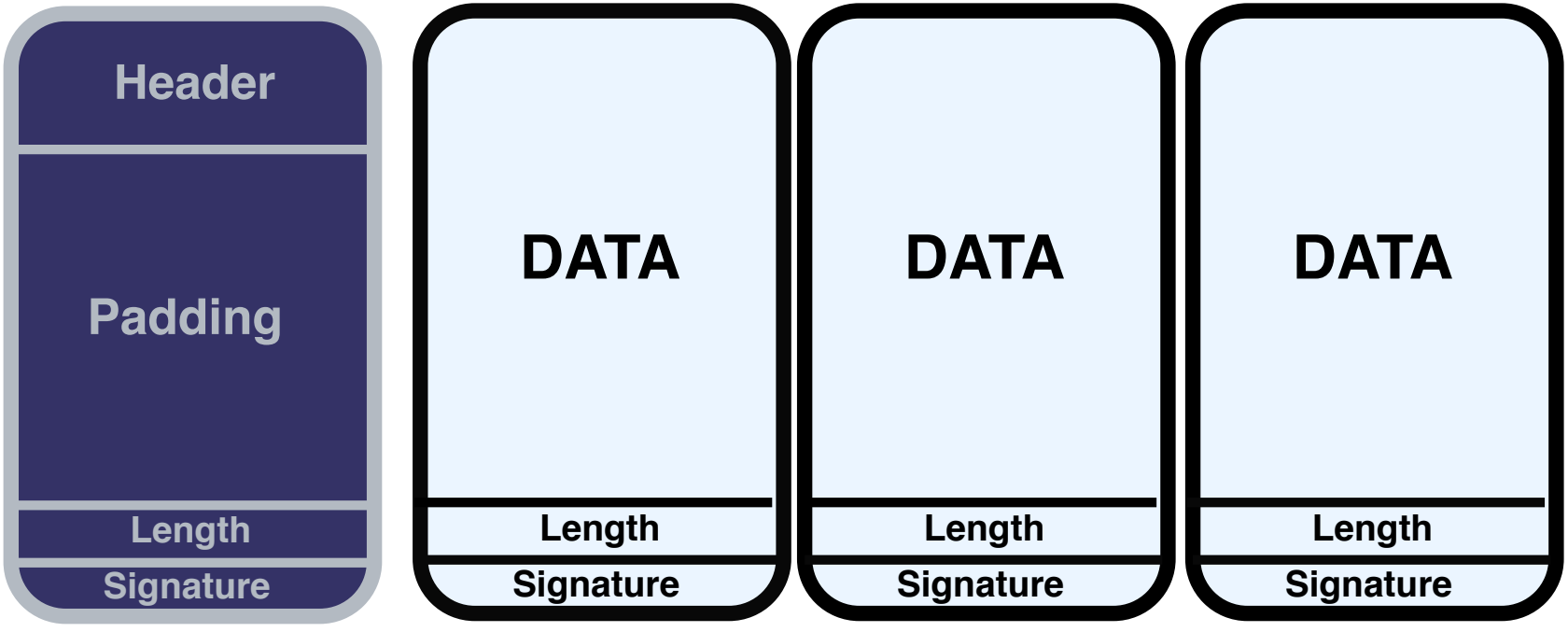
- OTP (One Time Passwords)
- U2F (Universal second factor)

Android

- Use a SEcube™ phone device
- Port SEcube™ host-side libraries to android
- Port Qt application to android

SEfile

A secured file has the structure shown in figure. The data is divided in sectors, and each of them is encrypted and signed. The first sector (header) contains metadata. When a portion of the file wants to be read or written, it is not necessary to process the whole file. Only the required sectors are manipulated.



Algorithms

Encryption: Advanced Encryption Standard (AES), established by the U.S. National Institute of Standards and Technology (NIST). For each data sector AES-256-CTR is used, while the header sector is encrypted using AES-256-ECB.

Authentication Each sector, including the header, is signed using SHA-256-HMAC, meaning that the signature depends on both the data contained in the sector itself and on a chosen encryption key.

To use two different keys to encrypt data and to digest authentication. SEfile leverages on the pbkdf2()

secureSQLite

Based on the SQLite and SEfile, this API allows the user to create SEcube™ secured data bases.

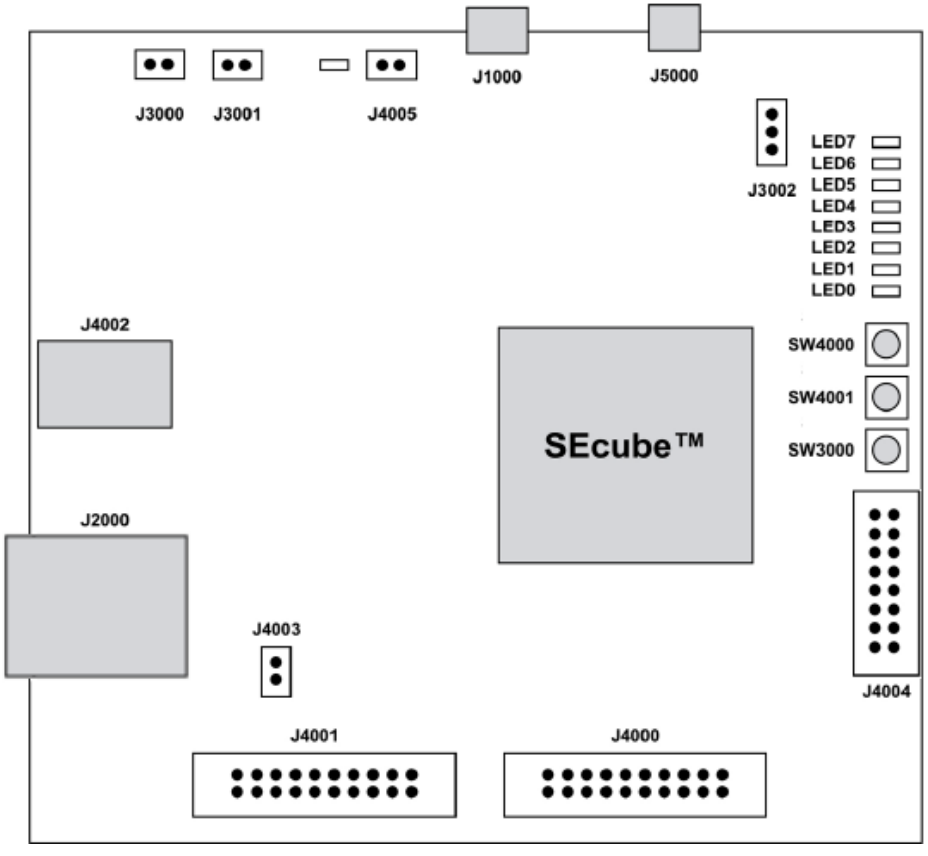
The SQLite system has been modified to use a wrapper based on SEfile, rather than using directly the OS calls. The development is based on a template for making a custom VFS interface distributed along with SQLite.

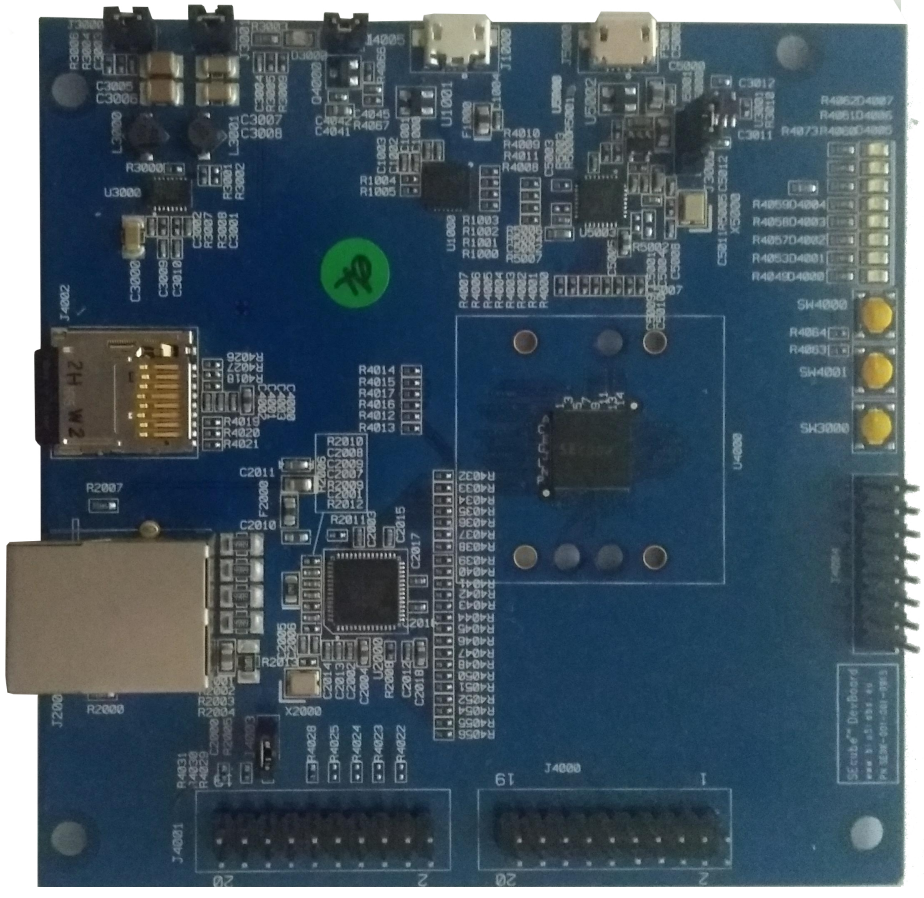
Every database created with secureSQLite is cyphered and signed, thus making it impossible to read the database contents without the SEcube™.

Development Board

J1000 USB 2.0 to UART
J2000 Ethernet 10/100
J4000 FPGA and CPU GPIOs
J4001 JTAG
J4002 microSD card

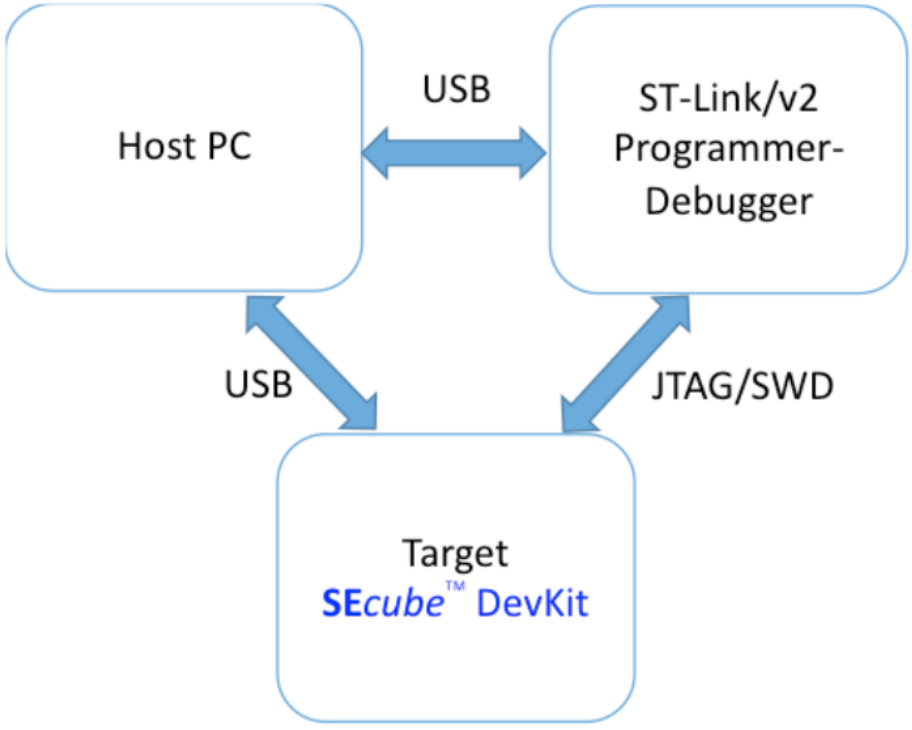
J4004 FPGA and CPU GPIOs
J5000 USB 2.0 High Speed
LEDx Leds
SWx00y Switches





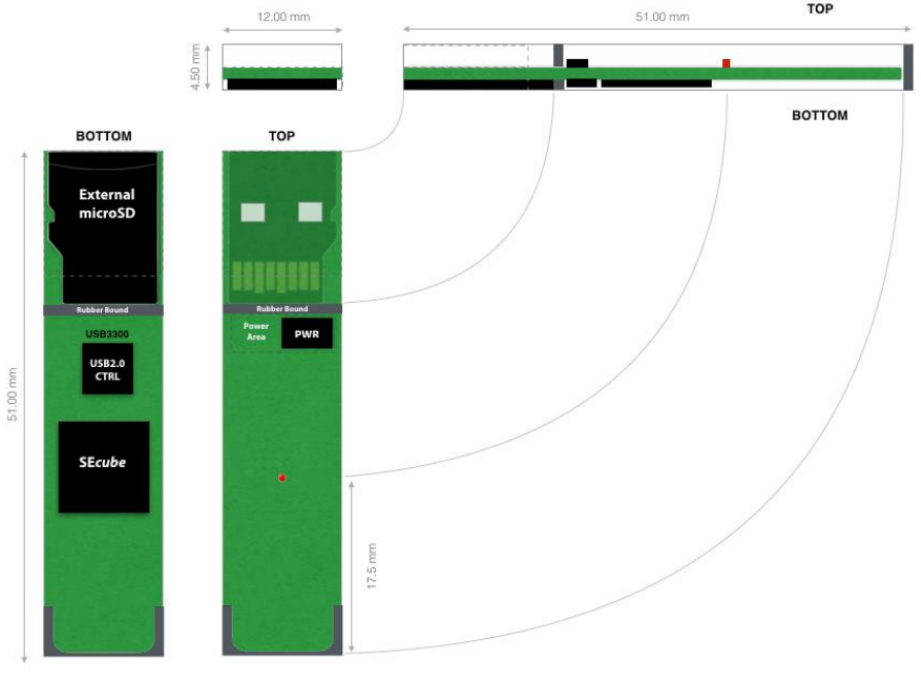
JTAG Connections

To program and debug the chip, we use an **ST-Link/V2** which communicates with the MCU using the JTAG/SWD connection in the DevKit board.



The USEcube™ Stick

- SEcube™ chip + USB 2.0 High-Speed + SDcard socket.
- Compatible with any Operating System, and no need for drivers.
- Separation of encrypted data from the encryptor/decryptor.
- microSD can be changed to adjust size and speed.
- Dust and water-resistant
- No JTAG interface. To inject firmware, secure bootloader.



YubiKey

Family of hardware authentication devices developed by Yubico

Unfortunately, not open source.

Supports Google Accounts, Facebook Accounts, GitHub, Dropbox

- Static Passwords
- Yubico One-Time Password (OTP)
- OATH – HOTP (EVENT)
- OATH – TOTP (TIME)
- Challenge and Response (HMAC-SHA1, Yubico OTP)
- PIV-Compatible Smart Card:
- OpenPGP
- FIDO U2F

Mooltipass: A Simple Offline Password Keeper

The Mooltipass emulates a standard USB keyboard, The Mooltipass has an internal flash in which the user encrypted credentials are stored, while a PIN-locked smartcard contains the AES-256bits key required for their decryption. Open software and open hardware, kickstarter campaign.

Mooltipass: A Simple Offline Password Keeper

The Mooltipass emulates a standard USB keyboard, The Mooltipass has an internal flash in which the user encrypted credentials are stored, while a PIN-locked smartcard contains the AES-256bits key required for their decryption. Open software and open hardware, kickstarter campaign.

- Plugin the Mooltipass, no driver required.

Mooltipass: A Simple Offline Password Keeper

The Mooltipass emulates a standard USB keyboard, The Mooltipass has an internal flash in which the user encrypted credentials are stored, while a PIN-locked smartcard contains the AES-256bits key required for their decryption. Open software and open hardware, kickstarter campaign.

- Plugin the Mooltipass, no driver required.
- Insert the smartcard and unlock it with the PIN

Mooltipass: A Simple Offline Password Keeper

The Mooltipass emulates a standard USB keyboard, The Mooltipass has an internal flash in which the user encrypted credentials are stored, while a PIN-locked smartcard contains the AES-256bits key required for their decryption. Open software and open hardware, kickstarter campaign.

- Plugin the Mooltipass, no driver required.
- Insert the smartcard and unlock it with the PIN
- If using the browser plugin, the Mooltipass asks permission to send the stored credentials, or asks you to save new ones

Mooltipass: A Simple Offline Password Keeper

The Mooltipass emulates a standard USB keyboard, The Mooltipass has an internal flash in which the user encrypted credentials are stored, while a PIN-locked smartcard contains the AES-256bits key required for their decryption. Open software and open hardware, kickstarter campaign.

- Plugin the Mooltipass, no driver required.
- Insert the smartcard and unlock it with the PIN
- If using the browser plugin, the Mooltipass asks permission to send the stored credentials, or asks you to save new ones
- If not using the browser plugin Mooltipass can type credentials like a keyboard.

Mooltipass: A Simple Offline Password Keeper

The Mooltipass emulates a standard USB keyboard, The Mooltipass has an internal flash in which the user encrypted credentials are stored, while a PIN-locked smartcard contains the AES-256bits key required for their decryption. Open software and open hardware, kickstarter campaign.

- Plugin the Mooltipass, no driver required.
- Insert the smartcard and unlock it with the PIN
- If using the browser plugin, the Mooltipass asks permission to send the stored credentials, or asks you to save new ones
- If not using the browser plugin Mooltipass can type credentials like a keyboard.

- **ST662ACD-TR:** Power
- **ATMEGA32U4:** MCU
- **AT88SC102:** Smart Card
- **AT45DB011D-SSH-T:** Flash

PwGen Examples

`-s`: Random
`-y`: Symbols
`-v`: No vowels
`-B`: No ambiguous
`-A`: No capital
`-0`: No numerals

Password	Length	Options	Log Entropy
`iesohGhai3`	10	-	9.75 (Level 3)
`ees0cooLo2`	10	-	10.47 (Level 4)
`dX042wKqlW`	10	`-s`	17.86 (Level 4)
`@!,Q*l5}+H`	10	`-ys`	18.15 (Level 4)
`TBw4)9`	6	`-ys`	11.62 (Level 4)
`B7t34Lck`	8	`-v`	11.87 (Level 4)
`nofosootei`	10	`-BA0`	6.50 (Level 2)

PassPhrase Generator examples

PassPhrase	wor	len	uncom	Log Entr
Cocchio	1	-	-	4.27 (L1)
Legitimately	1	8	-	4.55 (L1)
Woodhaven	1	8	30%	4.94 (L1)
ShorelineCech	2	-	-	9.18 (L3)
MongoliaSimpsons	2	8	-	7.30 (L2)
McinnisPhaya	2	-	30%	9.14 (L3)
ZucchiniSalamandra	2	8	30%	9.19 (L3)
SacchettiVigevano	2	8	30%	9.11 (L3)
DrammaturgicoSbatacchiare	2	12	-	8.98 (L3)
MalformationsAstrophysical	2	12	-	9.60 (L3)
LatinaInterchangeFbo	3	-	-	13.5 (L4)
OsaAymanCantinflas	3	-	-	12.98 (L4)
ImmobileCwSites	3	-	-	11.43 (L4)
RimmelBragFaenza	3	-	30%	13.49 (L4)
RecliningCanberraEcuadorian	3	8	-	13.69 (L4)
InaspettatoRothschildsDisconcerting	3	8	30%	14.48 (L4)

Other contributions

Besides the application development, other results obtained during this work are:

- The implementation of an improved Login behaviour in the SEcube™ framework, that renders more usable SEcubeWallet and any other application that uses the SEcube™ authentication system.
- The discovery and fix of a bug in the SEfile library that did not allow to use the secureSQLite library in a FAT32 file system.

SEkey: key management for SEcube™

SEkey is a new library currently under development by Mateus Françani as his master thesis work. The library will sit next to SEfile and SElink.

Right now keys inside a SEcube™ chip can only be modified at factory reset. This is not very useful in a working environment, as the purpose of having multiple keys is to allow users to share information with selected people.

The job of the SEkey library will be to allow an admin to dynamically add and remove keys to SEcube™ devices.