

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

1. Introduction

2. Technologies used

- ▶ Software libraries
- ▶ The SEcube™ Framework

3. Design and implementation

4. Results

- ▶ Demos

5. Conclusions

6. Future Work

Outline

1. Introduction

2. Technologies used

- ▶ Software libraries
- ▶ The SEcube™ Framework

3. Design and implementation

4. Results

- ▶ Demos

5. Conclusions

6. Future Work

Introduction

This work regards the implementation as a desktop application that exploits the capabilities of the SEcube™ (Secure Environment cube) hardware and software framework to store and protect passwords.

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

4. Results

- ▶ 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: GUI and wrappers

Software Libraries

The following open source libraries were used:

Qt: GUI and wrappers

C++ library, cross-platform, elegant design

Software Libraries

The following open source libraries were used:

Qt: GUI and wrappers

C++ library, cross-platform, elegant design

SQLite: DataBase management

Software Libraries

The following open source libraries were used:

Qt: GUI and wrappers

C++ library, cross-platform, elegant design

SQLite: DataBase management

Self-contained, written in C, Transactional

Software Libraries

The following open source libraries were used:

Qt: GUI and wrappers

C++ library, cross-platform, elegant design

SQLite: DataBase management

Self-contained, written in C, Transactional

PwGen: Password generator

Software Libraries

The following open source libraries were used:

Qt: GUI and wrappers

C++ library, cross-platform, elegant design

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: GUI and wrappers

C++ library, cross-platform, elegant design

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: GUI and wrappers

C++ library, cross-platform, elegant design

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

Software

Developed by the Blu5
Group

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)

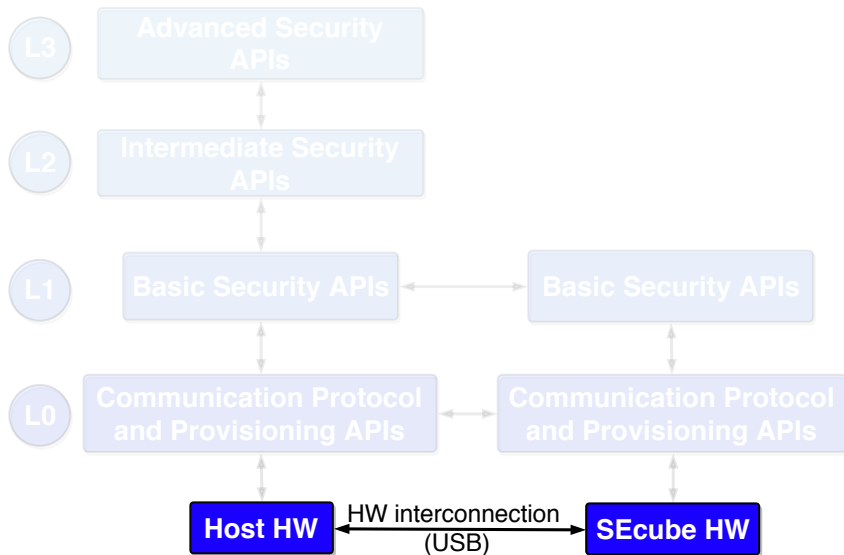
Software

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

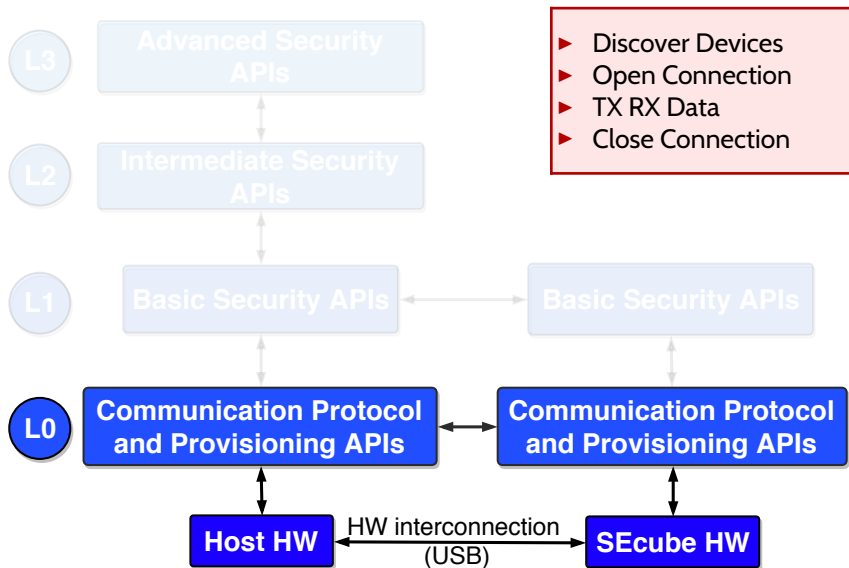
Firmware: Developers can customize the firmware to their needs, and load the updated version to the SEcube™ chip.

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

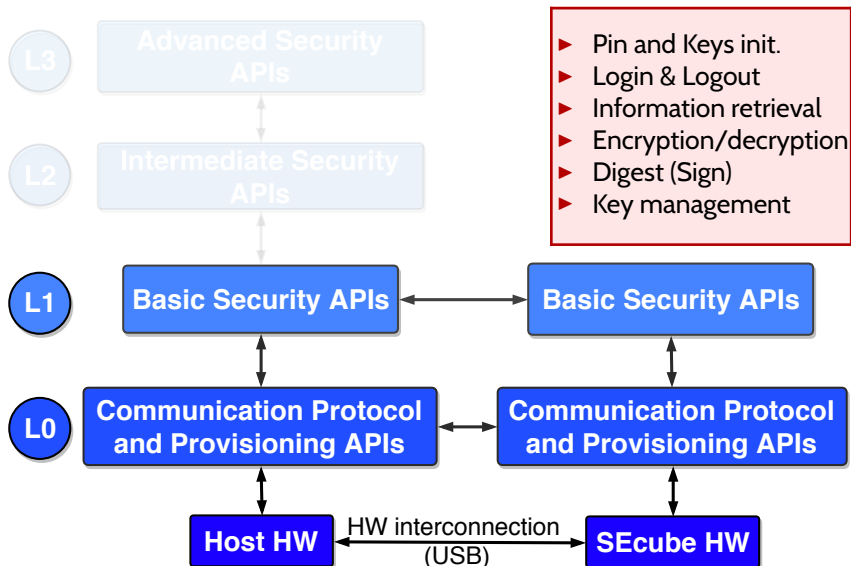
SEcube™ APIs hierarchy



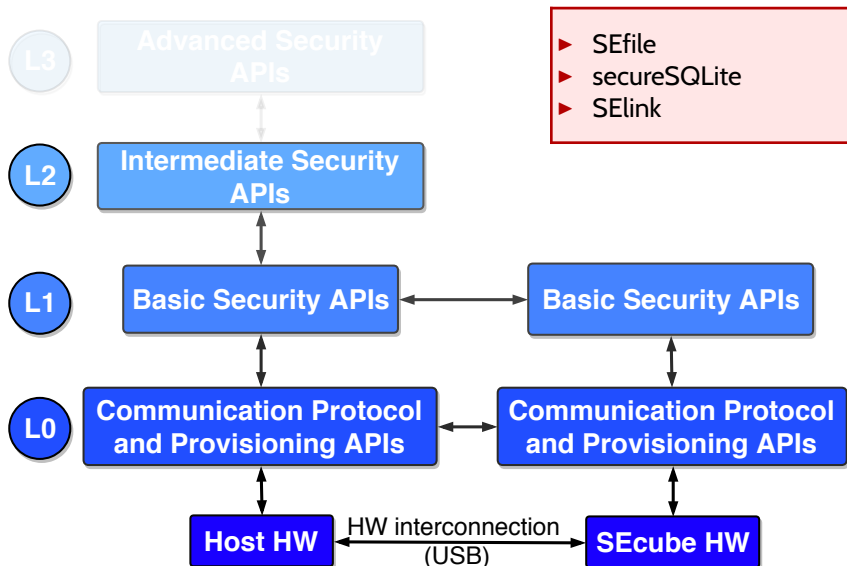
SEcube™ APIs hierarchy



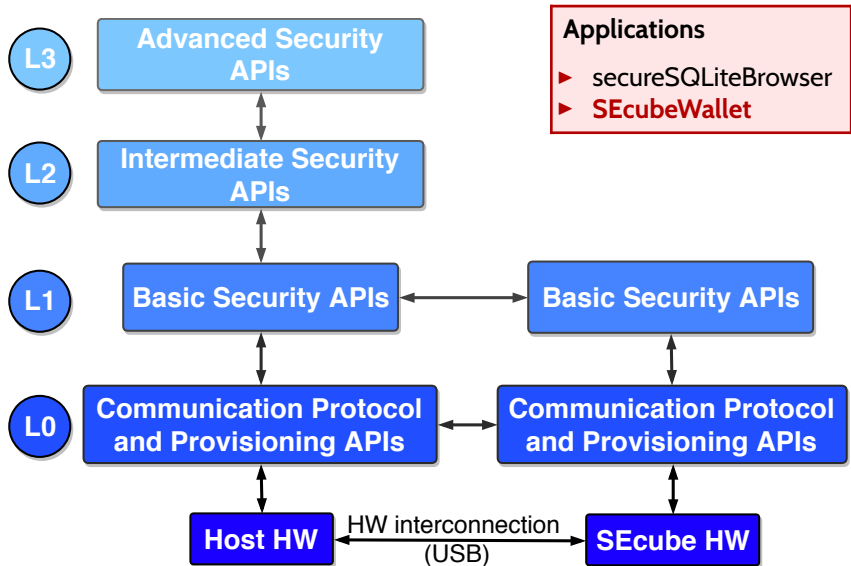
SEcube™ APIs hierarchy



SEcube™ APIs hierarchy



SEcube™ APIs hierarchy



Outline

1. Introduction

2. Technologies used

- ▶ Software libraries
- ▶ The SEcube™ Framework

3. Design and implementation

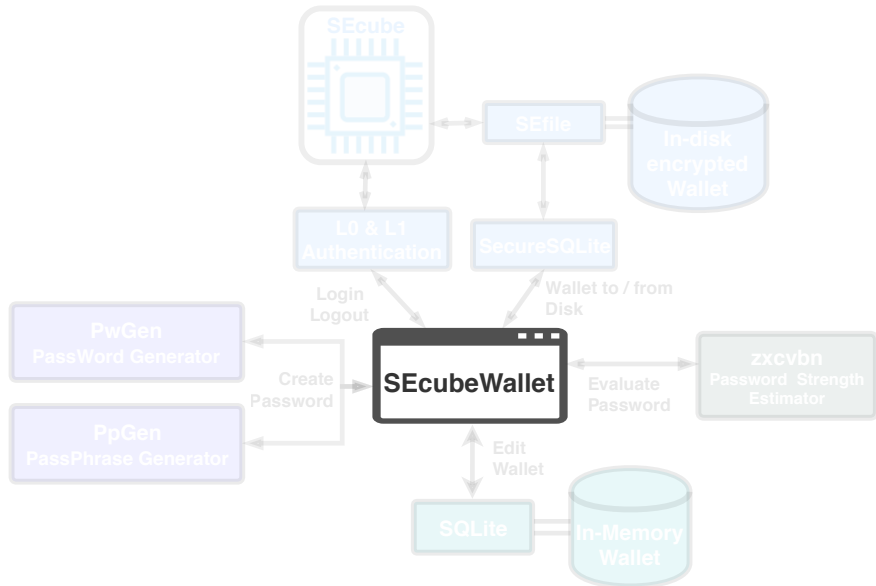
4. Results

- ▶ Demos

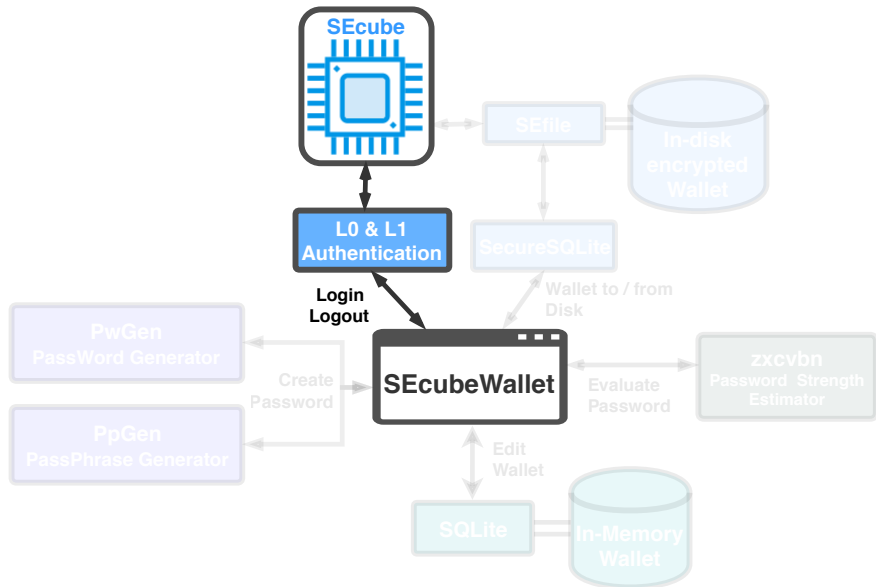
5. Conclusions

6. Future Work

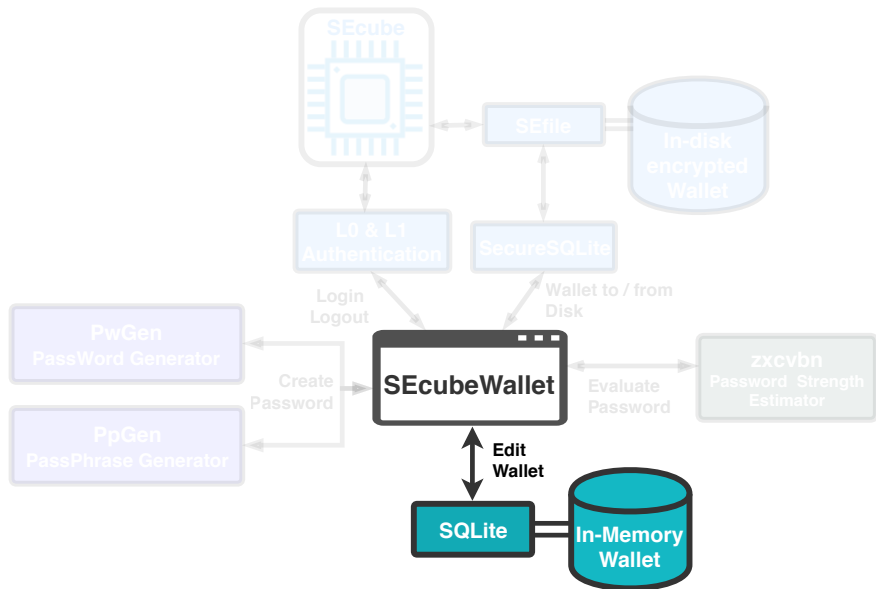
SEcubeWallet Application



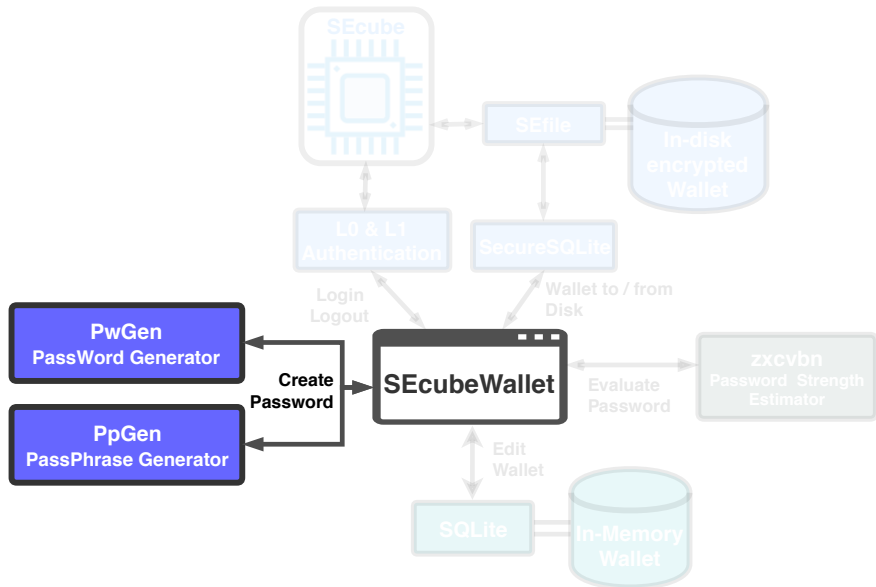
Open device and authenticate



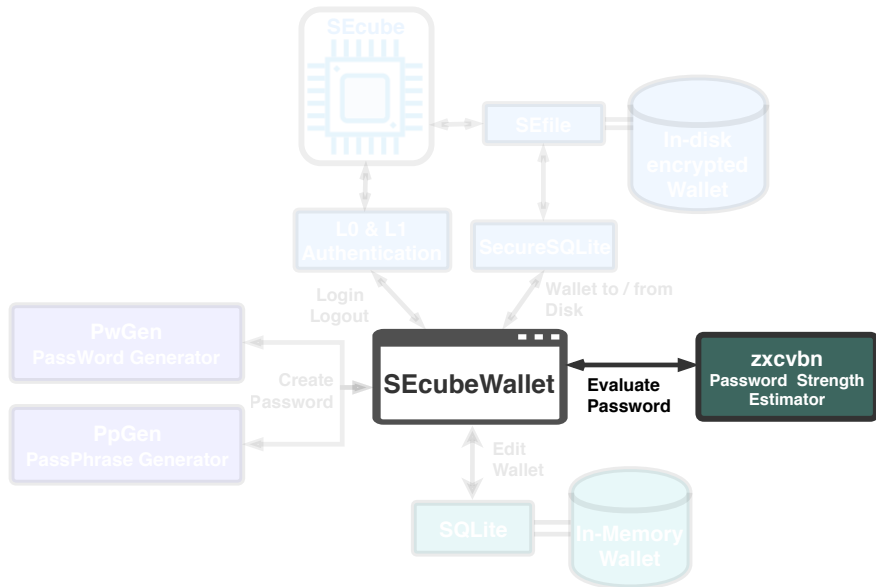
Create In-memory Wallet



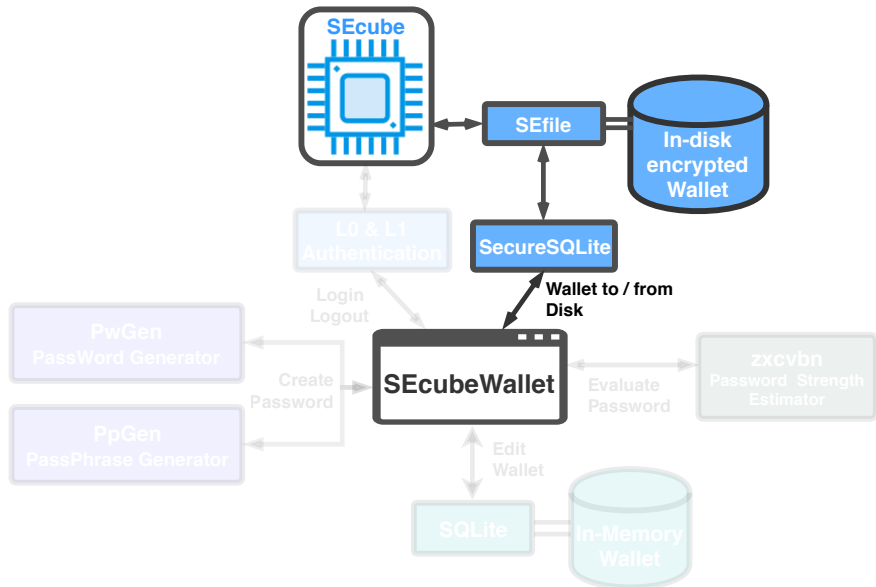
Generate Password/Passphrase



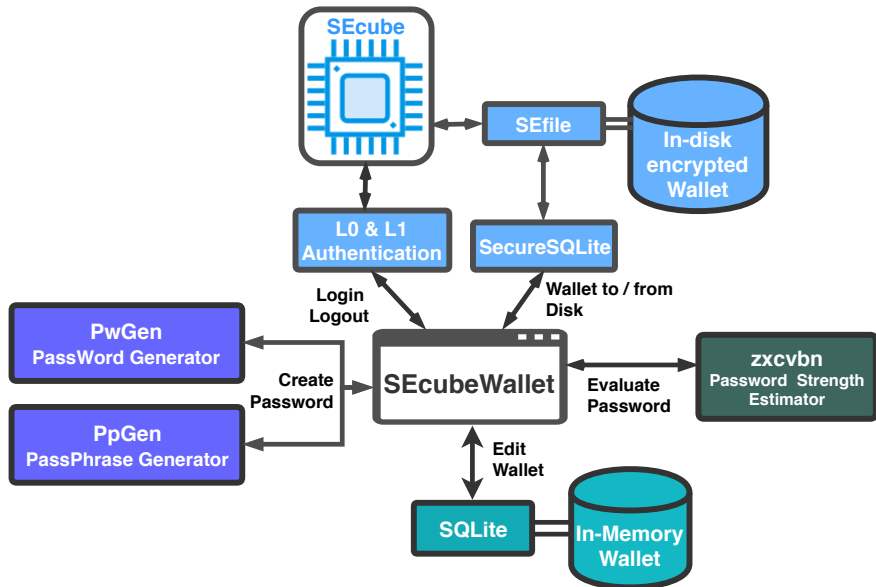
Evaluate Strength



Encrypt and Save Wallet to disk



General Architecture



Basics of Operation

Basics of Operation

- ▶ At factory initialization, an admin/developer writes to the SEcube™ flash memory:
 - ▶ Admin pin
 - ▶ User pin
 - ▶ User Keys (A single device can have multiple keys, and they can be used to share data with other SEcube™ users)

Basics of Operation

- ▶ At factory initialization, an admin/developer writes to the SEcube™ flash memory:
 - ▶ Admin pin
 - ▶ User pin
 - ▶ User Keys (A single device can have multiple keys, and they can be used to share data with other SEcube™ users)
- ▶ User logs in with its pin and a challenge-based authentication.

Basics of Operation

- ▶ At factory initialization, an admin/developer writes to the SEcube™ flash memory:
 - ▶ Admin pin
 - ▶ User pin
 - ▶ User Keys (A single device can have multiple keys, and they can be used to share data with other SEcube™ users)
- ▶ User logs in with its pin and a challenge-based authentication.
- ▶ User chooses which of the keys to use to encrypt/decrypt.

Basics of Operation

- ▶ At factory initialization, an admin/developer writes to the SEcube™ flash memory:
 - ▶ Admin pin
 - ▶ User pin
 - ▶ User Keys (A single device can have multiple keys, and they can be used to share data with other SEcube™ users)
- ▶ User logs in with its pin and a challenge-based authentication.
- ▶ User chooses which of the keys to use to encrypt/decrypt.

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.

In-Memory and In-Disk DBs

In-Memory and In-Disk DBs

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

In-Memory and In-Disk DBs

- ▶ **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

- ▶ **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

- ▶ **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

Windows and display elements

- ▶ Main Window
 - ▶ **Table View** for displaying the wallet entries
 - ▶ **Filters** So the user can search in each of the table's columns.
 - ▶ **Tool Bars** for Wallets, Tables and Entries.
 - ▶ **Menu Bar** with all the actions.
 - ▶ **Status Bar** used to display messages and the wallet's name

Windows and display elements

- ▶ Main Window
 - ▶ **Table View** for displaying the wallet entries
 - ▶ **Filters** So the user can search in each of the table's columns.
 - ▶ **Tool Bars** for Wallets, Tables and Entries.
 - ▶ **Menu Bar** with all the actions.
 - ▶ **Status Bar** used to display messages and the wallet's name
- ▶ Preference Window
 - ▶ Password Generators Configuration.
 - ▶ zxcvbn Configuration.

Windows and display elements

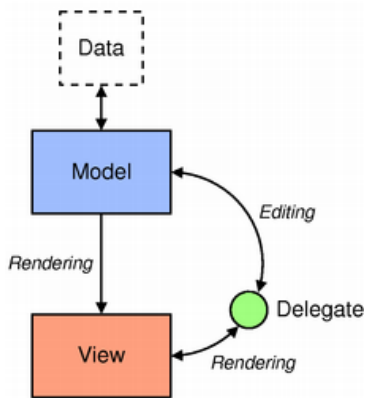
- ▶ Main Window
 - ▶ **Table View** for displaying the wallet entries
 - ▶ **Filters** So the user can search in each of the table's columns.
 - ▶ **Tool Bars** for Wallets, Tables and Entries.
 - ▶ **Menu Bar** with all the actions.
 - ▶ **Status Bar** used to display messages and the wallet's name
- ▶ Preference Window
 - ▶ Password Generators Configuration.
 - ▶ zxcvbn Configuration.
- ▶ Help Window

Windows and display elements

- ▶ Main Window
 - ▶ **Table View** for displaying the wallet entries
 - ▶ **Filters** So the user can search in each of the table's columns.
 - ▶ **Tool Bars** for Wallets, Tables and Entries.
 - ▶ **Menu Bar** with all the actions.
 - ▶ **Status Bar** used to display messages and the wallet's name
- ▶ Preference Window
 - ▶ Password Generators Configuration.
 - ▶ zxcvbn Configuration.
- ▶ Help Window
- ▶ Environment Window

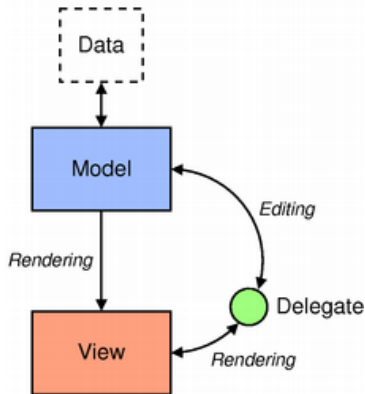
Data Display: *Model/View* architecture

Data Display: Model/View architecture



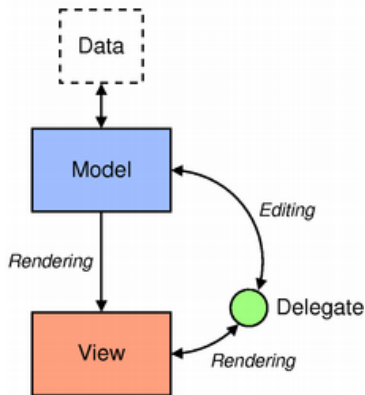
Data Display: Model/View architecture

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

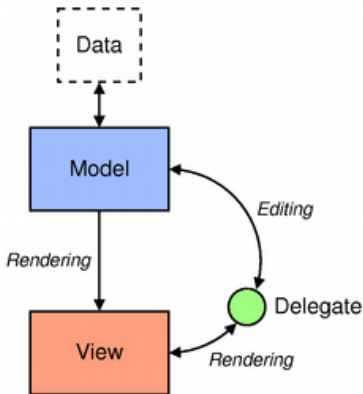


Data Display: Model/View architecture

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

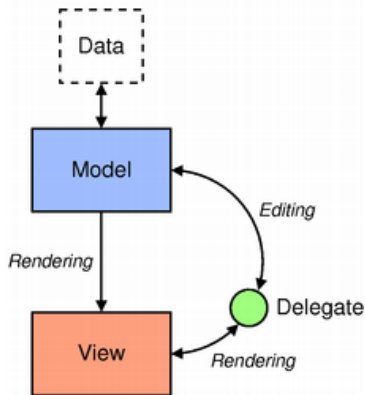


Data Display: Model/View architecture



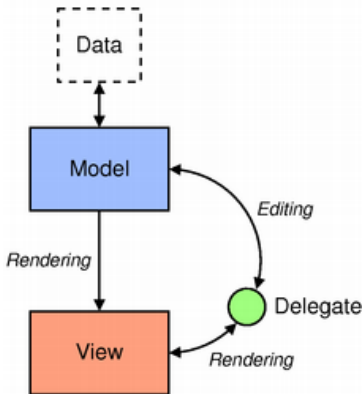
- ▶ **Data:** Entries in a table from the In-memory DB.
- ▶ **Model:** Wrapper for handling SQLite DBs easily.
- ▶ **Proxy Model:** Custom filtering for each column.

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.
- ▶ Included in the project as a **Dynamic Library**

SW Libraries

zxcvbn: Password strength estimator

- ▶ C/C++ open sources.
- ▶ Included in the project as a **Dynamic Library**

PwGen: Pronounceable Passwords Generator

Available as a Program, but instead the sources were directly included in the project.²

SW Libraries

zxcvbn: Password strength estimator

- ▶ C/C++ open sources.
- ▶ Included in the project as a **Dynamic Library**

PwGen: Pronounceable Passwords Generator

Available as a Program, but instead the sources were directly included in the project.²

PassPhrase Generator

- ▶ Implemented as a C++/Qt function.
- ▶ Works by extracting Random words out of dictionary files (plain text).

Other functionalities

Other functionalities

Other functionalities

- ▶ **Search for expired passwords**
Using date's column custom filter

Other functionalities

- ▶ **Search for expired passwords**
Using date's column custom filter
- ▶ **Launch entry's domain**
In default OS web browser

Other functionalities

- ▶ **Search for expired passwords**
Using date's column custom filter
- ▶ **Launch entry's domain**
In default OS web browser
- ▶ **l33t converter**
From **PenicuikCiting** to **9en1cu1kC1t1ng**.

Outline

1. Introduction

2. Technologies used

- ▶ Software libraries
- ▶ The SEcube™ Framework

3. Design and implementation

4. Results

- ▶ Demos

5. Conclusions

6. Future Work

Login and Open a Wallet



Generate and evaluate password



Outline

1. Introduction

2. Technologies used

- ▶ Software libraries
- ▶ The SEcube™ Framework

3. Design and implementation

4. Results

- ▶ Demos

5. Conclusions

6. Future Work

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

4. Results

- ▶ Demos

5. Conclusions

6. Future Work

Future Work

Future Work

Web Browse Integration

- ▶ Port the entire Application to a web browser complement.
- ▶ Web browser complement that "talks" with the SEcubeWallet
- ▶ Use keyboard emulation to Auto Fill forms

Future Work

Web Browse Integration

- ▶ Port the entire Application to a web browser complement.
- ▶ Web browser complement that "talks" with the SEcubeWallet
- ▶ Use keyboard emulation to Auto Fill forms

More than static passwords

- ▶ One Time Passwords
- ▶ FIDO U2F

Future Work

Web Browse Integration

- ▶ Port the entire Application to a web browser complement.
- ▶ Web browser complement that "talks" with the SEcubeWallet
- ▶ Use keyboard emulation to Auto Fill forms

More than static passwords

- ▶ One Time Passwords
- ▶ FIDO U2F

Android

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