

# Spis treści

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Dokumentacja struktur danych</b>          | <b>1</b> |
| 1.1      | Dokumentacja struktury seconn . . . . .      | 1        |
| 1.1.1    | Opis szczegółowy . . . . .                   | 1        |
| 1.1.2    | Dokumentacja pól . . . . .                   | 1        |
| 1.1.2.1  | onDataReceived . . . . .                     | 1        |
| 1.1.2.2  | onStateChange . . . . .                      | 1        |
| 1.1.2.3  | public_key . . . . .                         | 2        |
| 1.1.2.4  | state . . . . .                              | 2        |
| 1.1.2.5  | writeData . . . . .                          | 2        |
| <b>2</b> | <b>Dokumentacja plików</b>                   | <b>3</b> |
| 2.1      | Dokumentacja pliku seconn/seconn.h . . . . . | 3        |
| 2.1.1    | Dokumentacja typów wyliczanych . . . . .     | 3        |
| 2.1.1.1  | seconn_state . . . . .                       | 3        |
| 2.1.2    | Dokumentacja funkcji . . . . .               | 4        |
| 2.1.2.1  | seconn_get_public_key() . . . . .            | 4        |
| 2.1.2.2  | seconn_init() . . . . .                      | 4        |
| 2.1.2.3  | seconn_new_data() . . . . .                  | 5        |
| 2.1.2.4  | seconn_write_data() . . . . .                | 5        |
|          | <b>Indeks</b>                                | <b>7</b> |



# Rozdział 1

## Dokumentacja struktur danych

### 1.1 Dokumentacja struktury seconn

#### Pola danych

- **seconn\_state state**  
*Obecny stan połączenia.*
- **uint8\_t public\_key** [512]
- **int(\* writeData )(void \*src, size\_t bytes)**
- **void(\* onDataReceived )(void \*src, size\_t bytes)**
- **void(\* onStateChange )(seconn\_state prev\_state, seconn\_state cur\_state)**

#### 1.1.1 Opis szczegółowy

Główna struktura opisująca bezpieczne połączenie.

Nie powinna być wypełniana ręcznie, lecz za pośrednictwem funkcji `seconn_init`.

#### 1.1.2 Dokumentacja pól

##### 1.1.2.1 onDataReceived

```
void(* seconn::onDataReceived) (void *src, size_t bytes)
```

Wskaźnik na funkcję, do której przekazywane będą uwierzytelnione i zdeszyfrowane dane pochodzące od drugiego węzła.

Pierwszy argument zawiera wskaźnik na początek danych, drugi argument zawiera liczbę bajtów.

##### 1.1.2.2 onStateChange

```
void(* seconn::onStateChange) (seconn_state prev_state, seconn_state cur_state)
```

Wskaźnik na funkcję, do której przekazywane będą informacje o zmianie stanu połączenia.

Pierwszy argument zawiera poprzedni stan, drugi argument zawiera obecny stan.

### 1.1.2.3 public\_key

```
uint8_t seconn::public_key[512]
```

Klucz publiczny drugiego węzła. Wypełniany dopiero po zmianie stanu połączenia na AUTHENTICATED.

Uwaga! Nie mylić z wartością zwracaną przez funkcję `seconn_get_public_key`, która zwraca lokalny klucz publiczny.

### 1.1.2.4 state

```
seconn_state seconn::state
```

Obecny stan połączenia.

### 1.1.2.5 writeData

```
int(* seconn::writeData) (void *src, size_t bytes)
```

Wskaźnik na funkcję, która zostanie wywołana przez bibliotekę, gdy zajdzie potrzeba przesłania danych do drugiego węzła.

Pierwszy argument zawiera wskaźnik na początek danych do przesłania, a drugi liczbę bajtów które powinny zostać przesłane.

Funkcja powinna zwrócić liczbę bajtów, które rzeczywiście udało się przesłać.

## Rozdział 2

# Dokumentacja plików

### 2.1 Dokumentacja pliku seconn/seconn.h

#### Struktury danych

- struct **seconn**

#### Wyliczenia

- enum **seconn\_state** {  
    **NEW**, **HELLO\_REQUEST\_SENT**, **INVALID\_HANDSHAKE**, **SYNC\_ERROR**,  
    **AUTHENTICATED** }

#### Funkcje

- void **seconn\_init** (struct **seconn** \*conn, int(\*writeData)(void \*src, size\_t bytes), void(\*onDataReceived)(void \*src, size\_t bytes), void(\*onStateChange)(**seconn\_state** prev\_state, **seconn\_state** cur\_state), int(\*rng)(uint8\_t \*dest, unsigned size), int eeprom\_offset)
- void **seconn\_new\_data** (struct **seconn** \*conn, const void \*data, size\_t bytes)
- void **seconn\_write\_data** (struct **seconn** \*conn, const void \*source, size\_t bytes)
- void **seconn\_get\_public\_key** (struct **seconn** \*conn, uint8\_t \*public\_key)

#### 2.1.1 Dokumentacja typów wyliczanych

##### 2.1.1.1 seconn\_state

enum **seconn\_state**

Typ wyliczeniowy definiujący możliwe stany bezpiecznego połączenia.

Używany do określenia obecnego stanu połączenia w strukturze seconn oraz w wywołaniach funkcji onStateChange.

## Wartości wyliczeń

|                    |   |
|--------------------|---|
| NEW                | żadne dane nie zostały jeszcze wysłane ani odebrane.  |
| HELLO_REQUEST_SENT | wysłana została wiadomość HelloRequest  |
| INVALID_HANDSHAKE  | wystąpił problem w czasie nawiązywania połączenia, np. została odebrana nieprawidłowo uwierzytelniona wiadomość HelloResponse |
| SYNC_ERROR         | odebrane zostały dane, które są niezgodne z protokołem lub zostały nieprawidłowo uwierzytelnione                              |
| AUTHENTICATED      | połączenie zostało poprawnie nawiązane, uwierzytelniony został klucz publiczny drugiego węzła                                 |

## 2.1.2 Dokumentacja funkcji

### 2.1.2.1 seconn\_get\_public\_key()

```
void seconn_get_public_key (
    struct seconn * conn,
    uint8_t * public_key )
```

Funkcja odczytująca lokalny klucz publiczny.

Pierwszym argumentem jest zainicjalizowana funkcją `seconn_init` struktura typu `seconn`.

Drugim argumentem jest wskaźnik na miejsce w pamięci, do którego ma zostać wpisany klucz publiczny. Wymagane są 64 bajty pamięci.

### 2.1.2.2 seconn\_init()

```
void seconn_init (
    struct seconn * conn,
    int(*) (void *src, size_t bytes) writeData,
    void(*) (void *src, size_t bytes) onDataReceived,
    void(*) (seconn_state prev_state, seconn_state cur_state) onStateChange,
    int(*) (uint8_t *dest, unsigned size) rng,
    int eeprom_offset )
```

Funkcja inicjalizująca strukturę `seconn`. Należy ją wywołać jako pierwszą.

Pierwszy argument (`writeData`) zawiera wskaźnik na funkcję, która zostanie wywołana przez bibliotekę, gdy zajdzie potrzeba przesłania danych do drugiego węzła. Pierwszym argumentem tej funkcji (`src`) jest wskaźnik na początek danych do przesłania, a drugim (`bytes`) liczba bajtów które powinny zostać przesłane.

Drugim argumentem (`onDataReceived`) jest wskaźnik na funkcję, do której przekazywane będą uwierzytelnione i zdeszyfrowane dane pochodzące od drugiego węzła. Pierwszym argumentem tej funkcji (`src`) to wskaźnik na początek danych, drugi argument (`bytes`) zawiera liczbę bajtów.

Trzecim argumentem (`onStateChange`) jest Wskaźnik na funkcję, do której przekazywane będą informacje o zmianie stanu połączenia. Pierwszy argument (`prev_state`) zawiera poprzedni stan, drugi argument (`cur_state`) zawiera obecny stan.

Czwartym argumentem (`rng`) jest wskaźnik na funkcję, która będzie wywoływana przez bibliotekę w celu wygenerowania losowych danych. Pierwszym argumentem tej funkcji (`desc`) jest wskaźnik na miejsce w pamięci, do którego mają być wpisane losowe dane, drugim argumentem (`size`) jest liczba bajtów które powinny być wpisane. Funkcja powinna zwrócić wartość 1.

Ostatnim, piątym argumentem (`eeprom_offset`) jest miejsce w pamięci EEPROM do którego zapisywane i z którego odczytywane mają być lokalne klucze kryptograficzne. W tym miejscu powinno być 96 bajtów nieużywanej, ciągłej pamięci.

### 2.1.2.3 seconn\_new\_data()

```
void seconn_new_data (
    struct seconn * conn,
    const void * data,
    size_t bytes )
```

Funkcja którą należy wywołać w celu przekazania danych pochodzących od drugiego węzła do biblioteki.

Pierwszym argumentem jest zainicjalizowana funkcją seconn\_init struktura typu seconn.

Drugim argumentem jest wskaźnik na początek danych pochodzących od drugiego węzła.

Trzecim argumentem jest liczba bajtów danych.

### 2.1.2.4 seconn\_write\_data()

```
void seconn_write_data (
    struct seconn * conn,
    const void * source,
    size_t bytes )
```

Funkcja którą należy wywołać w celu przesłania do drugiego węzła danych. Dane te zostaną zaszyfrowane i uwierzytelnione, a następnie w postaci wiadomości EncryptedData przekazane do funkcji writeData ze struktury seconn.

Pierwszym argumentem jest zainicjalizowana funkcją seconn\_init struktura typu seconn.

Drugim argumentem jest wskaźnik na początek danych, które mają zostać przesłane.

Trzecim argumentem jest liczba bajtów danych.