

Embedded in Thin Slices

Bluetooth Mesh (Part 2)

Provisioning Pondered

Continuing his article series on Bluetooth mesh, this month Bob looks at the provisioning process required to get a device onto a Bluetooth mesh network. Then he examines two application examples and evaluates the various options for each example.

By
Bob Japenga

COLUMNS

A number of years ago we were working with a company on an OS/2 system. Does anyone remember that operating system by IBM? For a number of years, our company standardized on OS/2 for all of our desktops (**Figure 1**). Eventually Microsoft crushed the competition for the desktop market and made it impossible for us to continue using OS/2. We were working with one of our client's designers on a process control system using OS/2 [1]. After several weeks of interaction with this designer, I was still not sure if his basic design and the architecture of his software was pure spaghetti code or if it was a magnificent and complex architecture that I just couldn't understand. After several months, I think I finally settled on the fact that it was a bit of both. But the elegance of the design escaped me for quite a while.

This reminds me of reading the Bluetooth mesh specification. I am not sure if I am looking at a mishmash of ideas thrown together or an elegant masterpiece that I am missing because of the complexity. Given the team that was involved in creating the specification, I lean towards the elegant masterpiece. But let it be known that, for

me, it's a real challenge to figure out how things are supposed to work from the documentation provided.

I will make an attempt over the next several articles to help all us better understand how Bluetooth mesh works so we can determine if it is applicable to the project on which we are working. I hope to distill down the basic concepts and provide guidelines as to how we can apply this new technology. Hopefully I don't muddy the waters and make things less clear with my own mishmash of ideas.

DEFINITION OF TERMS

Provisioner: A Provisioner on a Bluetooth mesh network is the device that manages the provisioning process. Although multiple Provisioners are allowed on the network, the method of transferring the provisioning data is implementation specific. This means that there will probably be work to switch suppliers of the Bluetooth radio in the Provisioner if you use multiple Provisioners.

Provisioning: Provisioning is the steps performed to add an unprovisioned device to a Bluetooth mesh network. The provisioning process accomplishes two goals: Authenticate the device (confirm that it is a device that is supposed to be on the network) and create the



FIGURE 1

We once worked with one of our client's designers on a process control system using IBM's OS/2 operating system. The design had an elegance that reminds me of the Bluetooth mesh specification.

secure link (keys, algorithms, randomization information) to enable the device to join the mesh network.

Provisioning Data: This is the data that is exchanged during provisioning between the Provisioner and the device that is being added to the network. Among the things it includes are the network key, the address of the device and a number (IV Index) that is used to create a unique identifier for each message.

Symmetric and Asymmetric Key Encryption: These are the two basic types of encryption. Symmetric key encryption uses the same key for both encrypting and decrypting a message. Asymmetric key encryption (also called Public key encryption) uses a public and private key pair that each party generates separately (**Figure 2**) [1]. The public key can be sent to anyone unencrypted (also called “in clear text”). They can then encrypt messages with this public key and only the holder of the private key can decrypt the message. (**Figure 3**). The Bluetooth SIG chose to use the computationally less expensive symmetric encryption during normal networking where both the sender and the receiver have the same key. But during provisioning, which generally happens only once, asymmetric encryption is used. Check out the Bluetooth SIG’s glossary of terms for additional definitions of terms—a link is provided on the *Circuit Cellar* article materials webpage [2].

STATEMENT OF THE PROBLEM

Asymmetric key encryption is the primary method used on our PCs when we connect to a secure site on the Internet. The beauty of this method is that public keys can be transmitted in clear text (unencrypted) to start a secure session. Once I have your public key, I can encrypt my messages and send them to you securely. You use your private key (which is uniquely paired with your public key) to decrypt my message. You do the same thing in reverse with any data that you want to send to me. This beautiful scheme has worked wonderfully since it was first proposed in the 1970s.

The problem for Bluetooth mesh is that, since it chose to use symmetric key encryption where both parties have the same private key, how do all parties securely get that private key? The Bluetooth mesh team created a method of distributing a unique private key that is executed before any device joins the network. This is what we are going to describe here.

At the highest level, an unprovisioned device periodically sends a beacon in clear text with its unique identifier and its capabilities as a device. The Provisioner then opens a link for that device and the device

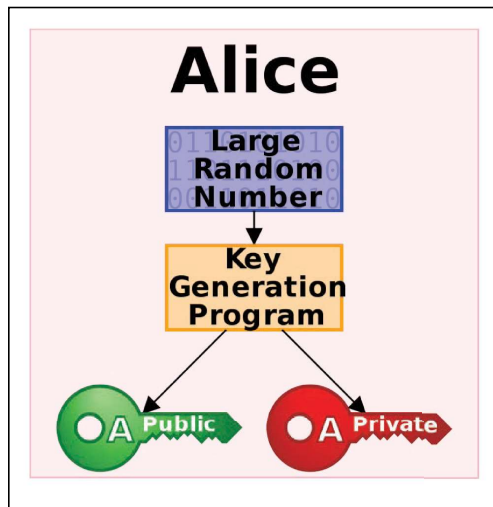


FIGURE 2

Asymmetric key encryption (also called Public key encryption) uses a public and private key pair, which each party generates separately.

acknowledges the Provisioner. Provisioning takes place when the Provisioner sends the provisioning data to the device. This data includes the private key, which is called the Network key. There is more than one type of key but we will only cover the primary private key in this article. Provisioning also authenticates that the unprovisioned device is that it says it is. Authentication, although part of provisioning, is another significant topic that is too large to cover in this article. We will look at authentication in a future article in this series. Once the provisioning data is successfully exchanged, the link is closed and the device is now part of the Bluetooth mesh network.

DRILLING DEEPER

The provisioning process consists of five phases. Phase 1 is called Beaconsing,

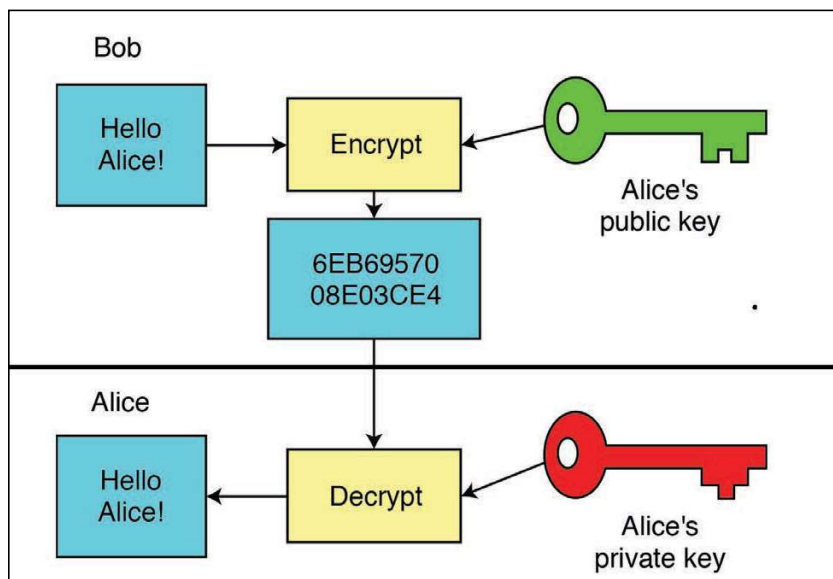
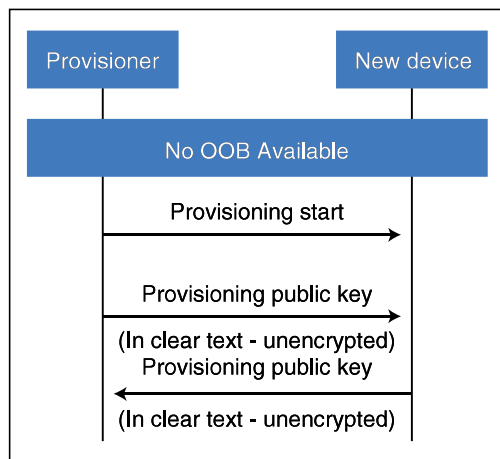


FIGURE 3

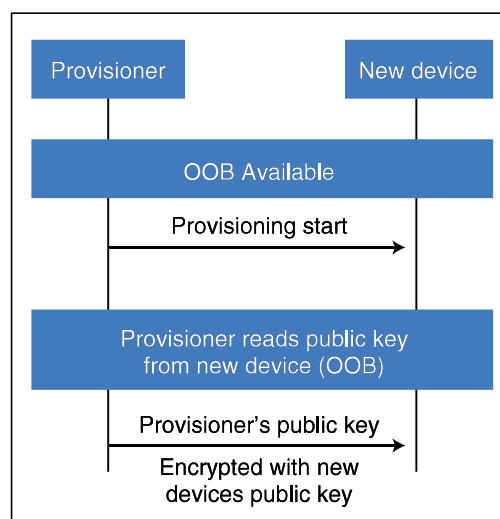
The public key can be sent to anyone unencrypted (also called “in clear text”). They can then encrypt messages with this public key and only the holder of the private key can decrypt the message.

FIGURE 4

Insecure provisioning

**FIGURE 5**

Secure provisioning



whereby the device lets the Provisioner know it is desiring to be provisioned. Phase 2 is called Invitation whereby the Provisioner invites the device to join the network. The device sends its capabilities including what encryption algorithms it supports, the type of public keys it supports and the type of Out-of-Band (OOB) capabilities it supports. Phase 3 is where the Provisioner and the device exchange their public keys. Phase 4 is where the Provisioner authenticates the device. Finally, in Phase 5 the Provisioner distributes the provisioning data.

In the case of a browser and a secure Internet site, the secure Internet site wants anyone to be able to connect securely to the site. However, not everyone has access to the site. Typically access to the site is limited by usernames and passwords.

This is not true for most mesh networks. A mesh network doesn't want to allow every

device in range to exchange the public keys in order to have a secure network connection. This is especially true with a mesh network where a lot of the work of the device is passing data from other devices to devices downstream. You wouldn't want a rogue device to be able to do this. So, for Bluetooth mesh, it would not be wise to pass the public keys in clear text (unencrypted).

With that in mind, the simple step (for PCs) of exchanging public keys is not so simple for Bluetooth mesh networks. For Bluetooth mesh networks, the passing of the public keys must be done securely to have a secure network. The Bluetooth specification allows two ways for the public keys to be exchanged. You can securely exchange the public keys through an OOB method or insecurely (**Figure 4** and **Figure 5**).

There are a number of OOB methods allowed by the specification. All of them involve user interaction. One is that the public keys are exchanged over a Near Field Communications (NFC) link. Since a lot of the Bluetooth chips include NFC capability, technically this is a viable solution with no extra recurring cost to the device. Other methods include some form of user input. Again, the specification allows a variety of input and output methods: audible, alphanumeric, visual, vibration or discrete button pushes. So, if your application can allow that, you can provide the public keys via some fairly simple I/O mechanizations—albeit requiring complicated user interaction.

REAL WORLD EXAMPLES

The Bluetooth mesh specification's statement that only the OOB method of exchanging public keys is secure is disconcerting for anyone concerned about security and simplicity of installation. We know that all of these OOB methods add significant complications for the user during provisioning. Next time we will drill into the details of the "insecure" method of passing the public keys that are used. I say that because, as I will attempt to demonstrate next time, the method is not as insecure as it might seem. For now, let's assume that you are designing a system for a customer using Bluetooth mesh. You have a customer who wants the network to be secure. The specification says that only the OOB method of passing the public keys is secure. Let's look at some real-world implementations and what options you have to create secure provisioning.

Home Automation: Imagine that you are designing a line of devices for home automation. You want the network to be secure. You want the installation to be simple. The simplest way would be to install the devices and turn them on. All devices would

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.

automatically be provisioned securely. How can this be achieved? One option to achieve this would be to completely bundle everything pre-provisioned at the factory. This removes the complexity of user involvement in the passing of the public keys, keeping the installation simple for the homeowner. The homeowner could just install the devices, turn them on and they would work. The problem is that you would be selling a system, not home automation devices. This may be acceptable in some designs but not in others.

Alternatively, the factory could pre-install all devices with the same shared public keys. No public key exchange would take place during provisioning. This would enable you to sell devices rather than systems and be able to replace faulty devices. This is similar to the default trust center link key approach of Zigbee and has some of the same pitfalls.

The NFC method is more flexible and quite simple for the user. Each device would need to be brought in close proximity (within inches) to the Provisioner as part of the installation procedure. If the device has a user interface, alphanumeric input is also a possibility although sometimes a cumbersome, error prone and time consuming process. The blinking LED, the vibrating box, the push button or the audible beep are even more complicated for the user but are open to you as designers.

Industrial Control: Imagine that you were designing some kind of lighting control for large buildings. There might be hundreds of your devices installed throughout the building. Your customers want it to be secure and they are going to want it to be simple to install and maintain. How would OOB public key exchange work in this example? Pre-provisioning at the factory is a possibility. But, as with home automation, it means that you are not selling devices but a system with a predetermined number of devices.

Requiring the installer to bring each device to the Provisioner or to use the user input or output on each device is the other option. One of our customers deemed it an unacceptable burden on the installer to use NFC OOB and require the installer to take the fixture and bring it into close proximity with the Provisioner. Even though the device had an LED and a push button on it making visual or discrete input OOB a possibility, the user input method would have been even more complicated. Requiring the installer to push the button X number of times or reading the pulse pattern on the LED was completely rejected by our customer.


We had to take a hard look at how insecure the Bluetooth provisioning process was without OOB in our environment. The method may not be as insecure as it seems. More on that next time.

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.



CONCLUSION

One only needs to read the number of vulnerabilities that are published daily on the CERT Vulnerability website [3] to know that security is very hard to achieve. The provisioning process is key (pun intended) to creating a secure Bluetooth mesh network. Next time we'll describe the "insecure" method of passing the public keys and evaluate when this method is an acceptable risk. But of course, only in thin slices. 

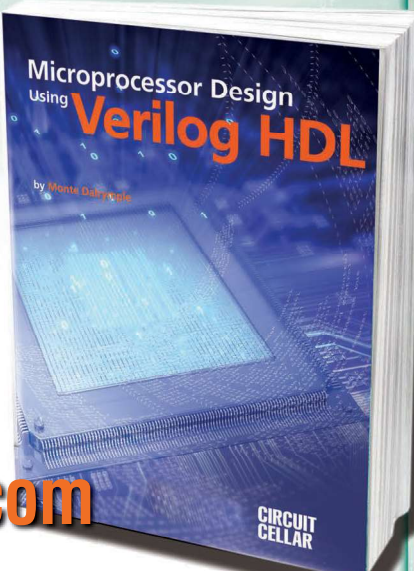
Verilog HDL

With the right tools

designing a microprocessor can be easy.

Okay, maybe not easy, but certainly less complicated. Monte Dalrymple has taken his years of experience designing embedded architecture and microprocessors and compiled his knowledge into one comprehensive guide to processor design in the real world.

Monte demonstrates how Verilog hardware description language (HDL) enables you to depict, simulate, and synthesize an electronic design so you can reduce your workload and increase productivity.



cc-webshop.com