secure-control-protocol - Work in progress

Table of contents

- secure-control-protocol - Work in progress
  - Table of contents
  - Hint
  - 0. About
    - 0.1 Purpose
    - 0.2 Goal
  - 1. Architecture
  - 2. Provisioning of devices
  - 3. Discovery of devices
  - 4. Security
  - 5. HTTP Ressources
  - 6. REST Message Types
    - 6.1 Discover message types
      - 6.1.1 discover-hello
        - Variables
    - 6.2 Control messages
      - 6.2.1 control-up
      - 6.2.2 control-down
      - 6.2.3 control-stop
      - 6.2.4 control-status
    - 6.3 Security messages
      - 6.3.1 security-fetch-nvcn
      - 6.3.2 security-pw-change
      - 6.3.3 security-wifi-config
      - 6.3.4 security-reset-to-default
      - 6.3.5 security-restart
  - 7. SCP Stack Software Architecture
    - 7.1 Used Libraries
    - 7.2 Class Diagrams
  - 8. Annex
    - 8.1 Default credentials
      - 8.1.1 Default device password
      - 8.1.2 Default Wifi Access Point credentials

        - 8.2 ESP8266 EEPROM Layout
    - Project Philosophy
    - License
    - Copyright

Hint

A PDF version of this README with all images is stored in the `./doc/` directory.

0. About

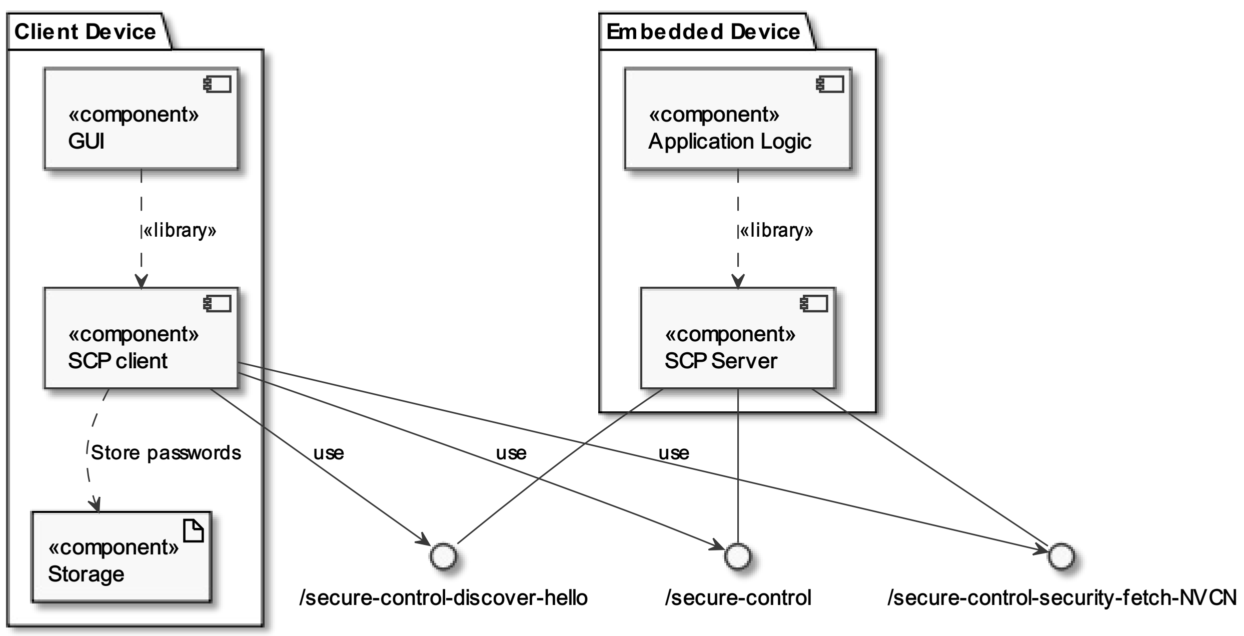
0.1 Purpose

The purpose of the Secure Control Protocol is to enable makers to develop home automation devices based on the ESP8266 with minimum effort while providing a substantial level of security compared to other solutions.

0.2 Goal

The goal is to provide a ready to use protocol and server where the user only has to register his own functions without a special need for configuration of the server.

1. Architecture



System Component Diagram

2. Provisioning of devices

When the default password of the secure-controller is set or no wifi credentials are provisioned the secure-controller provides a Wifi Access Point using WPA2-PSK which can be accessed with the default credentials from the annex.

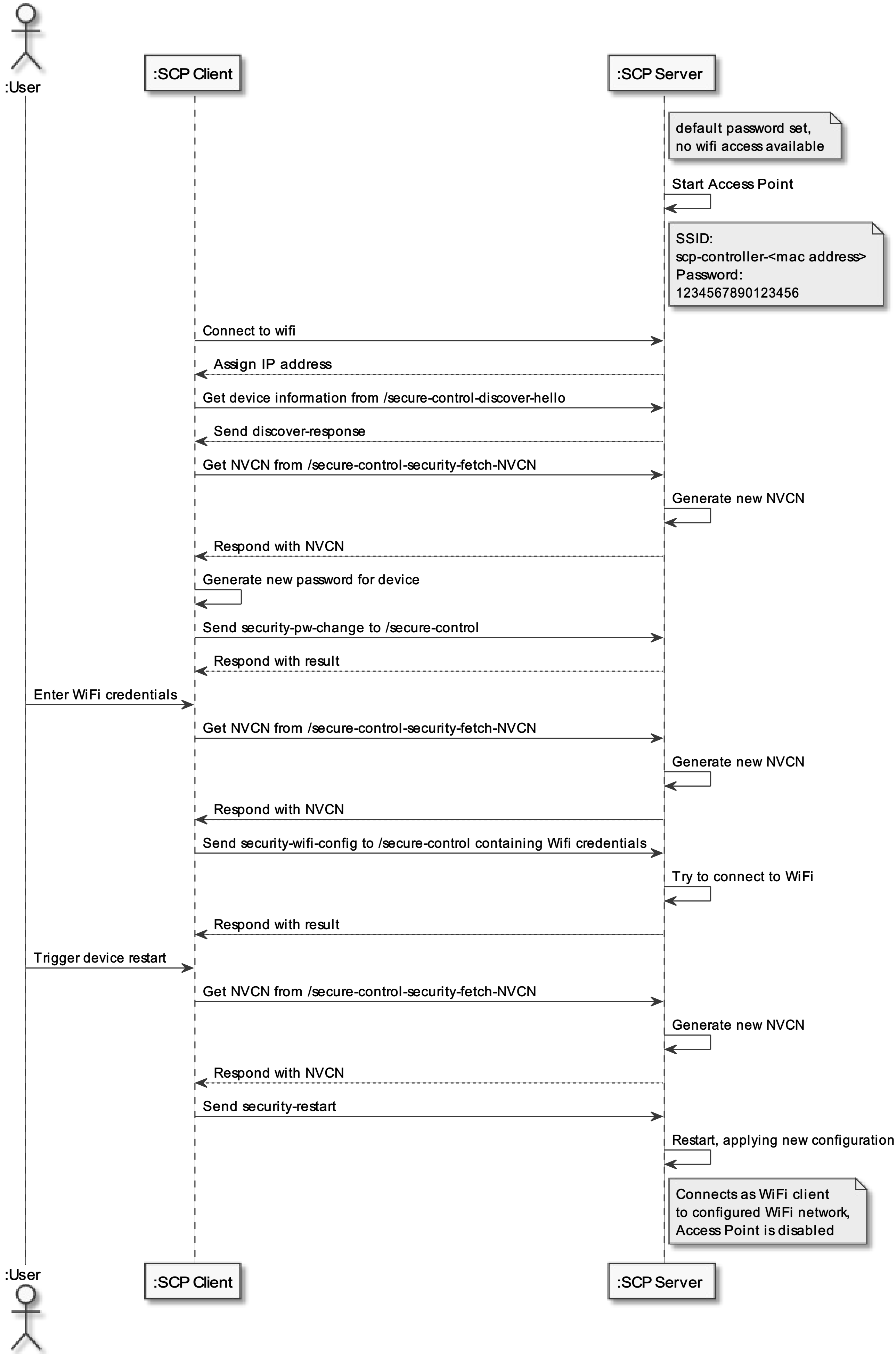
When the Wifi Access Point is available the control device connects to the wifi and the secure-controller acts as a DHCP server and provides an IP address from a small Class C IP subnet.

Now the control device can start the discovery of secure-controller in the IP subnet. If the secure-controller is found a new secure-control password must be set. This can be done via the security-pw-change message. As a second step the credentials of the home network wifi the secure-controller operates in should be supplied. The user sends a the security-wifi-config message containing the encrypted credentials to the secure-controller.

If a secure-controller receives a valid security-wifi-config message it tries to connect to the wifi and reponds with the result.

The third step is triggered by the security-reset message which restarts the secure-controller and thus applies the configured settings. If the secure-controller default password and the wifi credentials are changed / provisined, the secure-controller is started as in the wifi client only mode.

Note: If the connection to the supplied home network wifi fails, the secure-controller acts as a wifi access point in order to receive the new home network credentials. But in contrast to the beginning of this chapter the password for this wifi will now be the provisioned secure-controller password.



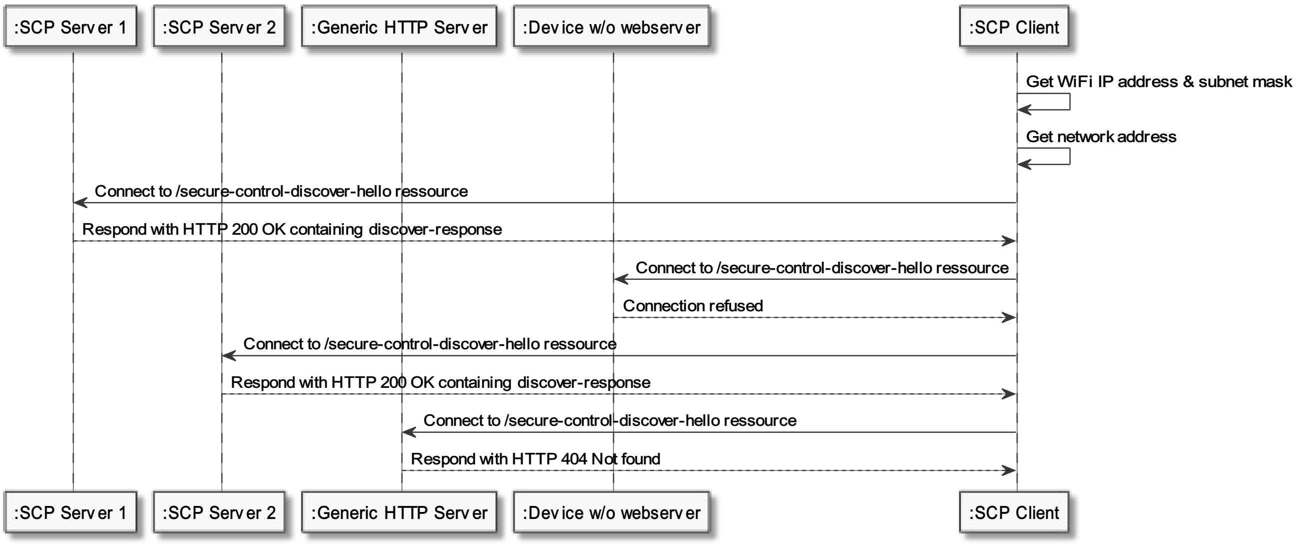
Provisioning Sequence Diagram

3. Discovery of devices

The Android app acts as a control device and is capable of discovering devices in the local subnet network range.

To do this the app connects to the secure-control-discover-hello ressource of each IP addresses of the configured IP address range.

The app stores the IP addresses of all devices which respond with a HTTP response 200 OK with information in the body.



Discover Sequence Diagram

4. Security

The security is based on pre-shared secrets. Each device has its own secret. The secret is only shared between the secure-controller and the control device(s).

Each secure-controller has a preconfigured password which has to be changed when the secure-controller is first accessed. The secure-controller does **not** accept any control messages or security messages (besides the security-pw-change message) if the current secure-controller password matches the default password from the annex.

The password has the be 16 characters long. This ensures a adequate security due to the large key space. The password will usually not be used by a human user. The secret is set by the controller device and sored in the secure-controller. Therefore the length limitation does not pose a usability problem.

All (except for 6.1.1 discover-hello) exchanged messages are encrypted. The method used is AES-128-CBC with the shared password, a NVCN and an initialization vector (IV).

To prevent attacks on the content / password of the exchanged messages an additional measure is taken. For every message an intitialzation vector is generated (a 128 bit random number). This number is generated by the sender of the message. It is added at the beginning of the message contents.

The NVCN shall prevent replay attacks and needs to be fetched from the secure-controller by the control device for every single message before encrypting it. The correctness of the supplied NVCN of each message is checked by the secure-controller.

Currently not covered:

No hardware attacks are currently considered. The flash memory of the secure-controller stores the secure-controller password which could be read and missused. This attack is considered to be unlikely and of limited use only since every device has a seperate password. Never the less any ideas / comments on this issue are very welcome.

5. HTTP Ressources

The device exposes the following HTTP ressources:

```
http://device-ip/secure-control
```

```
http://device-ip/secure-control/discover-hello
```

```
http://device-ip/secure-control/security-fetch-NVCN
```

6. REST Message Types

The secure-controller waits for HTTP-GET messages with the Content-Type application/x-www-form-urlencoded.

Almost all messages are encrypted, see security chapter for details on the algorithms and exceptions.

The initialization vector (NVCN) used for replay protection is randomly generated on secure-controller start-up. It is being fetched from the control device by using the security-fetch-NVCN message before sending a message to the secure-controller. The NVCN is incremented by the secure-controller after every security-fetch-NVCN message.

For all encrypted messages the following HTTP ressource is used:

```
http://device-ip/secure-control
```

The data send to the secure-controller is encoded in the payload parameter.

```
http://device-ip/secure-control?payload=encoded_data
```

The encoded_data consists of the base64 and urlencoded encrypted message.

```
encoded_data = urlencode(base64(encrypted_message))
```

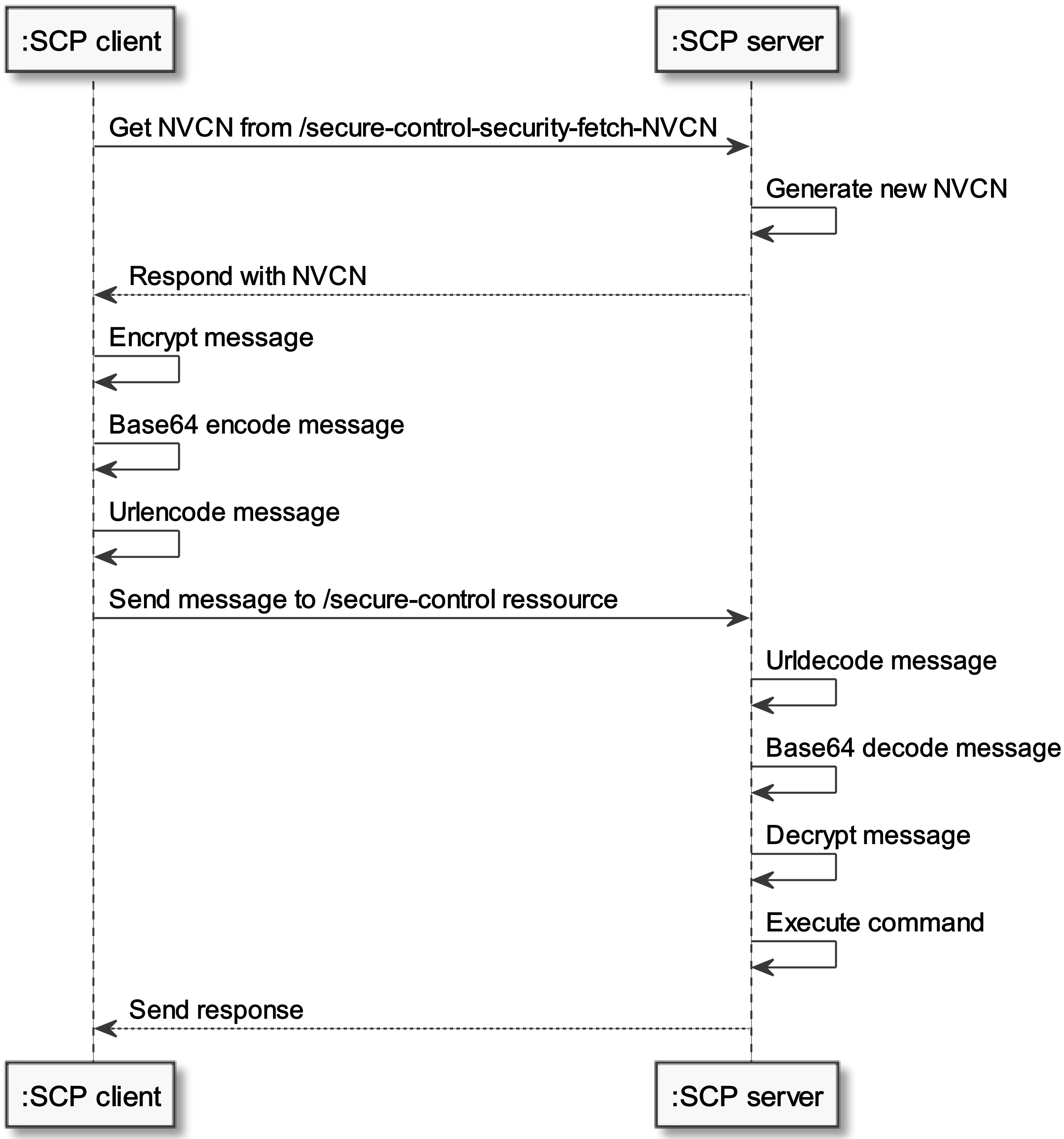
The encrypted_message consists of a IV, the device ID of the secure-controller, the NVCN and the type of the message. The NVCN, the device ID and the type are being concetenated. As seperator a colon is used. The message is then encrypted using the IV and the secure-controller password.

```
encrypted_message = IV + encrypt(NVCN + ":" + deviceID + ":" + message_type, IV, password)
```

All messages except for the discover-hello message, respond with a HTTP 200 OK message containing a JSON object with the encrypted payload:

Key	Possible values
payload	encrypted-payload

```
{
    "payload" : "encrypted-payload"
}
```



Message Processing Sequence Diagram

6.1 Discover message types

6.1.1 discover-hello

```
Ressource: http://device-ip/secure-control/discover-hello?payload=discover-hello
```

The discover-hello message is sent to all IP addresses of the home network subnet to determine which IP addresses beloang to a secure-controller. It is the only message being sent without encryption. If the device is a secure-controller it responds with a HTTP 200 OK message containing a JSON representation of the following information.

The HMAC is calculated as follows:

```
hmac = "discover-response" + deviceID + deviceType + IP Address + current password number
```

Variables

Key	Possible values
type	discover-response
device-id	device id (16 byte)
device-type	device type
current password number	number of password changes, 0 is default
hmac	Keyed-Hashed Massage Authentication Code

```
{
    "type" : "discover-response",
    "deviceId" : "device ID",
    "deviceType" : "secure-controller",
    "currentPasswordNumber" : number of password changes   ,
    "hmac" : Keyed-Hashed Massage Authentication Code
}
```

6.2 Control messages

6.2.1 control-up

Ressource:

```
http://device-ip/secure-control?payload=encoded_data
```

The control-up message tells the secure-controller to open.

The deviceID provided in the payload must match the configured device ID.

The NVCN provided in the payload must match (current_controller_NVCN-1).

The encoded_data payload is created according to REST message types and encoding using: message_type=control-up

The encrypted payload of the response consists of a JSON representation of the following data:

Key	Possible values
type	control-up
deviceId	device ID

status	neutral / up / error

```
{
    "type" : "control-up",
    "deviceId" : "device ID",
    "status" : "neutral" | "up" | "error"
}
```

The status vaules have the following meaning:

status	description
"neutral"	the power to the motor is not connected
"up"	the motor is turning towards upmost postition
"error"	some error occured

6.2.2 control-down

The control-down message tells the secure-controller to close.

The deviceID provided in the payload must match the configured device ID.

The NVCN provided in the payload must match (current_controller_NVCN-1).

The encoded_data payload is created according to REST message types and encoding using: message_type=control-down

The encrypted payload of the response consists of a JSON representation of the following data:

Key	Possible values
type	control-down
deviceId	device ID
status	neutral / down / error

```
{
    "type" : "control-down",
    "deviceId" : "device ID",
    "status" : "neutral" | "down" | "error"
}
```

The status vaules have the following meaning:

status	description

"neutral"	the power to the motor is not connected
"down"	the motor is turning towards downmost postition
"error"	some error occured

6.2.3 control-stop

Ressource:

```
http://device-ip/secure-control?payload=encoded_data
```

The control-stop message tells the secure-controller to stop.

The deviceID provided in the payload must match the configured device ID.

The NVCN provided in the payload must match (current_controller_NVCN-1).

The encoded_data payload is created according to REST message types and encoding using: message_type=control-stop

The encrypted payload of the response consists of a JSON representation of the following data:

Key	Possible values
type	control-stop
deviceId	device ID
status	neutral / stop / error

```
{
    "type" : "control-stop",
    "deviceId" : "device ID",
    "status" : "neutral" | "stop" | "error"
}
```

The status vaules have the following meaning:

status	description
"stop"	the power to the motor is not connected
"error"	some error occured

6.2.4 control-status

Ressource:

```
http://device-ip/secure-control?payload=encoded_data
```

The control-status message returns the current status of the secure-controller to the control device.

The deviceID provided in the payload must match the configured device ID.

The NVCN provided in the payload must match (current_controller_NVCN-1).

The encoded_data payload is created according to REST message types and encoding using: message_type=control-status

The encrypted payload of the response consists of a JSON representation of the following data:

Key	Possible values
type	control-status
deviceId	device ID
status	neutral / up / down / error

```
{
    "type" : "control-status",
    "deviceId" : "device ID",
    "status" : "neutral" | up | down | "error"
}
```

The status vaules have the following meaning:

status	description
"up"	the motor is turning towards upmost postition
"down"	the motor is turning towards downmost postition
"stop"	the power to the motor is not connected
"error"	some error occured

6.3 Security messages

6.3.1 security-fetch-nvcn

Ressource:

```
http://device-ip/secure-control/security-fetch-nvcn?payload=encoded_data
```

The security-fetch-nvcn message fetches the initialization vector from the device.

The deviceID provided in the payload must match the configured device ID.

The encoded_data payload is created according to REST message types and encoding using:
message_type =
encoded_data = deviceID

The unencrypted payload of the response consists of a JSON representation of the following data:

Key	Possible values
type	security-fetch-nvcn
nvcn	current secure-controller initialization vector

```
{
    "type" : "security-fetch-nvcn",
    "deviceId" : "device ID",
    "nvcn" : Stored initialization vector
}
```

6.3.2 security-pw-change

The security-pw-change message tells the device to change it's old password to the new one.

Additionally the deviceID provided in the payload must match the configured device ID.

decrypted payload = deviceID:security-pw-change:new password

Hint:
The old password does not has to be send because it is used by the device for the encryption of the message.

The encrypted payload of the response consists of a JSON representation of the following data:

Key	Possible values
type	security-pw-change
result	done / error

```
{
    "type" : "security-pw-change",
    "result" : done / error
}
```

6.3.3 security-wifi-config

The security-wifi-change message tells the device to set the Wifi client credentials it should use to access the target network.

Additionally the deviceID provided in the payload must match the configured device ID.

decrypted payload = deviceID:security-wifi-config:ssid:pre-shared-key

The encrypted payload of the response consists of a JSON representation of the following data:

Key	Possible values
type	security-wifi-config
result	successfull / failed / error

```
{
    "type" : "security-wifi-config",
    "result" : successfull / failed / error
}
```

6.3.4 security-reset-to-default

The security-reset-to-default message tells the device to reset all persistent changes to the factory default settings, e.g. the password.

decrypted payload = deviceID:security-reset-to-default

The encrypted payload of the response consists of a JSON representation of the following data:

Key	Possible values
type	security-reset-to-default
result	done / error

```
{
    "type" : "security-reset-to-default",
    "result" : done / error
}
```

6.3.5 security-restart

The security-restart message tells the device to apply a new configuration by restarting.

decrypted payload = deviceID:security-restart

The encrypted payload of the response consists of a JSON representation of the following data:

Key	Possible values
type	security-restart
result	done / error

```
{
    "type" : "security-restart",
    "result" : done / error
}
```

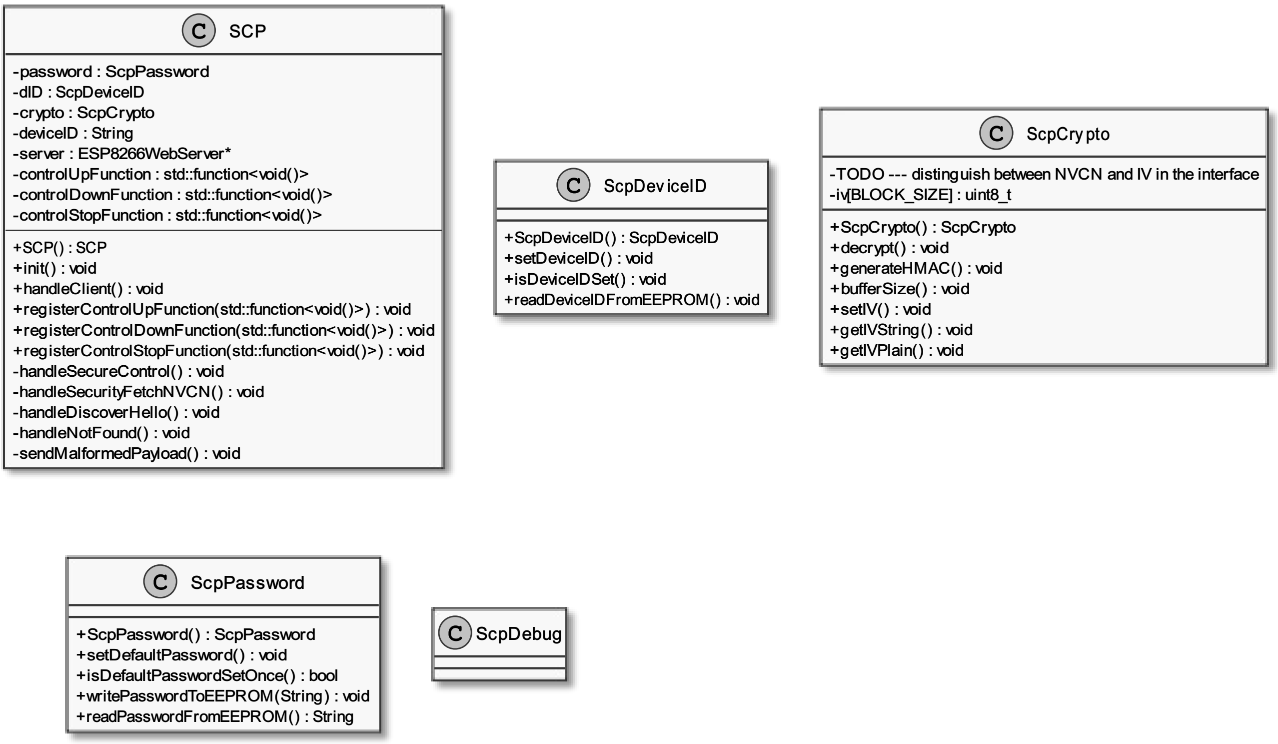
7. SCP Stack Software Architecture

7.1 Used Libraries

The SCP Arduino library uses the arduino-crypto and rBase64 libraries.

The can be found in the repositories described in the library.json file.

7.2 Class Diagrams



Class Diagram

8. Annex

8.1 Default credentials

8.1.1 Default device password

The default device password is 124567890123456.

8.1.2 Default Wifi Access Point credentials

SSID: "scp-controller-" + MAC Address

Pre-Shared-Key: default device password

8.2 ESP8266 EEPROM Layout

	0	1	2	3	4	5	6	7	8	9
0	res.	res.	res.	res.	res.	res.	DevID Set	Pw Set	PW	PW
16	PW	PW	PW	PW	PW	PW	PW	PW	DevID	De
32	DevID	DevID	DevID	DevID	DevID	DevID	DevID	DevID	Pw #	Pw
48	Pw #	Pw #	Pw #	Pw #	Pw #	Pw #	Pw #	Pw #	Pw #	Pw
64	Pw #	Pw #	Pw #	Pw #	Pw #	Pw #	Pw #	Pw #		

Project Philosophy

To enhance the security of the project and devices using the protocol the project is licensed under the GPL Version 3.0 or later to prevent third parties from using it while lowering the level of security without disclosure.

License

SPDX-License-Identifier: GPL-3.0-or-later

The full version of the license can be found in LICENSE.

If you need a license for commercial use, please contact Benjamin Schilling.

Copyright