SpringCard - PNA20248-AA

PUCK Smart Reader and SpringBlue Solution

Getting Started Guide

Introduction

SpringCard PUCK

SpringCard PUCK is a versatile NFC/RFID@13.56MHz USB-attached device.

In **Smart Reader** mode, **PUCK** is able to fetch virtually any kind of credential, token or user ID from a contactless card or an NFC smartphone or any other NFC object running in card emulation mode. The data is then transmitted to the host computer over the USB link, either using a **virtual communication port (serial) profile** or the **keyboard emulation profile** which is very convenient for legacy applications and/or to replace a barcode scanner.

More than that, the **PUCK Blue** version adds Bluetooth Low Energy (BLE) connectivity to read a credential, token or user ID provided by a BLE-enabled smartphone.

The SpringBlue solution

Developing, deploying and maintaining a complete NFC+BLE user authentication solution requires many skils and **a good expertise on secure transactions** to achieve a decent security level. Even for low-end use cases, where the cost of security could be seen as a useless expense, privacy concerns, the impact of a *bad buzz* and the risk of a system-wide fraud shall be taken in account and prevented.

SpringBlue is a both a reference design and a turn-key solution, which has been carefully designed by SpringCard's R&D team and security experts. The solution is easy to deploy in the field and rely on a very fast transaction scheme that rely on a minimal number of exchanges between the reader and the token, and standard AES128 cryptography to ensure both privacy and authenticity.

About this document

This document is a getting started guide for project managers, integrators or developers who want to evaluate or deploy a **complete NFC+BLE authentication solution** based on PUCK Blue devices together with the SpringBlue architecture and mobiles applications.

Understanding the SpringBlue transaction

The **SpringBlue** transaction is fully documented in **doc. PMD17128**.

Basically, the NFC or BLE token stores 5 informations:

- the `SiteID`, that uniquely identifies the organisation that 'owns' the credential,
- The `ObjectID` is a value that identifies the smartphone of the card,

- The `UserID` is the unique identifier of the holder (user) within the organisation,
- An AES key called `SOIK`, to protect the transmission of the `ObjectID`. This key is site-wide.
- An AES key called `OSUK`, to protect the transmission of the `UserID`. This key is specific to the smartphone or card, thanks to a diversification algorithm runned over the `ObjectID`.

Both keys are sensitive data and shall be protected into the smartphones (and into the PUCK readers, of course).

The transaction is divided into 3 exchanges only, which lead to a short execution time:

1. the PUCK sends SELECT APPLICATION, the object returns OK,
2. the PUCK sends a random challenge, the object returns its own random challenge,
3. the PUCK sends its `SiteID`, the object returns both its `ObjectID` and `UserID` in a secure crytogram.

The cryptogram is protected by 2 temporary, disposable AES keys that are computed over both random challenges and the `SOIK` and `OSUK` keys. This architecture prevents any kind of replay attack and expose little surface to brute-force attacks.

6 steps to instanciate SpringBlue for your own organisation

There a 6 steps to instanciate SpringBlue over a new organisation

1. Ask SpringCard to issue a new `SiteID` for you. This is importand to make sure different customers will never use the same `SiteID`. If you need only to evaluate and test the solution, please use `SiteID=00000001`.
2. Generate two keys using a strong random number generator. `SOIK` is your site-wide key to protect the transmission of the `ObjectID`. `MSUK` is the master key to compute a specific `OSUK` key for every object.
3. Create a database to store and manage your list of `UserID`.
4. Develop your application to run the SpringBlue transaction and deploy this application to the smartphones of all the users.
5. Push the configuration quartet `SOIK`, `ObjectID`, `OSUK`, `UserID` into all the smartphones (or other NFC/BLE objects).
6. Push the configuration triplet `SOIK`, `ObjectID`, `MSUK` into all the PUCK readers.

Configuring a PUCK to accept SpringBlue tokens

Howto configure a PUCK?

Principles

The PUCK features a non-volatile memory which is divided into register banks. Bank `02` (registers `0200` to `02FF`) stores the main part of the configuration. Bank `03` (registers `0200` to `02FF`) stores the so-called **Smart Reader Templates**, that tells the reader how to process the NFC (or BLE) cards or objects that comes into its nearby.

For a complete reference, please look at the detailed documentation of the **SpringCore family**, available online at docs.springcard.com/books/SpringCore.

There are 3 means of configuring a PUCK:

1. Using the `SpringCoreConfig` and `SpringCoreTool` command line utilities,
2. Using the **SpringCard Companion** service and cloud application,

3. Using a master cards

To ease the reading and to allow a developer to automate the configuration using his own scripts, we'll focus on the command line utilities, but the two other means would achieve the same result.

Note that to prevent unwanted configuration changes in-the-field (which would basically be a denial of service attack onto the readers), support of master cards shall be disabled by configuration, or a site-specific master card keyset has to be used.

The SpringCore command line tools are fully documented at docs.springcard.com/books/Tools/SpringCore/index.

Starting from a blank configuration

Use

```
SpringCoreTool load-defaults
```

and then

```
SpringCoreTool reset
```

to revert the PUCK to its out-of-factory configuration.

Namely, the out-of-factory configuration is

- PC/SC mode (register `02C0` set to `02`),
- All other configuration and template registers erased.

Writing a configuration step-by-step

To write a single register, use

```
SpringConfig --write <ADDR>=<VALUE>
```

Where `<ADDR>` is the address of the register, in hex, from `0200` to `03FF`, and `<VALUE>` its content in hex.

Writing the complete configuration at once

To write many registers at once, use

```
SpringConfig --file <FILENAME>
```

Where `<FILENAME>` is a configuration file. The configuration files follow the JSON syntax and contain the following structure:

```
{
    "type": "mastercardv2-content",
    "config": {
        "<ADDR>": "<VALUE>",
        "<ADDR>": "<VALUE>",
        ...
    },
    "templates": {
        "<ADDR>": "<VALUE>",
        "<ADDR>": "<VALUE>",
        ...
    }
}
```

Note that in this case the `<ADDR>` field is an hex value on one byte only, the first byte being set by the object's name (`02` for `"config"` and `03` for `"templates"`).

Using PUCK in keyboard emulation mode

Basics

First of all, we configure the operating mode as Smart Reader and the USB interface as HID (human interface device), keyboard emulation. Then we configure the BLE interface to read credentials (this part is ignored if the device doesn't have a BLE interface).

```
SpringCoreConfig --write 02C0=03
SpringCoreConfig --write 02C2=01
```

QWERTY keyboard layout

```
SpringCoreConfig --write 02A6=00
```

AZERTY keyboard layout

```
SpringCoreConfig --write 02A6=01
```

Using PUCK in virtual communication port mode

Activating SpringBlue over NFC (all PUCK versions)

Set `LKL=B0` in the Smart Reader template #1 to enable reading SpringBlue credential over NFC.

The `SiteID` and the two security keys `SOIK` and `MSUK` have to be configured as well.

```
SpringCoreConfig --write 0310=B0
SpringCoreConfig --write 0315=<SOIK><MSUK>
SpringCoreConfig --write 0316=<SiteID>
```

Activating SpringBlue over BLE (PUCK Blue only)

Set `LKL=B0` in the Smart Reader template #0 to enable reading SpringBlue credential over BLE.

The `SiteID` and the two security keys `SOIK` and `MSUK` have to be configured as well. Since all templates are independent, the keys must be explicitly specified in the BLE template, even if they are already known by the NFC template.

```
SpringCoreConfig --write 0300=00
SpringCoreConfig --write 0305=<SOIK><MSUK>
SpringCoreConfig --write 0306=<SiteID>
```

The SpringBlue mobile applications

In 2017, **SpringCard** has developed a reference mobile application using Cordova, a WebView-based framework that allows to share most of the code between Android and iOS applications.

Due to the lack of card emulation in iOS, the SpringBlue application for iOS implements only the BLE transaction. On the other hand, the SpringBlue for Android application implements both BLE and NFC, thanks to Android's HCE (host card emulation) feature.

Due to the frequent updates in both platforms and increasing constraints when it comes to NFC and BLE permissions, both applications are not usable anymore on today's systems.

Anyway, the code source is still available and could be used as a reference to implement new applications.