Verification and Redesign of OFDM Backscatter

Xin Liu[1]*, Zicheng Chi[2]*, Wei Wang[1], Yao Yao[1], Pei Hao[1], and Ting Zhu[1]†
[1]*University of Maryland, Baltimore County*
[2]*Cleveland State University*

Abstract

Orthogonal frequency-division multiplexing (OFDM) has been widely used in WiFi, LTE, and adopted in 5G. Recently, researchers have proposed multiple OFDM-based WiFi backscatter systems [33, 35, 36] that use the same underlying design principle (i.e., codeword translation) at the OFDM symbol-level to transmit the tag data. However, since the phase error correction in WiFi receivers can eliminate the phase offset created by a tag, the codeword translation requires specific WiFi receivers that can disable the phase error correction. As a result, phase error is introduced into the decoding procedure of the codeword translation, which significantly increases the tag data decoding error. To address this issue, we designed a novel OFDM backscatter called TScatter, which uses high-granularity sample-level modulation to avoid the phase offset created by a tag being eliminated by phase error correction. Moreover, by taking advantage of the phase error correction, our system is able to work in more dynamic environments. Our design also has two advantages: much lower BER and higher throughput. We conducted extensive evaluations under different scenarios. The experimental results show that TScatter has i) three to four orders of magnitude lower BER when its throughput is similar to the latest OFDM backscatter system MOXcatter [36]; or ii) more than 212 times higher throughput when its BER is similar to MOXcatter. Our design is generic and has the potential to be applied to backscatter other OFDM signals (e.g., LTE and 5G).

1 Introduction

By reflecting ambient signals, backscatter systems conduct passive communication which can provide low power consumption, low cost, and ubiquitous connectivity for Internet of Things (IoT) devices to support various applications (e.g., smart buildings and smart health) [4]. To achieve ubiquitous connectivity and leverage the advantages of the ambient signals, researchers have developed various backscatter systems that reflect ambient signals from TV or frequency modulation (FM) radio towers, LoRa, or WiFi [7, 12, 16, 27]. Therefore, these backscatter systems are complimentary to each other and have their own unique advantages in terms of energy efficiency, throughput, deployment cost, etc.

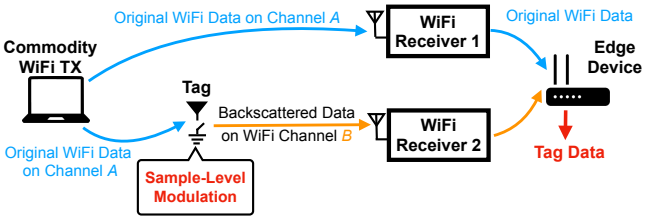


Figure 1: A general architecture of OFDM-based WiFi backscatter systems [33, 35, 36]. Different from existing systems that modulate the tag data at the symbol-level, our backscatter system uses the sample-level modulation (highlighted in a red color).

In this paper, we mainly focus on the design of the WiFi backscatter due to i) pervasively available WiFi signals inside buildings; ii) numerous WiFi devices; and iii) abundant applications supported by WiFi. All of these can significantly increase the adoption of our developed techniques. The pioneer work on WiFi backscatter [11] achieved up to 1 kbps throughput and 2.1 meters range. Follow up works improved the performance of WiFi backscatter by using customized full-duplex WiFi access points [6] or standard 802.11 b/g/n WiFi devices [33, 35, 36]. Since the majority of the WiFi signals in buildings are using an advanced modulation scheme – orthogonal frequency-division multiplexing (OFDM), researchers recently proposed a general architecture of OFDM-based WiFi backscatter systems (shown in Fig. 1) to leverage productive WiFi data communication from the surrounding WiFi devices. This architecture has demonstrated to be effective in supporting the smart offices application, in which WiFi receivers can be connected by Ethernet backhaul to decode the tag data [33, 35, 36].

However, these systems [33, 35, 36] require specific WiFi receivers that can disable the phase error correction. This is because the phase error correction can eliminate the phase offset created by the tag, and cause incorrect tag data decoding.

*Both authors contributed equally to the paper.
†Ting Zhu is the corresponding author.

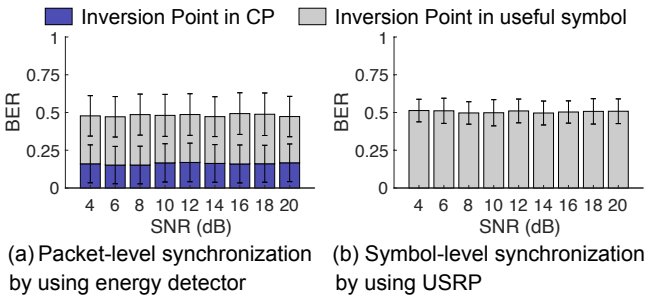


Figure 2: High BER identified in experiments with a total number of 128,000 data points transmitted

On the other hand, the phase error correction is very important for the WiFi demodulation because phase errors in the WiFi systems may be dynamically changing due to the changes of environment (e.g., temperature of the oscillators and object movement). Therefore, WiFi protocols always arrange a certain number of pilot subcarriers in each OFDM symbol to track and correct phase errors. Without the phase error correction, the WiFi demodulation error will increase. Similarly, disabling the phase error correction will also introduce the phase error into the demodulation procedure of the codeword translation, which will increase the tag decoding error.

To demonstrate the limitation of these OFDM-based WiFi backscatter systems, we rebuilt these systems using USRPs that ran the standard 802.11g stack which contains the phase error correction. We identified the high bit error rate (BER) which happens even when the signal to noise ratio (SNR) is high (shown in Fig. 2). By using an energy detector (which is a low-power component to measure the signal strength in the air) as introduced in previous systems, the tag is synchronized with the WiFi sender at the packet-level. From the result (Fig. 2(a)), we can observe that the total BER is around 50% even with a high SNR. Since a tag embeds the tag data by inverting the phase of the incoming signal, the inversion point can settle anywhere in a symbol with packet-level synchronization. After conducting a thorough analysis, we found that one portion of the inversion points (in blue) settle in cyclic prefixes (used to prevent inter symbol interference), which causes decoding failures because the cyclic prefix is removed at the receiver. The other portion of the inversion points (in gray) settle in useful symbols (used to carry real WiFi data), however, this portion will also cause the bit error. We also conducted symbol-level synchronization and found that the blue part can be eliminated with a precise synchronization (results shown in Fig. 2(b)). However, the BER is still very high across different SNRs.

To address this issue, we conducted comprehensive studies and designed a new OFDM backscatter system called TScatter to achieve higher reliability (lower BER) and higher throughput. We propose a sample-level modulation scheme, in which the phase offsets on pilot subcarriers are very different from that on the data subcarriers. Thus, the phase offsets on the data subcarriers cannot be eliminated by the phase error correction. Therefore, we can extract the tag data while the phase error correction is present. Moreover, since the phase error correction is present, our system is able to work in more dynamic environments and achieve much lower BER.

Our sample-level backscatter system has a lot of benefits. On one hand, the system provides high reliability (low BER) allowing tags to work in various scenarios, such as non-line-of-sight or underground. On the other hand, our TScatter can be configured in high throughput mode with Mbps level throughput to support IoT edge computing applications such as ubiquitous surveillance in smart buildings. Further more, high throughput provides high energy efficiency (bit/joule). Given the same amount of harvested energy, our backscatter system can transmit more data than existing ambient WiFi backscatter systems. Moreover, higher energy efficiency also provides another benefit – given the same amount of data to be transmitted, our backscatter system needs much less energy. Therefore, our backscatter system can be deployed in places that are further away from the wireless energy transferring sources than the latest WiFi backscatter systems [33, 35, 36].

The main contributions of this paper are as follows:

• We verified existing OFDM-based WiFi backscatter systems and redesigned the system with a novel sample-level modulation technique, which can retrieve tag data from the process of phase error correction in modern OFDM-based wireless communication system. Potentially, this technique can be applied to LTE and 5G which also use OFDM modulation schemes.

• To demodulate the high granularity modulated data, we built demodulation models that capture the WiFi demodulation procedure and derive a minimization function to estimate the tag data. We further enhanced our modulation and demodulation design to support different WiFi modulation schemes (e.g., BPSK, QPSK, 16QAM, and 64QAM). Our evaluation results demonstrate the effectiveness of our design.

• We built a hardware prototype of our proposed backscatter system that contains a low power FPGA and a simple RF switch. We also designed a tag IC for estimating the power consumption. Our empirical results show that TScatter has i) three to four orders of magnitude lower BER when its throughput is similar to the latest OFDM backscatter system MOXcatter [36]; or ii) more than 212 times higher throughput when its BER is similar to MOXcatter.

2 Background of Existing OFDM Backscatter

To fully reveal the impact of phase error correction on the codeword translation technique, it is necessary to first understand how the technique works. Generally, current OFDM backscatter techniques mainly consist of three parts: (i) the excitation OFDM signal generation on the sender side; (ii) the phase modulation on the tag side; and (iii) the codeword decoding on the receiver side.

The Sender: To produce the OFDM symbol, the sender conducts an Inverse Discrete Fourier Transform (IDFT) on its subcarriers in the frequency domain. The output samples of

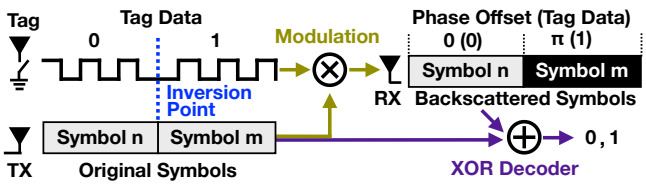


Figure 3: Overview of the existing OFDM backscatter technique

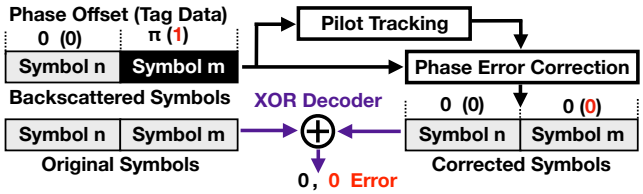


Figure 4: The phase offset π is eliminated by phase error correction, which causes tag data 1 to be mistakenly decoded as tag data 0.

the corresponding IDFT will form a single OFDM symbol $\mathcal{S}(t)$ in the time domain, which can be represented as:

$$\mathcal{S}(t) = \text{IDFT}\ [\ \mathcal{X}_k\ ] \tag{1}$$

where $\{\mathcal{X}_k\}$ are the subcarriers. Multiple symbols are concatenated to create the final excitation OFDM signal.

The Tag: Existing OFDM-based WiFi backscatter systems allow a tag to convey information by inverting the phase of OFDM symbols in the time domain [33, 35, 36]. An example is shown in Fig. 3, the backscatter tag uses zero phase offset to transmit data zero and a phase inversion to transmit data one. Then, the backscattered symbol $\mathcal{B}(t)$ is given by:

$$\mathcal{B}(t) = \begin{cases} \mathcal{S}(t)e^{j0} & \text{Tag data 0} \\ \mathcal{S}(t)e^{j\pi} & \text{Tag data 1} \end{cases} \tag{2}$$

To improve the data rate, the tag can create additional phase offsets to covey more information, which is shown below:

$$\mathcal{B}(t) = \begin{cases} \mathcal{S}(t)e^{j0} & \text{Tag data 00} \\ \mathcal{S}(t)e^{j\frac{\pi}{2}} & \text{Tag data 01} \\ \mathcal{S}(t)e^{j\pi} & \text{Tag data 10} \\ \mathcal{S}(t)e^{j\frac{3\pi}{2}} & \text{Tag data 11} \end{cases} \tag{3}$$

We note that to achieve above codeword translations (Eqn. 2 and 3), the tag needs to synchronize the tag data with the symbol. Formally, we define the point that concatenates two different tag data as the tag data **inversion point**. As shown in Fig. 3, the tag data inversion point (from 0 to 1) is aligned with the beginning of the OFDM symbol m.

The Receiver: A Discrete Fourier Transform (DFT) is performed to convert the backscattered symbols to the frequency-domain backscattered subcarriers. Because of the linearity of the DFT, the operation of the phase change in the time domain corresponds to the frequency multiplication. Therefore, the backscattered subcarriers can be represented as:

$$\mathcal{X}_k e^{j\delta} = \text{DFT}\ [\ \mathcal{B}(t)\ ] \tag{4}$$

where δ is the phase offset. Then, we can decode the tag data by computing the XOR operation of backscattered subcarriers and original subcarriers:

$$\mathcal{X}_k e^{j\delta} \oplus \mathcal{X}_k = \begin{cases} \text{Tag data 0} & \delta = 0 \\ \text{Tag data 1} & \delta = \pi \end{cases} \tag{5}$$

3 Why Existing OFDM Backscatter Systems Disable Phase Error Correction?

The underlying assumption in the codeword translation is that backscatter systems can be free of the effect of the phase error. However, we point out that failure to consider the effect of phase error will significantly increase the BER of OFDM backscatter systems. Therefore, in this section, we first investigate influences of the phase error correction on codeword translation. Then, we extensively analyze all the scenarios that will affect the BER of the codeword translation. At last, we outline the desired properties to reduce the BER of OFDM backscatter systems.

3.1 Tag Data is Eliminated

In real-world scenarios, due to the changes of environment (e.g., temperature of the oscillators and object movement), the OFDM receiver is required to perform the phase error correction to eliminate the phase error on the signal. This process also eliminates the phase offset created by the tag.

Specifically, the phase error correction is to use available pilot subcarriers $\{\mathcal{X}_p\}$ to track the phase error. Since the phase error can be simply modeled as a constant multiplicative component across a symbol (e.g., $e^{j\phi}$ [25]), the backscattered subcarrier in Eqn. 4 should be $\mathcal{X}_k e^{j(\delta+\phi)}$ while the backscattered pilot subcarrier should be $\mathcal{X}_p e^{j(\delta+\phi)}$. Then, the phase error $\mathcal{H}$ can be computed by comparing backscattered pilot subcarriers and original pilot subcarriers:

$$\mathcal{H} = \frac{\sum \mathcal{X}_p e^{j(\delta+\phi)}}{\sum \mathcal{X}_p} = e^{j(\delta+\phi)} \tag{6}$$

Finally, the backscattered subcarriers can be corrected as:

$$\mathcal{X}_k e^{j(\delta+\phi)} \cdot \mathcal{H}^{-1} = \mathcal{X}_k \tag{7}$$

The key observation from the above expression is that the phase error $e^{j\phi}$, as well as the phase offsets $e^{j\delta}$ from the codeword translation, are eliminated from subcarriers. In other words, as shown in Fig. 4, if the tag transmits arbitrary data, the XOR decoder may never output data 1. This is because the codeword translation makes the pilot subcarriers have the same phase offset as other subcarriers, which makes it feasible to eliminate phase offsets by the phase error correction. As a result, since phase offsets are eliminated, tag data 1 will be mistakenly decoded as tag data 0 (i.e., zero phase offset in Eqn. 2). The effect of phase error correction will become more severe when the tag increases its data rate. As shown in Eqn. 3, the tag uses four phase offsets to double the data rate. However, due to the phase error correction, the receiver may not extract these phases offsets correctly. In this case, the tag data 01, 10, 11 will be mistakenly decoded as tag data 00.

The above analysis mainly focuses on the scenario that the inversion point is perfectly aligned with the beginning of the

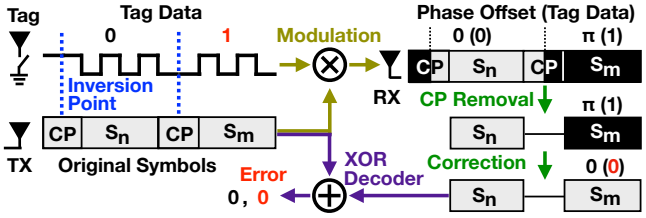


Figure 5: When the inversion point is located in the CP, the phase offset π representing tag data 1 can still be eliminated.

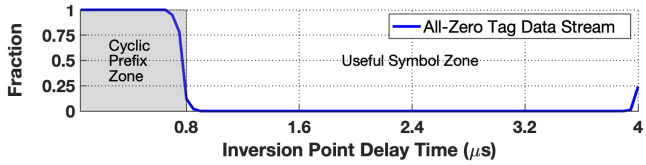


Figure 6: Experiments demonstrate that if the inversion point is within the cyclic prefix zone, the decoded tag data are all zeros regardless whatever data is sent by the backscatter tag.

symbol. In following sections, we will show that even when the inversion point is not perfectly aligned with the beginning of the symbol (i.e., the inversion point is in the cyclic prefix or the inversion point is in the useful symbol), the receiver may still face high bit error rate.

3.2 Inversion Point in Cyclic Prefix

The XOR decoder cannot extract the tag data correctly when the tag data inversion point is located in the cyclic prefix (CP). Because the OFDM receiver will first remove CP to prevent intersymbol interference introduced by the multipath effect, the inversion point in the CP is removed as well. As shown in Fig. 5, after the CP removal, although the useful symbol $\mathcal{S}_m$ still has the phase offset π, the phase error correction will eliminate the phase offset, which causes the decoding error.

We conduct an experiment to show the effect of the inversion point in the cyclic prefix. In this experiment, an 802.11g OFDM-based WiFi physical layer is implemented on the USRP B210 [1]. In order to precisely control the synchronization between the OFDM symbol and the tag data, the tag is connected to the USRP using two wires: one for common ground and another for signal. When the USRP transmits the WiFi signal to the air, it will also transmit the starting signal through the wire to tell the tag to embed tag data in the WiFi signal. By modifying the delay after the starting signal, we can synchronize the tag data inversion point with the different timings in the OFDM symbol duration (4μs). The original tag data is an arbitrary data stream. However, as shown in Fig. 6, when the inversion points are within the cyclic prefix zone (i.e., the inversion point delay is less than 0.7 μs), the decoded tag data are all zeros, because the phase offsets of the tag data 1 are eliminated. When the inversion points are within the useful symbol zone, the decoded tag data is not all zeros. However, as we will discuss in Section 3.3, it may still have a high BER. When the inversion point delay is between 0.7 μs and 0.8 μs, although it is theoretically within the cyclic prefix zone, the decoded tag data is not all zeros. This is because of the multipath effect in the real-world, so the inversion point may be moved to the useful symbol zone.

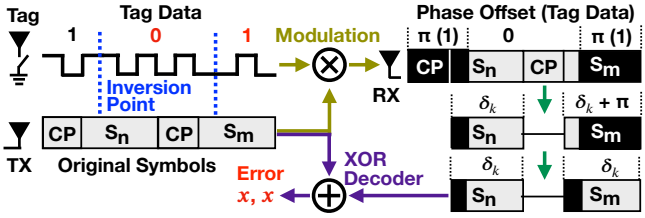


Figure 7: When the inversion point is located in the useful symbol, the decoding result is the same and thus causes the decoding error.

3.3 Inversion Point in Useful Symbol

In this section, we analyze the scenario that the inversion point is located in the useful symbol. As shown in Fig. 7, since the phase offsets introduced in the symbols of $\mathcal{S}_n$ and $\mathcal{S}_m$ are opposite, the subcarriers of $\mathcal{S}_n$ and $\mathcal{S}_m$ have the opposite phase offsets δ_k and $\delta_k + \pi$ after the CP removal, where k is the subcarrier index. Obviously, the subcarriers of $\mathcal{S}_m$ have a common phase offset π, which can be eliminated by the phase error correction. Therefore, the final phase offsets on $\mathcal{S}_n$ and $\mathcal{S}_m$ are almost the same, which causes the same decoding result and corresponding errors. We note that δ_k on each subcarrier is different and determined by the inversion point position. Hence, the decoding result x might be 1 or 0.

3.4 Desired Properties and Our Solution

From the analysis in Sec. 3.2 and 3.3, the OFDM backscatter should satisfy the following two properties to avoid the decoding error caused by the phase error correction:

1. *The tag data inversion point should fall into the useful symbol.*
2. *The tag data should be represented by phase offsets that cannot be eliminated.*

Since the backscatter tag is a low power device, we cannot use a high power circuit to conduct precise synchronization between the inversion point and the useful symbol. To overcome this challenge, we explore a high granularity modulation scheme, sample-level modulation. By doing this, the phase offsets on the pilot subcarriers are different from that on the data subcarriers. Thus, the phase offsets on the data subcarriers cannot be eliminated by the phase error correction.

To satisfy the second property, we utilize orthogonal coding technique for reliable data transfer. Specifically, to reduce the bit error rate and achieve high reliability, we propose orthogonal pseudo-random noise (PN) sequences to represent the tag data. With the help of the sample-level modulation, each data bit in the PN sequence can represent a phase offset. As a result, the sequences of phase offsets are also orthogonal and the tag data can avoid being eliminated.

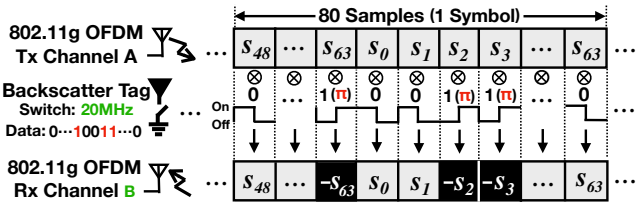


Figure 8: When a switching cycle is equal to a sample duration, the switching action can appropriately change the phase of the sample.

By satisfying the two properties, compared to prior works [33, 35, 36], our backscatter system can take advantage of the phase error correction on the demodulation side and is able to achieve much lower BER. On the modulation side, since multiple tag data bits can be transmitted within one symbol by using our sample-level modulation, more robust coding methods (such as orthogonal code or convolutional code) than codeword translation can be utilized to further improve the BER.

4 Sample-Level Modulation

Our sample-level modulation has two design goals: i) embedding the tag data in the OFDM signal at the sample-level, and ii) minimizing the interference from the original band channel. In order to achieve the first design goal, we propose to use the tag to change the phase of any sample in the OFDM symbol. To achieve the second design goal, the tag needs to shift the center frequency of the backscattered samples to the adjacent band channel. Fig. 8 shows how TScatter leverages the switching action to change the phase of the sample. For the sake of simplicity, we first assume that the on and off edges of switching actions are aligned with the samples. We discuss the situation that the switching actions and the samples are not edge-aligned in Sec. 6.1.

A fundamental basis for our sample-level modulation is that the OFDM symbol can be divided into the sample-level, whose duration is much shorter than the symbol-level. When 802.11g OFDM generates a symbol in the transmitter, the IDFT converts the 64 subcarriers into a 64-sample sequence. The samples can be represented by rewriting Eqn. 1 with the definition of IDFT:

$$\mathcal{S}_n = \text{IDFT}\,[\,\mathcal{X}_k\,] = \frac{1}{64}\sum_{k=0}^{63}\mathcal{X}_k e^{j2\pi kn/64} \tag{8}$$

Where, $n \in \{0,\ldots,63\}$, $\mathcal{S}_n$ denotes the n'th sample and X_k denotes the k'th subcarrier. To prevent inter-symbol interference, the last 16 samples $[\mathcal{S}_{48},\ldots,\mathcal{S}_{63}]$ are replicated in front of the 64 samples. Thus, a symbol consists of a total of 80 samples.

When the tag toggles its RF switch to backscatter the OFDM signals, it essentially uses square waves to modulate the phases of the samples. We use θ_n to represent the initial phase of the n'th square wave $\mathcal{W}_n$. Then, $\mathcal{W}_n$ can be represented using a Fourier series as follows [9]:

$$\mathcal{W}_n(f_s,\theta_n) = 0.5 + \frac{2}{\pi}\sum_{m=1,3,5,\ldots}^{\infty}\frac{1}{m}cos(2\pi m f_s t + \theta_n) \tag{9}$$

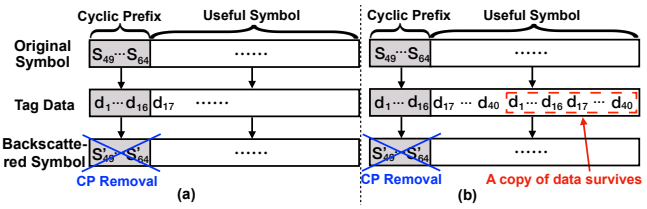


Figure 9: In (a), the tag data embedded in the CP are lost due to the CP removal. In (b), the tag embeds 40 bits of data twice in the symbol. A copy of data can survive after the CP removal.

Where, f_s is the toggling frequency of the switch.

Since each symbol is 4 μs long and contains 80 samples, when the tag toggles its switch at $f_s = 20MHz$, the switching cycle (0.05 $\mu s = \frac{1}{20MHz}$) will be equal to the sample duration (0.05 $\mu s = \frac{4\,\mu s}{80}$). Therefore, the switching action can appropriately modulate the phase of the sample. The n'th backscattered sample can be calculated by multiplying the n'th sample and the 1st harmonic of the n'th square wave:

$$\begin{aligned}\mathcal{S}_n\mathcal{W}_n^{m=1} &= \frac{2}{\pi}\mathcal{S}_n cos(2\pi f_s t + \theta_n)\\ &= \frac{1}{\pi}\{\mathcal{S}_n e^{j\theta_n}e^{j2\pi f_s t} + \mathcal{S}_n e^{-j\theta_n}e^{-j2\pi f_s t}\}\end{aligned} \tag{10}$$

Summary: From Eqn. 10, one can observe that our two design goals are achieved. First, there are two backscattered samples $\mathcal{S}_n e^{j\theta_n}$ and $\mathcal{S}_n e^{-j\theta_n}$. They are formed by changing the phase of the original sample $\mathcal{S}_n$ by θ_n and $-\theta_n$, respectively. If we change θ_n according to the tag data, the tag data is embedded into the sample. For example, if the tag wants to use the 4-phase scheme (0, $\pi/2$, π and $3\pi/2$) to transmit the tag data, we can define the tag data '00' as $\theta_n = 0$, '01' as $\theta_n = \pi/2$, '10' as $\theta_n = \pi$ and '11' as $\theta_n = 3\pi/2$. Second, the two backscattered samples are multiplied by $e^{j2\pi f_s t}$ and $e^{-j2\pi f_s t}$, respectively, which means the backscattered samples are shifted by $f_s = \pm 20MHz$ into the adjacent band channels and thus isolated from the original band channel. Therefore, a OFDM receiver can obtain the backscattered samples without the interference from the original band channel [34]. For these two sidebands, one is desired and the other is unwanted and wasted. The unwanted sideband can be easily eliminated by making the signal have a negative copy on the unwanted sideband as introduced in HitchHike [32].

5 Tag Coding Scheme

In this section, we discuss the coding methods on the tag, which aim to avoid the tag data inversion point being removed by the cyclic prefix removal and the phase offset being eliminated by the phase error correction, and make TScatter achieve highly reliable backscatter communication.

5.1 Surviving from Cyclic Prefix

Since the receiver removes the cyclic prefix (i.e., the first 16 samples of each symbol) directly prior to the procedures in the OFDM module, a portion of the tag data may be removed as well because the tag may embed this data in the cyclic

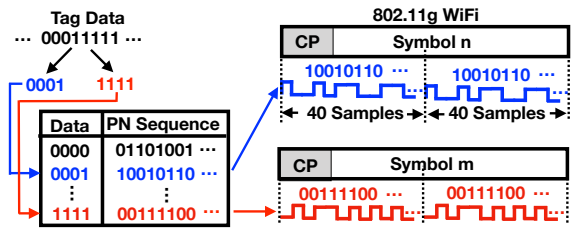


Figure 10: TScatter uses PN sequences to represent the tag data.

prefix. Fig. 9(a) shows an example that the tag data d_1 to d_{16} are lost due to the CP removal.

A naive solution is to utilize an envelope detector to detect the start of the cyclic prefix and embed the tag data in the useful symbol. However, it is unlikely to obtain an accurate CP detection using a packet-level detection on the envelope detector. Instead, TScatter simply embeds 40 bits of tag data twice on 80 samples in a symbol (shown in Fig. 9(b)). By doing this, there still exits a 40-sample sequence which stores a copy of 40 bits of tag data after the CP removal. Hence, the tag does not have to rely on the synchronization accuracy of the envelope detector to avoid the cyclic prefix. This approach can be easily implemented on the tag without extra energy cost or computing resources.

At the receiver side, to decode the 40 bits of tag data in each symbol, we need to know which sample in the symbol is the starting sample of the 40-sample sequence. To find the starting sample, we designed a backscatter header (shown in Appendix B) with a duration of two symbols. The backscatter header is predefined as a flag sequence. Even though a portion of the flag sequence is removed due to the CP removal, there still exits a backscattered symbol whose samples are modulated only by the flag sequence. We add a sliding-window-based algorithm to the decoding algorithm (detailed in Sec. 6.3) to search which part of the flag sequence is the best matching decoding result of this symbol . Then we can obtain the position of the starting sample. When the decoding result matches the two-symbol long predefined backscatter trailer, it signifies the end of the tag data.

5.2 Coding Scheme for Low BER

To avoid the phase offsets being eliminated, TScatter uses predefined nearly orthogonal pseudo-random noise (PN) sequences to represent the tag data (Appendix A lists the PN sequence table). As shown in Fig. 10, the tag data is divided into multiple groups. Each group will be spread to a special 40-bit long PN sequence. Then each bit in the PN sequence is transmitted by using the sample-level modulation. With the help of the sample-level modulation, each bit in the PN sequence can represent a phase offset. As a result, the sequences of phase offsets are orthogonal to each other and the tag data can avoid being eliminated. As described in Sec. 5.1, to avoid being removed by the CP removal, each sequence is transmitted twice. To reduce the coding complexity on the tag, we implant a lookup table of the sequences to map the tag data to the corresponding PN sequence.

Using orthogonal PN sequences to represent the tag data has three benefits. First, since the sequences are orthogonal to each other, the effective signal-to-noise ratio (SNR) of backscattered signals is improved at the receiver side. Second, since the sequences are predefined, the receiver can solve the decoding algorithm by directly finding which sequence is the best-match rather than estimating the sequence, which lowers the computational complexity (detailed in Sec. 6.3). Third, TScatter can be used in lower order modulation schemes of OFDM, such as BPSK or QPSK.

5.3 Coding Scheme for High Throughput

The coding scheme for high throughput is a combination of convolutional code and predefined data when the OFDM sender utilizes higher order modulation schemes, such as 64QAM or 16QAM. The convolutional code is a rate 2/3 feedforward encoder. The predefined data is to increase the estimation accuracy by constricting the estimation result to a smaller range of values during the decoding process (detailed in Sec. 6.3). For 64QAM, in the 40-sample sequence, we use 32 samples to embed the convolutional coded tag data. The remaining 8 samples are modulated by the predefined data. For 16QAM, we use 18 samples to embed the convolutional coded tag data and 22 samples to embed the predefined data.

6 Decoding of Tag Data

One benefit of our sample-level modulation is that it does not require an accurate synchronization with the OFDM sender because the phase offset caused by a lack of accurate synchronization can be corrected by the phase error correction. In this section, we first analyze the phase offset, then describe the OFDM demodulation model and mathematically demonstrate how the phase offset is corrected by the OFDM receiver. In the end, we describe how to decode the tag data.

6.1 Phase Offset Analysis

The phase offset can be characterized by three factors:

• *Envelope Detection Delay.* The tag leverages an envelope detector to identify the WiFi transmission. However, due to an uncertain delayed response, the detector cannot accurately identify when the sample starts. Then it always occurs that the switching actions and the samples are not edge-aligned. We can consider a time delay τ that exists between them and causes a phase offset $\beta = 2\pi f_s \tau$.

• *Wireless Channel.* The wireless channel introduces a phase offset of ϕ to the samples.

• *Frequency Offset.* The frequency offset causes two phenomenas: carrier frequency offset (CFO) and sampling frequency offset (SFO). In Appendix B and C , we demonstrate the CFO and SFO can be estimated and corrected.

Thus, a backscattered sample can be expressed as follows:

$$\mathcal{R}_n = e^{j(\beta+\phi)} e^{j\theta_n} \mathcal{S}_n \tag{11}$$

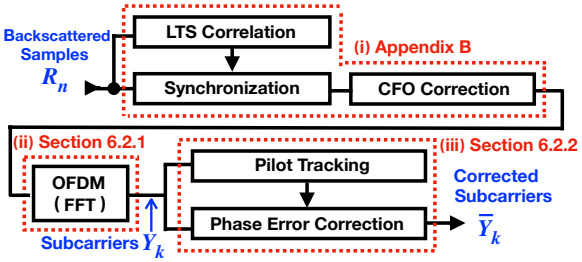


Figure 11: The demodulation process in 802.11g WiFi receiver from the backscattered samples to the corrected subcarriers.

In Sec. 6.2.2, we demonstrate that the phase offset caused by the envelope detection delay and wireless channel can be eliminated and do not affect the values of data subcarriers.

6.2 Demodulation Model

The objective of TScatter's demodulation is to decode the tag data on OFDM subcarriers. To achieve this objective, we need to understand how the OFDM receiver converts the backscattered samples to the subcarriers. To do so, we divide the workflow of an 802.11g receiver into 3 parts (shown in Fig. 11), and build models for these parts.

6.2.1 Modeling OFDM

The 80 backscattered samples of each symbol first go through the CP removal module, where the first 16 samples are removed. Then the remaining 64 samples are applied to the DFT to generate the complex values of OFDM subcarriers. The subcarrier's value can be derived as:

$$Y_k = \text{DFT}\,[\,\mathcal{R}_n\,] = e^{j(\beta+\phi)} \sum_{n=0}^{63} e^{j\theta_n} \mathcal{S}_n e^{-j2\pi nk/64} \quad (12)$$

Where, $k \in \{0,...,63\}$ and Y_k is the complex value of the k'th OFDM subcarrier. Eqn. 12 shows that the tag data is transfered from the backscattered samples in time domain to the subcarriers in frequency domain by OFDM.

6.2.2 Modeling Phase Error Correction

While the tag data is transfered into the subcarriers, the phase offset $\beta+\phi$ is also transfered into the subcarriers (shown in Eqn. 12), which may increase the tag data decoding error. We had an important observation that the OFDM receiver can leverage pilot subcarriers to eliminate the phase offset. In this section, we mathematically demonstrate the procedure of phase error correction.

The phase error estimation is performed by calculating the rotating phase of the four pilot subcarriers. In the sender, the four pilot subcarriers are $\{\mathcal{X}_{11},\mathcal{X}_{25},\mathcal{X}_{39},\mathcal{X}_{53}\}$ and their initial values are $\{1,1,1,-1\}$ [2]. In the receiver, we use $\{Y_{11},Y_{25},Y_{39},Y_{53}\}$ (shown in in Eqn. 12) to represent the four pilot subcarriers. The rotating phase of the pilot subcarriers can be estimated as:

$$\begin{aligned}\Psi &= \angle(Y_{11}+Y_{25}+Y_{39}-Y_{53}) - \angle(\mathcal{X}_{11}+\mathcal{X}_{25}+\mathcal{X}_{39}-\mathcal{X}_{53}) \\ &= \beta+\phi+\angle\Gamma \end{aligned} \quad (13)$$

Where,
$$\Gamma = \sum_{n=0}^{63} e^{j\theta_n} \mathcal{S}_n \Big(\sum_{k=11,25,39} e^{-j2\pi nk/64} - e^{-j2\pi n\frac{53}{64}}\Big) \quad (14)$$

From Eqn. 13 and 14, we observe that a new phase offset $\angle\Gamma$ is introduced which is caused by the backscatter modulation on the pilot subcarriers. Eqn. 13 also illustrates that although the backscatter modulation change the pilot subcarriers, the phase offset $\beta+\phi$ is isolated from the tag data.

Since 48 out of the 64 subcarriers are used to carry the payload data ($k \in \{6,...,58\}$ and $\notin \{11,25,32,39,53\}$), the OFDM receiver multiplies 48 data subcarriers by $e^{-j\Psi}$ to correct the phase offset:

$$\begin{aligned}\overline{Y}_k &= e^{-j\Psi}\, Y_k \\ &= e^{-j\angle\Gamma} \cancel{e^{-j(\beta+\phi)}} \cancel{e^{j(\beta+\phi)}} \sum_{n=0}^{63} e^{j\theta_n} \mathcal{S}_n e^{-j2\pi nk/64}\end{aligned} \quad (15)$$

Where,
$$e^{-j\angle\Gamma} = \frac{Re\{\Gamma\} - j*Im\{\Gamma\}}{\sqrt{Re\{\Gamma\}^2 + Im\{\Gamma\}^2}} \quad (16)$$

Eqn. 15 demonstrates that the phase offset $\beta+\phi$ is eliminated and does not affect the tag data decoding. On the other hand, Eqn. 15 also demonstrates that the values of data subcarriers were affected twice: The first one happens during the backscatter modulation stage, when the values of the data subcarriers were changed according to the tag data; the second one happens during the phase error correction stage, when the values of data subcarriers were corrected by the phase offset calculated from pilot subcarriers. Although Γ exists in Eqn. 15, Eqn. 14 shows that the only unknown value in Γ is the tag data. Therefore, we can leverage Eqn. 15 to decode the tag data without being affected by the phase offset $\beta+\phi$.

6.3 Decoding Tag Data

Eqn. 15 shows that the phase change on each subcarrier is different and determined by all 64 backscattered samples. If the tag data on any backscattered sample change, the phase change on each subcarrier changes. Therefore, we must use all subcarriers' information to decode each tag data.

We observe that the inputs of Eqn. 15 are the tag data θ_n and the original OFDM sample $\mathcal{S}_n$, while the outputs of Eqn. 15 are the corrected data subcarrier $\overline{Y}_k$. Hence, if we obtain the values of $\mathcal{S}_n$ and $\overline{Y}_k$, we can decode θ_n.

The values of original OFDM sample $\mathcal{S}_n$ can be calculated by using Eqn. 8. However, the challenge is that we cannot obtain the real value of $\overline{Y}_k$. Normally, OFDM backscatter uses the coded data linear transform technique in [33, 35, 36] to track the phase change on the subcarriers. If we follow the same technique to get the subcarrier values, we find that the

subcarrier values deviate from $\overline{Y}_k$. This is because the OFDM receiver maps $\overline{Y}_k$ to the nearest QAM points (we assume $\overrightarrow{Y}_k$).

Since $\overrightarrow{Y}_k$ are the nearest QAM points of $\overline{Y}_k$, we can estimate the tag data θ_n by calculating the minimum Euclidean distance between the corrected data subcarrier values $\overline{Y}_k$ and their mapped points $\overrightarrow{Y}_k$:

$$\begin{bmatrix} \tilde{\theta}_0 \\ \vdots \\ \tilde{\theta}_{63} \end{bmatrix} = \arg\min_{\substack{[\theta_0,\ldots,\theta_{63}] \\ k\neq 11,25,32,39,53}} \sum_{k=6}^{58} ||\overrightarrow{Y}_k - \overline{Y}_k||_2 \quad (17)$$

$$= \arg\min_{\substack{[\theta_0,\ldots,\theta_{63}] \\ k\neq 11,25,32,39,53}} \sum_{k=6}^{58} ||\overrightarrow{Y}_k - e^{-j\angle\Gamma} \sum_{n=0}^{63} e^{j\theta_n} \mathcal{S}_n e^{-j2\pi nk/64}||_2$$

Where, k are the indexes of the data subcarriers, $\overrightarrow{Y}_k$ are provided by the receiver and Γ can be represented using Eqn. 14. The only unknown value in Eqn. 17 is the tag data θ_n. Therefore, we can leverage Eqn. 17 to decode the tag data.

Since each tag data θ_n has limited values (i.e., either 0 or π), Eqn. 17 is a constrained linear least-squares problem. We note that the left side of Eqn. 17 is 64 unknown values $(\tilde{\theta}_0,\ldots,\tilde{\theta}_{63})$, while we only have 48 data subcarrier values $\overrightarrow{Y}_k$ ($k \in \{6,...,58\}$ and $\notin \{11,25,32,39,53\}$) on the right side. That means Eqn. 17 is not full row-rank. Mathematically, to determine a unique solution using Eqn. 17, we need to minimize the number of unknown values on left side to 48. A simple way to do this is that the number of unknown tag data in each symbol is no more than 48 and the remaining tag data are predefined. For example, we predefine $e^{j\theta_{48,\ldots,63}} = e^{j0} = 1$ and calculate the unknown data $e^{j\theta_{0,\ldots,47}}$. Therefore, the left side (the unknown tag data) and the right side (the data subcarrier values) have the same size, and Eqn. 17 is now full row-rank. In fact, Eqn. 17 illustrates that the maximum number of samples used to store the unknown tag data in each symbol is 48. This is because we rely on the data subcarriers' values to calculate the unknown tag data and each symbol provides only 48 data subcarriers. The coding scheme in Sec. 5.1 requires that the number of the tag data embedded in each symbol is no more than 40, which is also less than 48.

Eqn. 17 is solved by using the matrix decomposition, whose computational complexity is $O(n^3)$ [3], where n is the number of the unknown tag data. When n gets to the maximum value (i.e., 48), there are $n^3 = 110592$ floating-point operations. A low power edge computing platform Jetson Nano [21] (only cost $99) implemented on an ARM A57 processor with 472G floating-point operations per second, can solve the problem in $\frac{110592}{472*10^9} = 0.25\mu s$, which is less than a symbol duration $4\mu s$.

7 Lite Version of TScatter

In prior works [33, 35, 36], the codeword translation works only when the phase error correction is disabled. In this section, we demonstrate that TScatter can also use the codeword translation at the symbol level without disabling the phase error correction and introduce the lite version of TScatter.

Prior works use the symbol-level modulation (i.e., phase change once per symbol) to realize the codeword translation. Based on our observation from Sec. 3.3, when the inversion point settles in the useful symbol, the corrected subcarriers are different from the original ones. Since the symbol-level modulation is essentially a special case of the sample-level modulation (i.e., multiple phase changes per symbol) by setting the number of phase changes as 1 within one symbol, it is possible to use the codeword translation to achieve the basic idea of TScatter without disabling the phase error correction. We call this approach Lite TScatter because its design principle is derived from our sample-level modulation by setting all the tag data values inside a symbol to be all-one or all-zero.

Specifically, the whole procedure of Lite TScatter is as follows: before the tag transmits the data, it first transmits a long all-one (or all- zero) sequence to reflect the preamble and several symbols. Since there is no phase change in the long sequence, the corrected subcarriers are the same as the original ones. Then, the tag transmits the first data. The first data is predefined, and its value should be different from that of the long sequence. For example, if the long sequence is all-one, the first data should be defined as zero. Since there is a phase change between the long sequence and the first data, we can find the first symbol which is different from the original one. Next, the tag can transmit arbitrary data. If the corrected subcarriers are the same as the original ones, it implies there is no phase change and the tag data should be equal to the last one. If the corrected subcarriers are different from the original ones, it implies there is a phase change and the data should be different from the last one. Therefore, we can decode the tag data one by one. By modifying the codeword translation decoder in Eqn. 5, the decoding function of Lite TScatter is as follows:

$$\text{Tag data } d_n = \begin{cases} d_{n-1} & \mathcal{X}_k e^{j\delta_k} \oplus \mathcal{X}_k = 0 \\ 1 - d_{n-1} & \mathcal{X}_k e^{j\delta_k} \oplus \mathcal{X}_k \neq 0 \\ \text{Predefined} & n = 1 \end{cases} \quad (18)$$

Eqn. 18 provides a lite version of TScatter, which enables codeword translation work with WiFi protocols that enable the phase error corrections. However, Lite TScatter's throughput is relatively low because the tag data is modulated at the symbol level instead of the sample level.

8 Evaluation

In this section, we present the implementation of TScatter and show the experimental results of our extensive evaluation.

8.1 Implementation

TScatter (shown in Fig. 12) is implemented on an open-source backscatter platform [29], which mainly consists of two components: a Microsemi AGLN250 low power FPGA and an ADG902 RF reflective switch. To compare different levels of

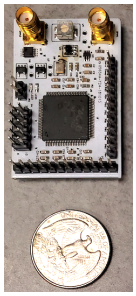


Figure 12: TScatter Tag

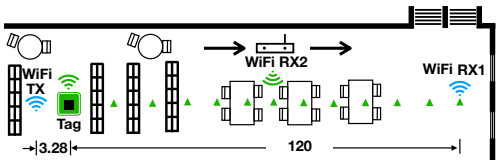


Figure 13: Library (in feet)

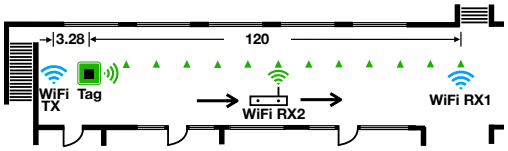


Figure 14: Hallway (in feet)

coding and modulation schemes, we implemented both the coding schemes of **binary** and **nibble** (i.e., 4-bit) (shown in Appendix A) and the modulation schemes of **2-phase** and **4-phase** (introduced in Sec. 4) in the FPGA.

To conduct fair comparisons, the distance between the WiFi transmitter and the backscatter tag is 1 meter (i.e., 3.28 feet), which is the same as the one in FreeRider [33]. The WiFi transmitter is implemented on a ThinkPad T420s laptop which transmits the IEEE 802.11g compatible signals. The WiFi transmitter utilizes 64QAM as its modulation scheme by default. In Sec. 8.6, we evaluate TScatter's performance with different WiFi modulation schemes.

To extensively evaluate TScatter's performance, the experiments were conducted in three different scenarios: **Library** (Fig. 13), **Hallway** (Fig. 14), and **Stadium**. The WiFi receivers are implemented using DELL XPS 9550 laptop and USRP-B210 with 802.11g PHY layer. WiFi receiver 1 is used to obtain the original WiFi information. WiFi receiver 2 listens for the backscattered WiFi signal. To assess the system performance, we use the following two metrics:

Throughput: The throughput is defined as correctly demodulated data bits at PHY layer.

Bit error rate (BER): The BER is defined as the number of bit errors divided by the total number of transmitted bits.

8.2 Comparing with State-of-the-art

We first compare the performance of TScatter with the reported best results produced by recent OFDM-based WiFi backscatter systems (i.e., X-Tandem [35], MOXcatter [36] and FreeRider [33]). Fig. 15 shows BER results when TScatter and state-of-the-art solutions have similar level of throughput. As we can observe from this figure, the BERs of TScatter are much lower than that of the state-of-the-art solutions. Fig. 16(a) shows the comparison of throughput. We can observe that, with 4-phase modulation scheme, TScatter achieves 10.61 Mbps. The main reason TScatter achieves orders of magnitude better performance is that it takes advantage of the phase error correction. State-of-the-art solutions require specific commodity NICs that can disable the phase error correction. However, the phase error correction is very important for the OFDM demodulation because the phase error may be dynamically changing due to the changes of environment

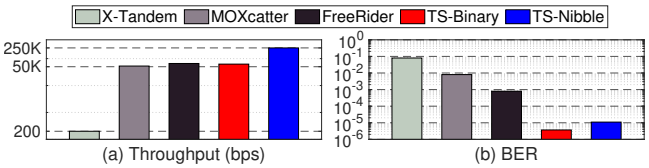


Figure 15: Comparing with state-of-the-art with similar throughput

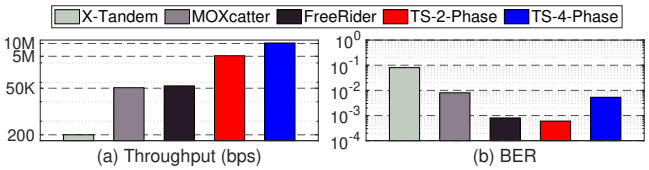


Figure 16: Comparing with state-of-the-art with similar BER

(e.g., temperature of the oscillators and object movement). Without the phase error correction, the OFDM demodulation error will increase. Similarly, disabling phase error correction will also introduce phase error into the decoding procedure of the codeword translation, which will increase the tag data decoding error. In contrast, TScatter takes the phase error correction into consideration. On the demodulation side, TScatter builds the demodulation model from the backscattered samples to the corrected subcarriers that can not only correct the phase error due to the dynamic environments but also decode the tag data by estimating the minimal Euclidean distance rather than by using XOR. On the modulation side, TScatter leverages the sample-level modulation to apply more robust coding methods to further improve the BER or throughput. As a result, the BERs of TScatter are around $\mathbf{10^2}$, $\mathbf{10^3}$ and $\mathbf{10^4}$ **times** as low as that of FreeRider, MOXcatter and X-Tandem, respectively. The throughputs of TScatter are 53,050, 212, and 176 times higher than X-Tandem, MOXcatter, and FreeRider, respectively.

8.3 Library

To understand how TScatter works in a multipath rich environment, we conducted the experiments in a library (shown in Fig. 13). Since there are a lot of shelves, tables, and chairs in the library, these obstacles not only block the direct line-of-sight transmission path but create more multipath effects as well. In this setting, the backscatter tag and WiFi transmitter are deployed on the shelves. We move the WiFi receiver from shelves to tables to vary the multipath environment.

Fig. 17 and Fig. 18 show the throughput and BER over different communication distances, respectively. As we can observe from these figures, the throughput of TScatter is similar over the change of distances while the BER increases slowly. When the communication distance reaches 120 feet, the average BERs of TScatter are still around 10^{-3} and 10^{-4} for Nibble and Binary, respectively. Since the multipath effect will become more severe as the distance increases in the library scenario, we can conclude that TScatter is resistant to the wireless channel interference. Because the demodulation process (described in Sec. 6.1) takes the phase error intro-

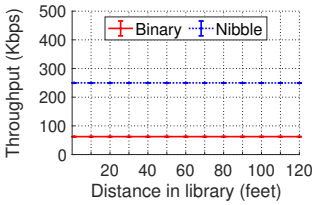


Figure 17: Throughput

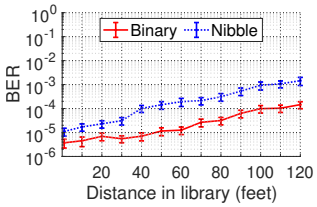


Figure 18: BER

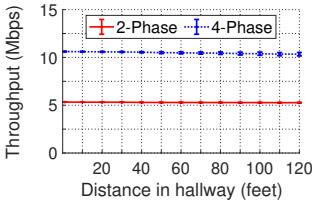


Figure 19: Throughput

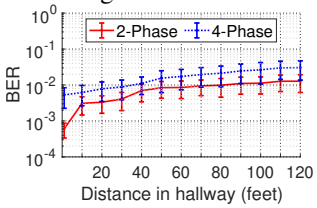


Figure 20: BER

duced by the wireless channel into consideration, TScatter shows great advantages in this scenario.

8.4 Hallway

In this section, we evaluate TScatter in the hallway scenario. As shown in Fig. 14, the hallway is on the second floor of an academic building. This scenario is selected to reflect TScatter's real-world performance with less environmental impact because the hallway is relatively spacious. In this scenario, the backscatter tag and the WiFi receiver were kept in line-of-sight. The distance between the tag and the receiver is changing from one foot to 120 feet.

Fig. 19 shows the throughput of our TScatter system using different modulation schemes (i.e., 2-phase and 4-phase). Overall, TScatter can achieve a maximum of 10.61 Mbps and the throughput is stable over different communication distances because it has a high granularity modulation scheme. Fig. 20 shows the BER under different communication distances. We can observe that the BER for the 2-phase modulation scheme is very low. The average BER is lower than 1% when the distance increases to 70 feet. Even when the distance increases to 120 feet, the average BER is still around 1%. This is because our demodulation model captures the impact of the phase error correction on the received subcarriers.

8.5 Outdoor Stadium

To understand the performance of TScatter with different modes (i.e., low BER, high throughput, and lite version) in the same scenario, we conduct experiments in an outdoor stadium. The distance between the tag and the receiver is changing from one foot to 160 feet.

Fig. 21 shows the throughput over different communication distances. The high throughput mode (2-phase) outperforms the low BER mode (binary) and Lite TScatter because it has a high granularity modulation scheme. When the receiver is 160 feet away from the tag, the throughput of binary, 2-phase, and Lite TScatter are around 62.5 Kbps, 5.26 Mbps and 62.3 Kbps, respectively. Fig. 22 shows the BERs at different

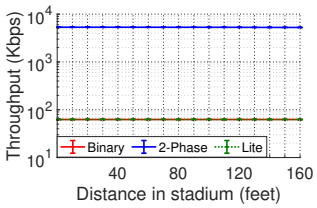


Figure 21: Throughput

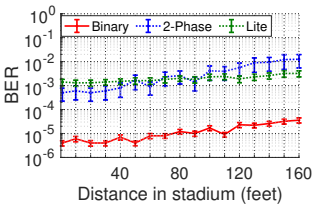


Figure 22: BER

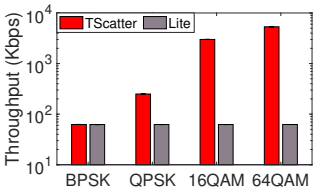


Figure 23: Throughput v.s. WiFi OFDM modulation scheme

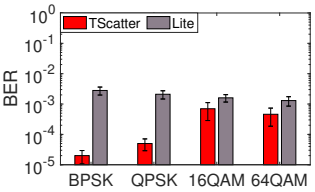


Figure 24: BER v.s. WiFi OFDM modulation scheme

distances. The BER of low BER mode and Lite TScatter is more stable than that of high throughput mode despite the fact that the signal strength degrades across distance. Specifically, the BER of low BER mode is much less than that of the other two modes. Therefore, the low BER mode has the significant advantage on providing reliable backscatter communications. When the distance reaches 160 feet, the average BERs of binary, 2-phase, and Lite TScatter are still around 10^{-5}, 10^{-2} and 10^{-3}, respectively.

8.6 Impact of OFDM Modulation Scheme

In Sec. 5.2 and 5.3, we introduced how TScatter deals with lower and higher level WiFi OFDM modulation schemes respectively. In this section, we compare sample-level modulation TScatter's performance (in terms of throughput and BER with low BER or high throughput coding scheme) with Lite TScatter (which uses symbol level modulation) under different WiFi OFDM modulation schemes (i.e., BPSK, QPSK, 16QAM, and 64QAM). Fig. 23 shows the throughput result. We observe that the throughput of Lite TScatter is almost the same under different OFDM modulation schemes because one bit of data is embedded in per symbol no matter what the OFDM modulation scheme is. For TScatter, the throughput increases while the OFDM modulation scheme changes from the lowest level (i.e., BPSK) to the highest level (i.e., 64QAM). At 64QAM, the throughput reaches 5.33 Mbps. The reason of this growing trend is that higher level OFDM modulation schemes can provide higher resolution to decode the tag data, which can accommodate more tag data per symbol.

Fig. 24 shows the result of BER versus OFDM modulation scheme. We can observe that TScatter's BER is lower than Lite TScatter's across all kinds of modulation schemes even when the OFDM transmitter utilizes BPSK. This is because the demodulation model captures the impact of the phase error correction on the received subcarriers. Moreover, TScatter leverages the sample-level modulation to apply more robust coding methods to further improve the BER.

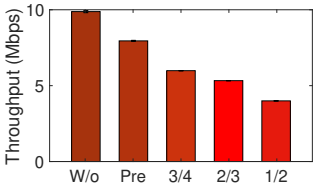


Figure 25: Throughput v.s. Tag Coding Scheme

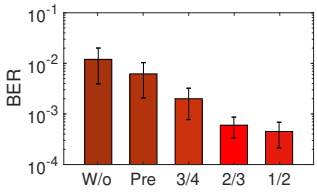


Figure 26: BER v.s. Tag Coding Scheme

8.7 Impact of Tag Coding Scheme

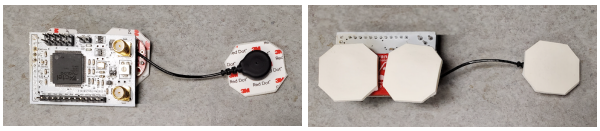
In Sec. 5.3, we have described TScatter's coding schemes for high throughput. Fig. 25 and 26 show TScatter's performance with different code rates. "W/o" denotes the results without using coding scheme. "Pre" denotes the results that TScatter only uses predefined data without coding. "3/4", "2/3", and "1/2" denote the results that TScatter uses both predefined data and coding with a code rate at 3/4, 2/3, and 1/2, respectively. Since the experiments show similar trends, we show the results with the 2-phase modulation scheme in the hallway scenario. The WiFi transmitter utilizes 64QAM.

When no coding scheme is utilized, the whole 40-sample sequence is used to store the unknown tag data (described in Sec. 5.1). Consequently, TScatter has the highest throughput (shown in Fig. 25) as well as the highest BER (shown in Fig. 26) among all coding schemes. The reason of the high BER is that the corrected data subcarrier values deviate from their mapping points, which cause estimation errors in Eqn. 17. To reduce the estimation errors, we replace 8 unknown tag data with the predefined data to constrict the estimation result to a smaller range of values. Fig. 26 shows that the BER decreases to below 1%. To achieve a more reliable communication, we combine the predefined data and the convolutional code to generate the tag data. When the code rate is up to 2/3, the BER is lower than 0.1% and the throughput is as high as 5.33 Mbps, which provides a reliable high throughput communication.

8.8 Energy Consumption Analysis

We designed an integrated circuit for TScatter's digital processing module and conducted a simulation using Cadence Spectre for TSMC 65nm process. In the power consumption simulation, multiple factors are considered including Vdd (1.6V $\sim$ 2V), temperature (25°C), system clock frequency, and power mode.

The IC design consists of four modules: clock, energy detector, control module and RF transistor. TScatter first detects the WiFi packet by using the energy detector. When the WiFi packet is detected, the clock synthesizes a 20MHz frequency for the control module. Then the control module reads the data from the sensor, latches and codes the sensor data, and toggles the RF transistor according to the coded data. Since the power consumption of the energy detector and the RF transistor is very low (around 0.3μW), we mainly introduce the implementation details of the clock and the control module:



(a) Front (b) Back

Figure 27: EMG connected TScatter tag

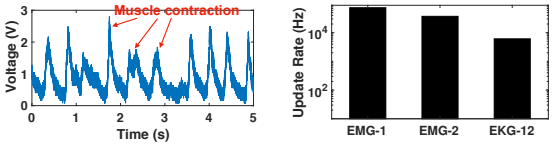
Clock. The bottleneck of the power consumption of the TScatter system is the clock. We use a ring oscillator to synthesize a 20MHz frequency. The frequency accuracy is sensitive to temperature change. We add a thermistor to design a temperature compensation circuit to compensate for the frequency drift. Moreover, we add a trimming pad to correct the frequency. The simulation result shows that the power consumption of the ring oscillator is 23.7μW.

Control module. The control module contains a cache circuit to store the sensor data, a code circuit to code the sensor data, and an inverter to toggle the RF transistor as a fan-out. The cache is constructed by the basic cache cell. The cache cell is composed of one D latch and one transmission gate. Through connecting the output of D latch to the input of the transmission gate, a cache cell with simultaneous read and write capability is created. The reason we use the latches rather than flash to store the sensor data is that the power consumption of latches is very low even when they are working at 20MHz. The code circuit contains a lookup table for low BER mode, which is composed of 32 PN sequences (shown in Appendix B), and a convolutional encoder for high throughput mode, which is composed of 7 latches and 3 XOR gates. The lookup table is also constructed by basic logic gates. The inverter is minimum sized. The simulation result shows that the power consumption of the control module is around 6.2μW.

From the above analysis, the overall power consumption of TScatter is **30.2μW**, which is similar to that of other OFDM backscatter systems. This is mainly because the most power-hungry part of the backscatter tag is the clock. No matter the sample-level modulation or the symbol-level modulation, they all need to shift the incoming WiFi signal to the adjacent band channel to minimize the interference from the original band channel, which can be done by toggling the RF switch at a specific frequency. The minimum value of the frequency equals a WiFi channel bandwidth, i.e., 20MHz. Consequently, to toggle the RF switch at 20MHz, the logic control modules of the symbol-level modulation and the sample-level modulation are all operating at 20MHz. Although TScatter performs the convolutional coding in high throughput mode, the encoder which is constructed by 7 D latches and 3 XOR gates has a very low power consumption even when it works at 20MHz. Therefore, the energy consumptions of TScatter and other OFDM backscatter systems are similar.

9 Application

In this section, we show that biometric measurements (such as EMG and EKG) can be wirelessly transmitted in real-time by using TScatter. Fig. 27(a) (front side) and Fig. 27(b)



(a) EMG data (b) Update Rate (log scale)

Figure 28: (a) shows the EMG data while the system is attached to arm muscle and the participate is doing dumbbell lifting. Each peak corresponds to a muscle contraction. (b) shows the average update rate when the system is connected with 1-channel EMG (sleeping monitoring), 2-channel EMG (fitness monitoring), and 12-channel EKG.

(back side) show a prototype that an EMG sensor that is connected to the TScatter tag. By using foam electrodes, the tag and sensor can be attached to a muscle to record the electrical activity produced by the muscle. Fig. 28(a) shows one set of data while the system is monitoring arm activities. In this figure, each peak corresponds to a muscle contraction. Fig. 28(b) shows the average update rate under different configurations: i) EMG-1, 1-channel EMG is attached to a leg muscle for monitoring sleep abnormalities [5]; ii) EMG-2, 2-channel EMG is attached to upper and lower arm for fitness monitoring; and iii) EKG-12 , 12-channel EKG is generated to test how TScatter works with multi-channel EKG. With a typical sampling rate at 1 KHz and 24-bit resolution, the required bit rate for 12-channel EKG is 288 Kbps. With different modulation schemes, our TScatter can provide around 5 $\sim$10 Mbps throughput at the physical layer. Even after deducting the performance loss due to the wireless signal attenuation caused by the human body and upper layer overheads (e.g., ACKs), TScatter's throughput should be sufficient for 12-channel EKG measurements.

10 Related Work

Recently, backscatter has played an attractive role in the passive communication field because of its significant performance on power consumption. Researchers have proposed various kinds of novel techniques [8, 10, 17–20, 26, 30, 31] to support various applications. For example, ReMix [26] enables backscattering of deep tissue devices by overcoming interference from the surface of a human body and localizing the in-body backscatter device. Living IoT [8] enables smart farming applications by placing backscatter devices on live insects. NICscatter [31] introduces a backscatter communication method on commercial Wi-Fi NICs that enables malware to covertly convey information. PAB [10] enables long-term ocean applications by backscattering acoustic signals in underwater environments.

To leverage existing infrastructures, such as TV band [16, 22], FM radio [27], LoRa [7, 23, 24], Bluetooth [9], ZigBee [13], and WiFi [6, 11, 12, 32–37], researchers have also explored various solutions. Specifically, in the WiFi backscatter field, Passive WiFi [12] demonstrates for the first time that the backscatter can generate 802.11b transmissions. Interscatter [9] shows that Bluetooth transmissions can be used to create 802.11b transmissions using backscatter communication. HitchHike [32] allows a tag to backscatter existing 802.11b transmissions from a commodity WiFi transmitter. To further support more complex WiFi structures, such as OFDM, WiFi Backscatter [11] conveys information by modulating the CSI and RSSI measurements at the OFDM receivers. BackFi [6] uses customized full-duplex devices to clean out the effect during backscattering OFDM signals. FS-Backscatter [34] minimizes the interference from the original band by shifting the backscattered OFDM signals to an adjacent band. FreeRider [33] proposes the OFDM-based codeword translation technique, which piggybacks the tag data by changing the phase of backscattered OFDM symbols. Built on top of FreeRider, MOXcatter [36] and X-Tandem [35] realize a MIMO OFDM backscatter system and a multi-hop OFDM backscatter system, respectively. However, FreeRider, MOXcatter, and X-Tandem require specific WiFi receivers that can disable the phase error correction.

Different from the above backscatter systems, TScatter is the first work that conducts sample-level coding and subcarrier-level decoding based on OFDM waveforms. More importantly, it is able to achieve orders of magnitude lower bit error rate or much higher throughput than existing approaches.

11 Discussion & Conclusion

Although our system was tested using USRPs, we believe that our Lite TScatter can potentially be deployed on commodity WiFi devices with little modification because it is a natural extension of FreeRider and MOXcatter. Based on the description of prior works and WiFi specification [2, 14, 33, 35, 36], to implement the full version of TScatter on commodity devices, we expect to address the following engineering problems: 1) Since the backscattered WiFi packets have been modified, commodity WiFi receivers may drop these packets due to their failure to pass the cyclic redundancy check (CRC). To address this problem, we need to use a WiFi card that can be configured into monitor mode which can obtain packets with bad checksums [33]; 2) we need to know the scrambling seed (i.e., the initial state of shift register) to decode the tag data. For different WiFi cards, we need to take different approaches. For example, in ath5k supported WiFi cards (e.g., Atheros AR5112 and AR2425), the scrambling seed can be derived by setting the register of the driver [14].

We note that the main contribution of this paper is a novel backscatter design principle (i.e., sample-level modulation) for OFDM backscatter. We extensively evaluated the performance of TScatter in various real-world scenarios. Evaluation results demonstrate that TScatter is able to achieve orders of magnitude lower bit error rate or much higher throughput than existing approaches [33, 35, 36].

Acknowledgments. This work is in part supported by NSF grants CNS-1824491 and CNS-1652669. We thank our shepherd Dr. Shyam Gollakota and other reviewers for their valuable comments.

References

[1] https://github.com/bastibl/gr-ieee802-11.

[2] Ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012) (Dec 2016).

[3] Computational complexity of mathematical operations — Wikipedia. https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations, 2019. [Online].

[4] BAHL, V. Edge computing: a historical perspective and direction.

[5] BASTUJI, H., AND GARCA-LARREA, L. Sleep/wake abnormalities in patients with periodic leg movements during sleep: factor analysis on data from 24-h ambulatory polygraph. Journal of Sleep Research.

[6] BHARADIA, D., JOSHI, K. R., KOTARU, M., AND KATTI, S. Backfi: High throughput wifi backscatter.

[7] HESSAR, M., NAJAFI, A., AND GOLLAKOTA, S. Netscatter: Enabling large-scale backscatter networks. NSDI '19.

[8] IYER, V., NANDAKUMAR, R., WANG, A., FULLER, S. B., AND GOLLAKOTA, S. Living iot: A flying wireless platform on live insects. MobiCom'19.

[9] IYER, V., TALLA, V., KELLOGG, B., GOLLAKOTA, S., AND SMITH, J. Inter-technology backscatter: Towards internet connectivity for implanted devices. SIGCOMM '16.

[10] JANG, J., AND ADIB, F. Underwater backscatter networking. SIGCOMM '19.

[11] KELLOGG, B., PARKS, A., GOLLAKOTA, S., SMITH, J. R., AND WETHERALL, D. Wi-fi backscatter: Internet connectivity for rf-powered devices. SIGCOMM '14.

[12] KELLOGG, B., TALLA, V., GOLLAKOTA, S., AND SMITH, J. R. Passive wi-fi: Bringing low power to wi-fi transmissions. SIGCOMM '16.

[13] LI, Y., CHI, Z., LIU, X., AND ZHU, T. Passive-zigbee: Enabling zigbee communication in iot networks with 1000x+ less power consumption. SenSys'18.

[14] LI, Z., AND HE, T. Webee: Physical-layer cross-technology communication via emulation. In Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MobiCom '17.

[15] LING, P., LU, F., SNOEREN, A. C., AND VOELKER, G. M. Enfold: Downclocking ofdm in wi-fi. GetMobile: Mobile Comp. and Comm..

[16] LIU, V., PARKS, A., TALLA, V., GOLLAKOTA, S., WETHERALL, D., AND SMITH, J. R. Ambient backscatter: Wireless communication out of thin air. SIGCOMM '13.

[17] LIU, X., CHI, Z., WANG, W., YAO, Y., AND ZHU, T. Vmscatter: A versatile MIMO backscatter. NSDI'20.

[18] LUO, Z., ZHANG, Q., MA, Y., SINGH, M., AND ADIB, F. 3d backscatter localization for fine-grained robotics. In 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), NSDI'19.

[19] NADERIPARIZI, S., HESSAR, M., TALLA, V., GOLLAKOTA, S., AND SMITH, J. R. Towards battery-free HD video streaming. NSDI '17.

[20] NANDAKUMAR, R., IYER, V., AND GOLLAKOTA, S. 3d localization for sub-centimeter sized devices. Sensys'18.

[21] NVIDIA. Jetson nano brings ai computing to everyone. https://devblogs.nvidia.com/jetson-nano-ai-computing/, 2019.

[22] PARKS, A. N., LIU, A., GOLLAKOTA, S., AND SMITH, J. R. Turbocharging ambient backscatter communication. SIGCOMM '14.

[23] PENG, Y., SHANGGUAN, L., HU, Y., QIAN, Y., LIN, X., CHEN, X., FANG, D., AND JAMIESON, K. Plora: A passive long-range data network from ambient lora transmissions. SIGCOMM '18.

[24] TALLA, V., HESSAR, M., KELLOGG, B., NAJAFI, A., SMITH, J. R., AND GOLLAKOTA, S. Lora backscatter: Enabling the vision of ubiquitous connectivity. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol..

[25] VASISHT, D., KUMAR, S., RAHUL, H., AND KATABI, D. Eliminating channel feedback in next-generation cellular networks. SIGCOMM'16.

[26] VASISHT, D., ZHANG, G., ABARI, O., LU, H.-M., FLANZ, J., AND KATABI, D. In-body backscatter communication and localization. SIGCOMM '18.

[27] WANG, A., IYER, V., TALLA, V., SMITH, J. R., AND GOLLAKOTA, S. Fm backscatter: Enabling connected cities and smart fabrics. NSDI'17.

[28] XIE, Y., LI, Z., AND LI, M. Precise power delay profiling with commodity wifi. MobiCom '15.

[29] XU, C., AND ZHANG, P. Open-source software and hardware platforms for building backscatter systems. GetMobile: Mobile Comp. and Comm..

[30] XU, X., SHEN, Y., YANG, J., XU, C., SHEN, G., CHEN, G., AND NI, Y. Passivevlc: Enabling practical visible light backscatter communication for battery-free iot applications. MobiCom'17.

[31] YANG, Z., HUANG, Q., AND ZHANG, Q. Nicscatter: Backscatter as a covert channel in mobile devices. MobiCom '17.

[32] ZHANG, P., BHARADIA, D., JOSHI, K., AND KATTI, S. Hitchhike: Practical backscatter using commodity wifi. SenSys '16.

[33] ZHANG, P., JOSEPHSON, C., BHARADIA, D., AND KATTI, S. Freerider: Backscatter communication using commodity radios. CoNEXT '17.

[34] ZHANG, P., ROSTAMI, M., HU, P., AND GANESAN, D. Enabling practical backscatter communication for on-body sensors. SIGCOMM '16.

[35] ZHAO, J., GONG, W., AND LIU, J. X-tandem: Towards multi-hop backscatter communication with commodity wifi. MobiCom '18.

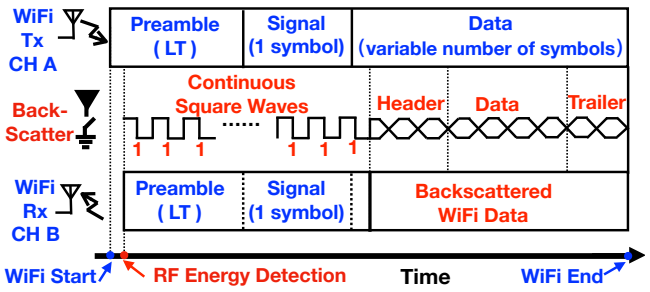


Figure 29: Synchronization between the WiFi sender and the WiFi receiver. The backscatter packet contains four parts: Continuous Square Waves (CSW), Header (in Sec. 5.1), Data and Trailer (in Sec. 5.1). CSW is only a series of square waves and used to reflect the preamble without embedding tag data.

[36] ZHAO, J., SGONG, W., AND LIU, J. Spatial stream backscatter using commodity wifi. MobiSys '18.

[37] ZHAO, R., ZHU, F., FENG, Y., PENG, S., TIAN, X., YU, H., AND WANG, X. Ofdma-enabled wi-fi backscatter. MobiCom '19.

Appendix A Map Tag Data to PN Sequence

No.	Tag Data	PN Sequence
1	0000	01101001111101110110011001111010101001010
2	0001	10010110110111010001000001011111100010 10
3	0010	10001011010100110110100111110010100110000
4	0011	11110000010001110101111010100000100011111
5	0100	00101000010001001000110110011110111011 1
6	0101	11010111011010001100011110000110011000 01
7	0110	01011100111010100110100001001001011101 01
8	0111	00100111111110100101101100110000100110100
9	1000	01010011011101000111000001010111010111 00
10	1001	11001100110100101101001100110001100111 00
11	1010	10110001110111001010101010011000100011 10
12	1011	11101011110010011001110111001110100110 01
13	1100	00010010110010110100111011110100110 0001
14	1101	10001101011001110000010011101000011101 11
15	1110	01100110011001011010101100100011011000 11
16	1111	00111100011110100111010100011110010001 0

Appendix B Synchronization Between Sender and Receiver & CFO Correction

In this section, we build a mathematical model for the synchronization and carrier frequency offset (CFO) correction (i.e., part (i) in Fig. 11). Once the WiFi transmission is identified, the WiFi receiver first leverages the preamble to synchronize the symbol timing and then correct the CFO. Therefore, we start with an analysis of how the tag performs the synchronization between the WiFi sender and the WiFi receiver.

Fig. 29 shows our synchronization solution. The preamble includes two long training (LT) sample sequences. Each sequence is composed of 64 samples. By identifying where the two peak values of the correlation between the received samples and training pattern occur, the WiFi receiver can find where the symbol starts [15]. Therefore, to avoid changing the training sample's value, when TScatter detects the WiFi transmission by using the RF energy detector, it first uses continuous square waves to only reflect the preamble and does not embed any data into the preamble. After the square waves, TScatter starts to modulate the WiFi samples.

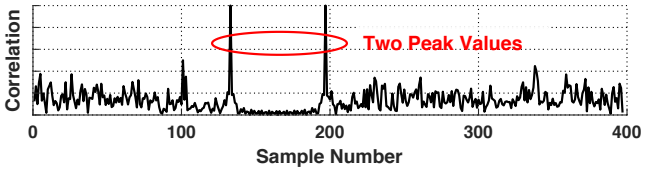


Figure 30: By obtaining the indexes of two peak values, the WiFi receiver can achieve a precise synchronization.

One may ask whether the phase offset (described in Sec. 6.1) will change the preamble value, therefore affects the synchronization. We note that TScatter does not change the preamble sample's value. Without loss of generality, let us assume the phases of continuous square waves are $\theta_n = 0$ and the training pattern value is $\overline{LT}_i$. Then, the positions of two peak values of the correlation between the received samples $\mathcal{R}_l$ (described in Eqn. 8) and the training pattern $\overline{LT}_i$ are:

$$\begin{aligned}(l_1,l_2) &= \underset{l}{\text{argmax}_2}||\sum_{i=0}^{63}(\mathcal{R}_{l+i} * \overline{LT}_i)||_2 \\ &= \underset{l}{\text{argmax}_2}||e^{j[\beta+\phi]}e^{j*0}\sum_{i=0}^{63}(\mathcal{S}_{l+i} * \overline{LT}_i)||_2 \\ &= \underset{l}{\text{argmax}_2}||\sum_{i=0}^{63}(\mathcal{S}_{l+i} * \overline{LT}_i)||_2 \end{aligned} \quad (19)$$

Where, argmax_2 provides the two sample numbers (l_1,l_2) for two peak correlation values.

Eqn. 19 proves that the positions of two peak correlation values are only related to the original sample values $\mathcal{S}_l$ and are not affected by the phase offset. Therefore, the phase offset will not affect the synchronization. Our experimental result (shown in Fig. 30) also illustrates that when the tag uses continuous square waves to reflect the WiFi signals, it does not interfere with the positions of two peak correlation values, and the receiver can achieve a precise synchronization.

In the WiFi receiver, the received sample at frequency f_c is down-converted to baseband with a local carrier frequency $(1+\varepsilon)f_c$. At the receiver side, the frequency offset εf_c causes two phenomenas: carrier frequency offset (CFO) and sampling frequency offset (SFO). The CFO introduces an extra phase offset of $\alpha = 2\pi\varepsilon\frac{f_c}{f_s}$ on the samples during each down-conversion procedure. When the l'th sample of the frame is down-converted, the phase offset applied to the sample is accumulated to $l\alpha$.

$$\mathcal{R}_{l,n} = e^{j(\beta+\phi)}e^{j(l\alpha)}e^{j\theta_n}\mathcal{S}_n \quad (20)$$

Where l and n point to the same sample. Since a WiFi frame contains multiple symbols, l is the sample number in the frame, while n is the sample number in the symbol.

Since the tag does not affect the synchronization, the CFO can be estimated accurately. An estimation of the CFO can be obtained simultaneously when the synchronization is achieved. According to the WiFi specification [2], the WiFi receiver uses LT samples to estimate the CFO. The starting LT sample number is computed as l_1. Let us denote the t'th LT sample as $\mathcal{R}_{l_1+t}$. The CFO estimator is given by:

$$\alpha = \frac{1}{64}\angle\left(\sum_{t=0}^{63} \mathcal{R}^{*}_{l_1+t}\ \mathcal{R}_{l_1+t+64}\right) \tag{21}$$

To correct the CFO, the received sample $\mathcal{R}_{l,n}$ is multiplied by $e^{-jl\alpha}$. Through the synchronization, the WiFi receiver learns the sample number l. Thus, the received sample can be updated to the corrected sample S_n:

$$\mathcal{R}_u = \mathcal{R}_{l,n} * e^{-jl\alpha} = e^{j(\beta+\phi)} e^{j\theta_n} x_n \cancel{e^{jl\alpha} e^{-jl\alpha}} \tag{22}$$

Eqn. 22 shows that the impact of CFO is canceled by the CFO correction module in the WiFi receiver.

Appendix C SFO Correction

Like the CFO, an SFO also exists, which causes a phase offset to OFDM subcarriers after the DFT [28]:

$$\alpha_{sfo} = -2\pi\varepsilon r k \frac{N+N_g}{N} \tag{23}$$

Where, $N = 64$ is the DFT size, $N_g = 16$ is the cyclic prefix length, r is the symbol number which is learned through the synchronization, k is the subcarrier number and ε can be calculated from the estimated CFO ($\varepsilon = \frac{\alpha f_s}{2\pi f_c}$, described in Sec. 6.1). The OFDM subcarriers are corrupted as $Y_k e^{j\alpha_{sfo}}$.

From Eqn. 23, we know the key to correct the SFO is to estimate CFO accurately. In Appendix B, we have demonstrated the tag does not change the preamble value so that the receiver can achieve a precise CFO estimation. The output subcarriers are multiplied by $e^{-j\alpha_{sfo}}$ (i.e., $Y_k = Y_k \cancel{e^{j\alpha_{sfo}} e^{-j\alpha_{sfo}}}$) to correct the offset.