Frequency Hopping Spread Spectrum (Wireless Communications)

Blaz Repas

February 7, 2017

Seminar paper, Course: Cryptography and Computer Security

GOALS: Present frequency hopping, it's potential for securing wireless communications and show a possible scheme for frequency hopping

Abstract

IoT devices are more and more popular, even more so when they utilize wireless communications. However these wireless data links are more often then not unsecured. That leaves the devices prone to various attacks. One of the DoS attacks is jamming the operating frequency and thus disabling the communication. Measures should be taken to prevent such attacks and at the same time make the communication channel less feasible for eavesdropping.

1 Intro

Wireless communication can be done in many ways. In this paper we will focus on point-to-point communication between two devices. At any given time, one of the devices will be the transmitter (Alice) and the other one the receiver (Bob). Normally this two devices operate at a fixed, predetermined frequency. Since the frequency is fixed, both devices know where to send and where to receive. However in this case, a careful eavesdropper (Eve) can determine the frequency, by scanning the spectrum. When in possession of this knowledge, Eve can tune to the frequency and receive the data, just like the valid receiver - Bob. If the data is not encrypted, then Eve can also determine the contents.

Even if the data is in fact encrypted, there are still malicious attacks that Eve can do. She can overpower the legitimate transmitter (Alice), and

thus makes receiving valid data impossible - she can jam the frequency. This DoS attack can be prevented if the frequency is unknown to Alice. She can not disrupt the communication, because she does not know the operating frequency. However Eve can still find the transmission by means of scanning if the operating frequency is fixed.

2 Frequency hopping

Now let us consider changing the operating frequency f while transmission is in effect. If Alice can change f very quickly - in other words hops to a different frequency, Eve does not have enough time to scan the spectrum and determine the new frequency. This mostly solves the problem of jamming, but creates a new problem. Like Eve, Bob also does not know on which frequency to listen in order to receive the transmission.

Somehow Alice and Bob must know in advance on which frequencies they will operate at any given time, or be able to use a deterministic calculation to calculate the next frequency to use. In other words, both need to be able to produce the same sequence of frequencies in a deterministic fashion. That is usually done by splitting the usable spectrum space S into channels. Each channel is given a unique number and a corresponding (center) frequency. For example we can have, $n_{ch} = 256$ channels

$$c_i \in \{0, 1, 2, ..., 255\}$$

and usable spectrum space of

$$S = 4MHz$$

.

Then each channel is $w = \frac{4}{256}MHz = 15.6KHz$ wide and operating frequency is calculated by the formula

$$f = f_0 + i * w$$

Then, usually, a pseudo-random number generator (PRNG) is used to determine the channel number

$$c_i \in prng() \ mod \ n_{ch}$$

every ten to a few hundred milliseconds. For Alice and Bob to be able to communicate, they have to share the seed for the PRNG and keep generating next channel number in a synchronized way.

When we keep the next operating frequency hard to guess, we obtain a jamming resistant scheme. An interesting side-effect of jamming and interference resistance is the fact that multiple transmitters can coexist and transmit at the same time. This is relied upon in radio control applications, like for example, flying an RC airplane or quadcopter. Many commercial RC transmitters employ a similar frequency hopping spread spectrum scheme to ensure uninterrupted control. This allows multiple RC operators to fly their contraptions without interfering and loosing control.

2.1 Weaknesses

The goal to combat jamming attacks is achieved by rapidly changing the frequency. However, if Eve can somehow predict the next transmission frequency, she can also synchronizes her jammer to hop to the new f at the same time as Alice and Bob. In the basic scheme we are using a PRNG to generate the sequence of channel numbers and thus the predictability of f is directly tied to the predictability of the PRNG used.

It is also worth noting that the period of the PRNG is very important in this application. New element of a channel number sequence is generated relatively frequently and eventually the sequence repeats. An attack exploiting the relatively frequently repeating channel number sequence will be presented in the section *Attacks with software-defined radio*.

2.2 Hardware requirements

The scheme described here quite general, however there are certain requirements for the implementation to be feasible. Firstly we need some way to to execute the PRNG calculation and data transmission handling. That is usually achieved by utilizing a general purpose computer, or in the world of IoT devices, a microcontroller. Secondly, to be able to transmit/receive, we need a wireless transceiver that operates in the frequency band of interest (for example 868MHz or 2.4GHz). Obviously, we require that the transceiver can tune to different frequencies within the band. Depending on the hop interval, we might require that the transceiver is able to tune a frequency very fast. Needles to say, tuning needs to be controllable by the microcontroller.

Depending on the application and desired data rate, we make requirements also on the bandwidth of the usable spectrum. This also affects the number of channels available for hopping.

3 Pseudo-random sequences

This section will be rather brief since there is a lot of articles and information available on pseudo-random number generators. I will try to explain what PRNGs are commonly used in frequency hopping applications.

The PRNGs commonly used in frequency hopping applications are relatively simple and more importantly not CPU intensive as the embedded hardware is not capable of complex computations in real time. Most often used are linear feedback shift registers (LFSR) with a LFSR polynomial.

Similarly, linear congruential generators (LCG) PRNGs are also used due to their simplicity. In employing LCG, one needs to choose a good polynomial to keep the period as long as possible.

On embedded hardware that supports fast AES executed (possibly hardware accelerated) AES256 with appropriate mode can be used to generate random numbers [4].

4 Synchronization

As mentioned before, Alice and Bob need to share a starting seed s_0 for PRNG they both use. The scheme would not be secure if s_0 was fixed and never changed. Therefore we need to change s_0 and distribute it securely to Alice and Bob before they establish a transmission. That implies that we need to implement a protocol for key exchange to generate and exchange the s_0 between Alice and Bob as securely as possible, with consideration for hardware limitations that we are facing in a certain setup.

I will assume that both Alice and Bob have a pre-shared key k_{psk} that is securely stored (I will not cover physical security in this paper) and I will assume that physical access to the devices in question is not within reach of the attacker.

In the simplest form, synchronization is done on a predefined frequency f_{sync}. Alice generates a random number, either by using some form of PRNG or by using a hardware backed random number generator exploiting some physical phenomena. In either case Alice generates a random number x_{sync} and encrypts it, using for example AES and the locally stored k_{psk} as key to obtain

$$y_{sync} = enc(x_{sync}, k_{psk})$$

for example:

$$y_{sync} = AES256CBC(x_{sync}, k_{psk})$$

Encryption algorithm can also be something else, as long as it provides reasonable security. When dealing with IoT devices, asymmetric ciphers

might be too computationally complex to satisfy the real-time nature of wireless communications. That is why I would suggest using AES (more specifically AES256-CBC), as some microcontrollers include hardware acceleration for it.

Encrypted message y_{sync} is then sent from Alice to Bob on frequency f_{sync}. Bob receives y_{sync} and decrypts it

$$x'_{sync} = dec(y_{sync}, k_{psk})$$

and uses x'_{sync} as s_0, the seed for frequency hopping PRNG. Alice does the same thing after receiving confirmation message from Bob. This scheme would suffice to establish a frequency hopping transmission between Alice and Bob.

There is, however, an issue. The synchronization frequency f_{sync} is fixed and can facilitate another attack vector. Eve can jam f_{sync} and prevents synchronization to succeed, effectively preventing Alice and Bob from communicating.

One important aspect of synchronization is resynchronization. This should be done periodically to prevent PRNG to reach its period and to prevent Alice's and Bob's timing to drift too much and inhibit the transmission.

4.1 Preventing DoS during synchronization

We need to extend the scheme for synchronization to prevent or at least make DoS attacks less feasible. There already is a way to prevent jamming once Alice and Bob are synchronized - we use wide range of frequencies instead of just one that is fixed. That idea can be modified to fit the synchronization scheme as well. Bob would randomly pick a channel number and corresponding frequency f_{sync} from a subset of channels

$$chns_{sync} \subset chns_{trans}$$

that are normally used for transmission. Alice would then also pick a random channel from the same subset and try to synchronize with Bob. If Alice picked the wrong channel, she would simply repeat the process and randomly select another. Each time Alice would succeed with the probability of

$$P(sync) = \frac{1}{|chns_{sync}|}$$

On average this would take $|chns_{sync}|$ iterations. This gives as a clue: number of synchronization channels should be kept relatively low (to allow for a

fast synchronization), but still high enough to make guessing it hard (at least in the time needed for synchronization) to avoid being jammed during synchronization process.

There is still one detail to address: how do we pick/obtain elements of $chns_{sync}$. This can be a function of either the shared secret k_{psk} or we can have a dedicated PRNG for generating elements of $chns_{sync}$ and its seed value would be a function of k_{psk}. Most important is that both Alice and Bob use the same deterministic way to obtain the subset.

5 Encrypted sequences

To combat the (relatively) predictable nature of PRNGs, the article [1] suggests using a form of encryption over the generated channel number

$$c_i' = enc(c_i, k_j) \ mod \ n_{ch}$$

where k_j is the key used for symmetric encryption algorithm. The channel number c_i' would then be used instead of c_i to determine the operating frequency. I propose a simple extension of the scheme by using a PRNGs, like LFSR to generate keys for encryption k_j at each hop. This would require Alice and Bob to have a separate PRNG with its own starting seed for generating keys. Synchronization protocol would be extended to provide a random starting seed for this PRNG as well.

In the article [1] we can find algebraic evaluation of the scheme and also results of simulations when such a scheme is employed.

6 Attacks with software-defined radio

The scheme described is quite secure to jamming attacks and interference, but let us consider a more incognito attack - eavesdropping. It has been said that Eve the attacker can not tune and listen if she does not know the operating frequency. This premise is true if Eve can observe sufficiently small chunk of spectrum at any given time. That is, Eve is not able to tune in if frequency is changing enough such that the next observed hop is likely to be outside Eves current spectrum observation window. This criteria is easily achieved by using very wide spectrum (large channel spacing), however due to practical reasons it is very infeasible. Also there are hardware interface cards used in the field of software-defined radio that can observe the spectrum with bandwidth up to 60 MHz. This is very likely to cover most of if not all of the spectrum available for transmission on a certain frequency band.

With these SDR (software-defined radio) devices it is possible to record and store the raw data from the air. Eve can use this to record the transmission and then find where the transmission was taking place at any given time and effectively reconstruct the transmission without hopping. This is done by observing where are peaks of energy in the spectrum. That would allow her to then listen to the reconstructed stream (without hops) and obtain the data transmitted.

This could be fixed by Alice and Bob by encrypting the data before sending it. It would require more CPU power (which is limited) and might increase latency in time sensitive applications. Also if the encryption scheme used is not secure enough, Eve could break the encryption. It also adds complexity to the system.

7 Dual streams

We have seen in the previous section that wideband SDR devices can allow the attacker to reconstruct the transmission by detecting the spectral power and tuning to that frequency. This attack assumes that there is only one transmitter with similar power active at any given time. If there were two transmitters, it would become very difficult to distinguish which time slot / hop belongs to which transmitter. There is still a way to carefully observe the timing as two distinct transmitters would not be synchronized.

Under careful consideration of this difficulty for the attacker, we can use it to our advantage. Alice can use two transmitters, one for the actual stream of data, and one for decoy. It is of paramount importance that the two transmitters are synchronized and use very similar output power and other parameters. The goal here is that they are indistinguishable by the attacker. If we can achieve this, then we obtain a very powerful tool to protect the transmission.

The extended scheme would then be almost the same as before, we just use two instances of PRNGs and channels and feed each transmitter with this. Synchronization is just extended to send another set of PRNG seeds from Alice to Bob.

7.1 Security

Extended scheme is different from the attackers point of view. If the two transmitters are indistinguishable (apart from transmit frequency) then Eve would have to *choose* which transmitter is the one carrying the data. The important thing to notice here is that the probability of choosing the next

correct signal (of the two) is $\frac{1}{2}$. This is very good for Alice and Bob as it means that in order to reconstruct the whole transmission, it is exponentially hard to obtain the right order of chunks. And more interestingly, it grows with the number of hops:

$$P(\text{correctly choosing signals}) = \frac{1}{2^{N_{hops}}}$$

We can make shorter hops (and thus more hops in the same time) and increase the security even more. A crucial requirement here is also that Eve is not able to distinguish which chunk is the real data and which one the decoy.

This scheme should also allow for some form of forward secrecy, even if Eve captures the whole spectrum to her disk and then later analyzes this offline.

7.2 Considerations

It is worth considering that using dual streams (dual transmitters) is also less practical. It costs more to obtain two transmitters, it is an engineering challenge to synchronize them, and nevertheless consume twice as much power as a single transmitter. Also only half as many distinct users can use the available spectrum, so it is important to consider weather or not to use such a scheme in very crowded and/or limited environment.

8 Conclusion

In this paper I have presented frequency hopping as a way of securing wireless communications from jamming (DoS) attacks and partially form eavesdropping. I have presented a relatively simple scheme and its weaknesses and offered two extensions: encrypted channel numbers and dual transmitters. Each has its benefits and its drawbacks and it depends on the goals and implementations which scheme to use. One can obtain a relatively secure scheme by implementing both extensions.

I would have also liked to present more on how commercial wireless communication devices employ similar schemes but have failed to do so as manufactures go to great lengths to hide their proprietary implementations. It would present a challenge and maybe yield an interesting paper on analyzing these commercial systems, but this is out of scope for this seminar paper.

In conclusion, frequency hopping schemes are a good way to prevent jamming attacks and therefore enable coexistence of many such devices.

References

[1] Ebrahimzadeh, Amirhossein, and Abolfazl Falahati. "Frequency Hopping Spread Spectrum Security Improvement with Encrypted Spreading Codes in a Partial Band Noise Jamming Environment." Journal of Information Security 4, no. 1 (2013): 1.

[2] Capkun, Srdjan. "Uncoordinated Frequency Hopping Spread Spectrum." In Encyclopedia of Cryptography and Security, pp. 1346-1346. Springer US, 2011.

[3] Emek, Yuval, and Roger Wattenhofer. "Frequency hopping against a powerful adversary." In International Symposium on Distributed Computing, pp. 329-343. Springer Berlin Heidelberg, 2013.

[4] Atmel Inc., "Random Number Generation Using AES", URL: `http://www.atmel.com/Images/article_random_number.pdf`, Accessed: 2017-02-07