

Embedded in Thin Slices

Bluetooth Mesh (Part 3)

Secure Provisioning

In this next part of his article series on Bluetooth mesh, Bob looks at how to create secure provisioning for a Bluetooth Mesh network without requiring user intervention. He also takes a special look at an attack called Man-in-the-Middle which Bluetooth's asymmetric key encryption is vulnerable to.

By
Bob Japenga

Both of our cars are more than 15 years old. My only new car envy is with the lack of a modern audio system. With a rental car, I'm always envious of the Bluetooth support and the seamless way I can connect and reconnect my phone to the car's system. Most of the new audio systems are well thought out and easy to use. For my birthday, I got a Bluetooth device that would connect my phone to my dumb audio system in both cars. I have been very happy with the devices although they have two quirks. One is that they don't work when the car has been left outside and it's below zero. After the car warms up, it will happily function. But it doesn't like subzero temperatures.

The other quirk—pointed out by my grandchildren—is that when it powers up, it announces: "Waiting for Pairing." And then when it is paired, it reports "Paired." The quirk is that instead of saying "Waiting for Pairing" it sounds like it is saying "Waiting for Perry." The first time my grandkids were in the car, they asked: "Who is Perry and why are we waiting for him?" Now I can only hear "Waiting for Perry" when I turn on the car.

Pairing is the way two standard Bluetooth devices establish the initial link for one-to-one networking (**Figure 1**). Bluetooth mesh needs a much more sophisticated and secure method of linking the many-to-many network. That method is called provisioning. I introduced Bluetooth mesh provisioning in my last article (*Circuit Cellar* 345, April 2019) [1]. So, if you haven't read that article, as a

minimum, it will be important to go back to understand the terms that were defined in that article and which I will be using in this article.

As I mentioned last time, the Bluetooth specification [2] states that only if an Out-of-Band (OOB) public key is used or if an OOB action is taken to pass the public key (using user supplied information), "provisioning is Insecure Provisioning." This statement will basically jettison any project that does not use one of these two OOB methods when presented to a savvy IT group. It did for us. Imagine presenting to your CEO a new product line using Bluetooth mesh that doesn't use one of these two methods. Most likely the savvy CEO will ask: "What is the projected return on our investment?" AND "Is it secure?" Would you want to say: "Well, we are using Insecure Provisioning but other than that it is secure?"

I'm not convinced that the specification is entirely accurate in this statement and would appeal to the Bluetooth SIG to reconsider their wording. I want to elaborate on this idea in this article and provide some means for making provisioning secure without using either of the two OOB methods to pass the public keys.

MAN-IN-THE-MIDDLE

As I mentioned last time, Bluetooth uses asymmetric key encryption during the first part of provisioning. Asymmetric key encryption has one basic security flaw. It is subject to what is called a Man-in-the-Middle (MitM) attack. Let me illustrate this attack.

Imagine that Bob and Barbara are happily married. I know, normally everyone uses Alice in these illustrations, but my wife's name is Barbara. They want to communicate some secret birthday plans about their grandson Sean. So, they both send over clear text their public keys (B1 and B2) (**Figure 2**). Bob encrypts all of his messages with Barbara's public key B2, and sends them to Barbara. Barbara decrypts all of Bob's messages using her private key B2P. Barbara sends all of her messages to Bob using Bob's public key B1 to encrypt the data. Bob decrypts Barbara's messages with Bob's private key B1P.

Imagine that grandson Sean is a curious computer whiz and wants to know what's he is going to get for his birthday. He intercepts the public key exchange B1 and B2 between his grandparents. Instead of passing on their public keys, he sends them his public key S1. So, when Bob and Barbara send their messages encrypted with S1 to each other he intercepts them and decrypts them using his private key S1P since they are encrypting their messages with his public key S1. He finds out what he is getting for his birthday and then encrypts the messages using Bob and Barbara's public keys and sends them back to them. Bob and Barbara are clueless to the fact that Sean now knows what he is getting for his birthday.

That example illustrates that, if during the provisioning process, the public keys are not exchanged OOB, the process would be insecure because they would be subject to a MitM attack. However, during normal asymmetric key encryption, the way this can be prevented is through authentication. If Bob can know that a key is authentically from Barbara, he would immediately recognize that the key that Sean sent was not from Barbara. During normal Internet asymmetric key encryption this authentication is done

through Certificates of Authority created by a trusted signing authority.

The Bluetooth provisioning process includes authentication of the device as part of the process. Authentication can either be using an OOB technique or without OOB. So, I would contend that if you use some means of authenticating that does not transfer the credentials over the Bluetooth network, your provisioning process would be secure in spite of what the Bluetooth specification says (I am definitely treading on thin ice here!).

POSSIBLE OOB AUTHENTICATION

Obviously, the more user intrusive OOB methods (input, output, NFC) would work (**Figure 3**). But what kinds of Static OOB techniques could you use so that you would not require a user interface or user interaction during provisioning. Let me propose a few, but there are many other ways.

Unique Device Identifier: As part of the beaconing process (described in my April article), the unique device identifier (UUID) is passed to the provisioner. This identifier is much like a MAC address and is unique to the Bluetooth radio. It is 16 octets long. Each device (including the provisioner) could have a secret algorithm to hash this UUID. This hashed UUID can be used as a private key to encrypt and decrypt the public key thus providing authentication. This would provide Static OOB authentication. In all likelihood, the provisioner will be connected to the cloud and have a list of the devices that are going to be installed. This was the case with one system on which we worked. This was necessary for the provisioner to create a map of the installation—to show where each device was located throughout the building. This was used for maintenance as well as normal operation.

Secret Number: The device could have its own unique authentication value known to



FIGURE 1

Pairing is the way two standard Bluetooth devices establish the initial link for one-to-one networking.

the provisioner. This could be used like the UUID in the above example. This secret could also be the same in all devices but I would recommend a unique secret per device.

These are just a sample of the ways this could be done. The key point is that non-OOB passing of the keys plus static OOB authentication would be secure in spite of what the Bluetooth specification says. No matter how you pass the initial keys—with or without OOB—there are three things that are also in your favor to prevent an MitM attack: the extreme difficulty of mounting an MitM attack, the actual damage that the MitM causes and your ability to detect it when it happens. Let's look at each of these.

MANAGED FLOODING

Part of providing realistic security measures is the understanding that MitM attacks are not easily achieved. One of the challenges of mounting a MitM attack is in that simple word "intercept" in my

description. Imagine someone mounting an MitM attack between your browser and your bank in order to get your credentials. Their first line of attack might be your Wi-Fi network in your home. The attacker pulls up a van next to your house and starts listening to your Wi-Fi traffic. Assuming it is unencrypted (heaven forbid) or that the attacker obtains the pass key, the attacker can read all of the data passed between you and your router. The attacker still has the problem of trying to intercept the communication between your browser and the bank. This is non-trivial.

But imagine you were trying to perform this non-locally. How does one intercept your message in the mesh network called the World Wide Web? Certainly, if you replaced the name server you could redirect all requests between you and the bank to you. But without insider help that is non-trivial. Hijacking one of the World Wide Web routers would be another to intercept your traffic. But this would require serious resources.

Let's switch back to the Bluetooth mesh network, which is not a routed mesh network like the World-Wide-Web but uses managed flooding. Managed flooding was described in Part 1 of this article series (*Circuit Cellar* 343, February 2019). It has no name servers. There are no routing tables to manipulate. This means that all messages go out to all devices that relay messages not for them, and answer messages that are for them. The route can be different on every transmission. Imagine mounting a MitM attack in this environment. One of the outer devices issues its beacon to start the provisioning. You are the attacker sitting on a shelf in a big box store. You have to be positioned such that you get the message before the Provisioner does. And then you have to be able to intercept the subsequent messages before the Provisioner. Although that is possible, it is non-trivial. I would love to see some researcher actually document how a real-world MitM attack can be made on a managed flooding Bluetooth mesh network. It seems like it would not work deterministically.

ASSETS LOST WITH MitM

Putting aside the technical challenges of mounting an MitM attack, let's assume that they are actually successful in mounting the attack. What do they have? The private network key and one device key and one application key (we will discuss these next time). If you design your network correctly, they can only decode one device's data. You certainly can perform nefarious functions like sending bad data for one device. If you evaluated the assets that you want to protect (as described in my August 2018 article, *Circuit Cellar* 337 [3]) you may find that the risk of harm is very slight

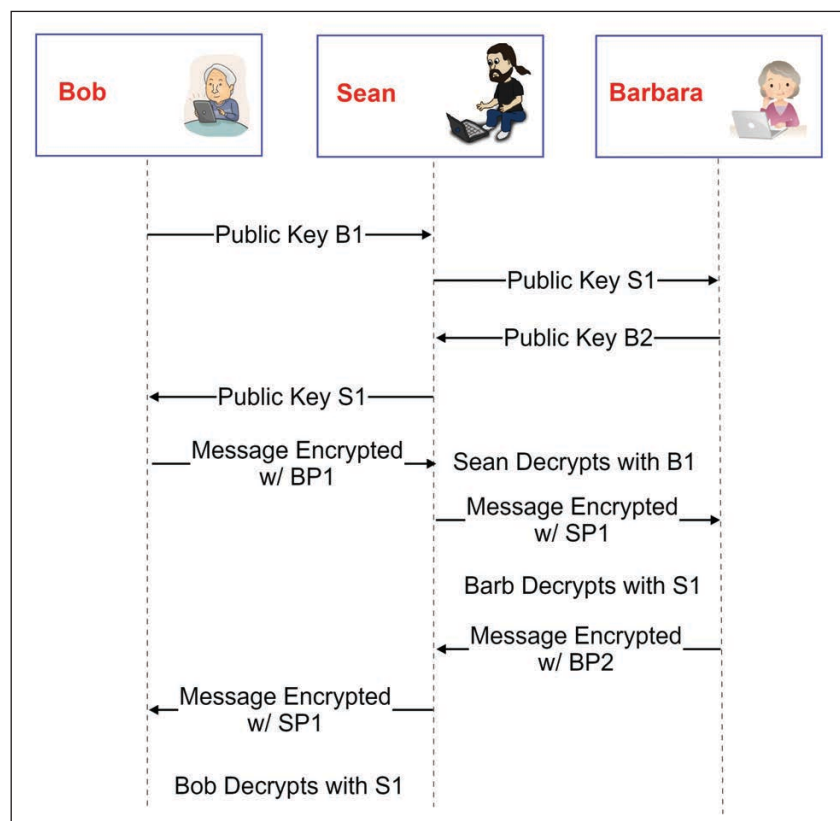


FIGURE 2

Shown here is an example exchange that would be insecure because it would be subject to a Man-in-the-Middle attack. However, during normal asymmetric key encryption, the attack can be prevented through authentication.

For detailed article references and additional resources go to:
www.circuitcellar.com/article-materials

References [1] through [3] as marked in the article can be found there.

from a successful MitM attack. Obviously, this is dependent upon your application. Between you and the bank, much harm can be caused. But in the applications that we have evaluated, the risk of harm from a successful MitM attack is very slight. And the major risks are that the attacker has the ability to disrupt the network. For example, between one lighting fixture and the host, a successful MitM attack could cause no more harm than a simpler brute force attack which would be simple to perform. If some hacker wanted to disrupt your network, it would be much easier and more deterministic to just create a bunch of devices that flood the Bluetooth radio cloud with so much noise that the network becomes non-functional. Hide three of those on shelves in the big box store and you simply shut the network down.


TAMPER DETECTION

There are two ways to guard against a MitM attack. One is authorization and the other is tamper detection. Every system will be different, but I am convinced that you can design most Bluetooth mesh systems to detect MitM attacks. Remember our earlier article about IoT security where we said: "Don't put all your eggs in one security basket." Design your secure network like you would protect your castle. You have the moat, the wall, the fortified door, the boiling oil and your warriors. Even if you design your system using OOB key transmission and OOB authentication, don't believe the Bluetooth SIG that you are secure. You should still provide checks for tampering. Let me illustrate a few methods for detecting a MitM attack.

Imagine that you are replacing an existing unit in the big box store. The new device needs provisioning. Imagine further that an attacker mounts a successful MitM attack in spite of the fact that you are doing everything in a secure fashion. There is a chance that there are some markers that might indicate a security breach. Most likely the MitM device will alter the route time and number of hops. Another way is that with managed flooding the device may occasionally take a different route by-passing the MitM and the edge gateway would get a message from the device encrypted with the the device's private key instead of the MitM's private key. Your provisioner/gateway should be designed to report errors that might be caused by a MitM attack.

CONCLUSION

The Bluetooth mesh specification says that only if you exchange your public keys over OOB can provisioning be secure. I contend that, with OOB authentication similar to what I outlined here, you can have secure provisioning. I invite anyone to help me see otherwise. This stuff is complex and it is easy to miss things. In

addition, I contend that MitM attacks are probably not where a hacker will attack your network. In addition, tamper detection needs to be provided in your system to help you plan for the unexpected. Of course, this is easy for me, because I am writing this in thin slices. When you design these networks, you have to have the whole pie. 

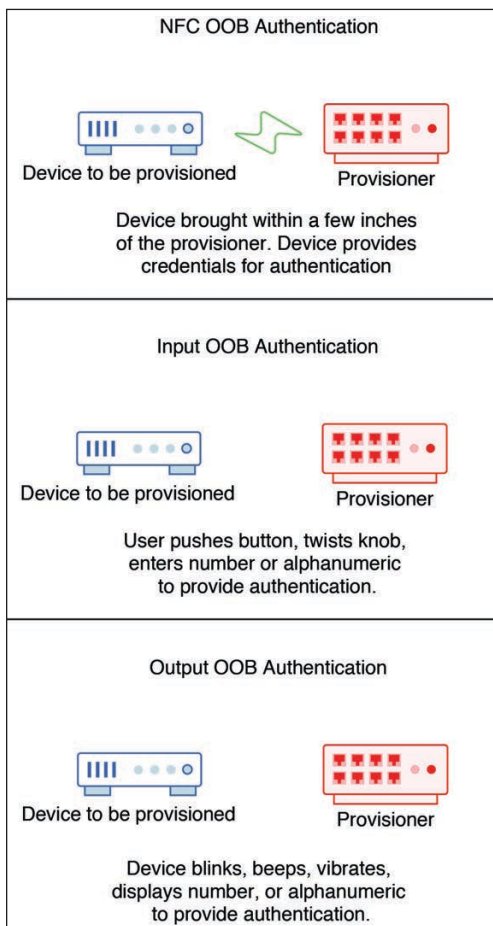


FIGURE 3

Shown here are three types of Out-of-Band (OOB) authentication: NFC, input and output. But these are user intrusive. In contrast, Static OOB techniques have the advantage that they don't require a user interface during provisioning.

ABOUT THE AUTHOR

Bob Japenga has been designing embedded systems since 1973. In 1988, along with his best friend, he started MicroTools, which specializes in creating a variety of real-time embedded systems. MicroTools has a combined embedded systems experience base of more than 200 years. They love to tackle impossible problems together. Bob has been awarded 11 patents in many areas of embedded systems and motion control. You can reach him at rjapenga@microtoolsinc.com.

