Weave Pairing Codes

2020/02/24
Revision 5

Overview

One of the requirements of Weave joining (as referred to as on-boarding) is that it be secure against attacks that happen during the joining process itself. In particular, the joining process must ensure that:

1. The device selected by the user ends up joining the user's fabric
2. The user's fabric admits only the device selected by the user, and no others
3. An eavesdropper on the joining process cannot gain information that would allow them to impersonate the new device or join their own device to the fabric at a later time.
4. Any disruption of the joining process by a malicious third party is obvious to the end user, and results in no breach of the security of the fabric.

Requirements 3 and 4 are achievable via standard encryption techniques. Requirements 1 and 2, however, are somewhat more difficult to achieve. Many systems use public key cryptography to strongly authenticate a party to a conversation. Weave devices follow this model in that each possess a unique certificate and private key that they can use to identify themselves. However, in a world where every device on the shelf at the local hardware store contains a valid certificate/private key pair, the problem remains as to how ensure that the device being joined to a fabric is indeed the device the user *intends* to join the fabric.

The solution to this problem in Weave is the use of pairing codes. A pairing code is a small random number that is both known to the firmware running on a device and printed on the outside of the device. Secure joining is achieved by requiring the user to read the pairing code printed on the device and enter it into a mobile application or web page which has the privilege to join the device to the fabric. As a convenience some devices may also display a QR-code that allows mobile applications to read the code automatically.

Weave pairing codes provide a means to confirm proof of physical possession at pairing time. This proof of possession is mutual. Specifically, using the pairing code, the user's application can confirm that the device being joined is indeed the one the user has in their hand. Similarly, the device can confirm that it is being paired by a user that is at least physically proximate to it.

Note that there's no requirement that a pairing code be unique across all devices; only that it must be unguessable by brute-force attack within a reasonable amount of time. Thus pairing codes can be relatively low-entropy (i.e. small).

Syntax of a Pairing Code

Pairing codes are strings of between 6 and 16 ASCII characters. The individual characters are drawn from an alphabet of 32 characters that has been chosen to improve readability. The initial characters of a pairing code are chosen randomly from the specified alphabet. The last character is a check character that can be used to detect input errors. (The check character also conforms to the pairing code alphabet).

The alphabet for pairing codes is as follows:

```
0 1 2 3 4 5 6 7 8 9

A B C D E F G H J K L M N P R S T U V W X Y
```

In an effort to make pairing codes easier to use, pairing codes on Nest devices are limited to 6 characters--five random characters plus the check character. E.g.:

```
AB713H
```

This gives 32^5 or 33,554,432 distinct codes.

Check Character

The check character provides a means to detect transcription errors introduced by the user during entry. The check character can be used by applications to perform an initial validation of a pairing code before it is used in the joining process.

The check character is generated using Verhoeff's Algorithm adapted to a base-32 character set (dihedral group D_{16}). The specific algorithm for pairing codes uses the following custom generated permutation in the digit permutation step:

(1 2)(7 11 13 5 20 23 9 6 27 15 21 25 14 10 8 31 26 4 16 22 12 29 18 24 28 17 3 30 19 0)

This version of the algorithm detects all cases of single digit errors (ABC → AVC) and adjacent transposition errors (ABC → ACB), and 97% of jump transposition errors (ABC → CBA).

An open implementation of Verhoeff's Algorithm for base-32 is available in Python and C++ in the GitHub OpenWeave project (https://github.com/openweave/openweave-core).

Input Normalization

To improve user experience, applications which allow users to manually enter a pairing code should normalize the input characters such that the system is tolerant of accidental substitution of visually similar, but illegal characters. For example, if a user enters the letter O, which is not in the legal character set, instead of rejecting the input the system should assume that they meant to enter the numeral 0 instead.

In general, the follow character substitutions should be made when normalizing a pairing code:

Input Character	Normalized Character
I (ASCII 73)	1 (ASCII 49)
O (ASCII 79)	0 (ASCII 48)
Q (ASCII 81)	0 (ASCII 48)
Z (ASCII 90)	2 (ASCII 50)

An alternative to input normalization is to disallow entry of illegal characters altogether, e.g. by blocking keys on a virtual keyboard.