SubScatter: Sub-symbol WiFi Backscatter for High Throughput

Longzhi Yuan and Wei Gong*
University of Science and Technology of China
longzhi@mail.ustc.edu.cn, weigong@ustc.edu.cn

2022 IEEE 30th International Conference on Network Protocols (ICNP) | 978-1-6654-8234-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICNP55882.2022.9940419

***Abstract*—Throughput is one of the key performance indicators in backscatter communication systems. Existing works either have limited throughput or modify the transmitter or receiver to fit the backscatter tag, thereby causing incompatibility with commodity radios. In this paper, we present SubScatter, which realizes a high throughput and keeps compatibility at the same time. For high throughput, SubScatter uses one CCK-modulated 802.11b WiFi symbol to carry eight tag bits by manipulating the phase of the backscattered signal in eight-time slots of a symbol separately. For compatibility with commercial-off-the-shelf (COTS) radios, SubScatter leverages only the physical service data unit (PSDU) to recover the backscatter modulation in which the tag bits are conveyed. And still, to fit the sub-symbol backscatter modulation, SubScatter calculates the Hamming distance between the binary envelope provided by the synchronization circuit and a reference sequence in real-time for synchronization. Extensive experiments in our prototype have proven the effectiveness of SubScatter. SubScatter achieves a throughput of about 11× over state-of-the-art backscatter systems compatible with COTS radios. The Hamming-distance-based synchronization outperforms the design of merely detecting the change of signal power and helps reduce the bit error rate from over 10% to below 1%.**

***Index Terms*—Design, Internet of Things, Sensor networks**

I. Introduction

Ultra-low-power wireless links have made backscatter more and more popular in both academia and industry. Different from traditional wireless nodes, backscatter tag passively modifies and reflects radio-frequency signals from other radios by toggling an RF-switch. When the RF-switch is in open or closed state, the corresponding reflection coefficient is +1 and -1, respectively. The RF-signal will be multiplied with this coefficient and then reflected. Tag can add a specific frequency or phase shift to this signal to convey its bits by properly scheduling the state of the RF-switch. For example, if tag wants to add a frequency shift of Δf to the ambient RF signal, it just alternately keeps the RF-switch in open and closed states for the time of $\frac{1}{2\Delta f}$. The reflected signal can be seen as the product of the incident RF signal and a square wave with a period of $\frac{1}{\Delta f}$. Ignoring the harmonics caused by square wave, the frequency shift can be realized. Similarly, phase shift can be added to the incident signal by introducing an additional time delay to the square wave above. Commodity radios, including WiFi and Bluetooth, are sensitive to the frequency or phase shift so that the tag information contained in reflected signal can be recovered. Such a manner ensures the backscatter tag to be both energy-efficient and simple in structure. And thus backscatter is especially attractive in power-constrained scenarios such as large-scale Internet of Things (IoT) and sensor networks.

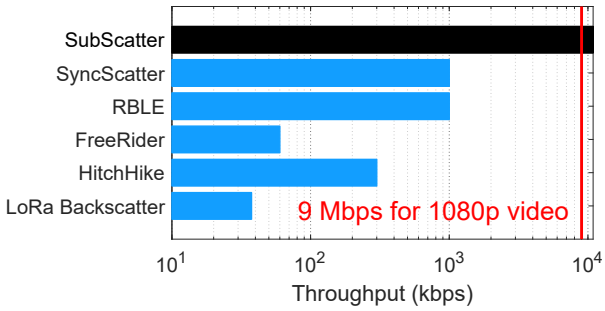


Fig. 1. Existing backscatter systems compatible with COTS radios have a throughput lower than 1 Mbps, while SubScatter achieves over 10 Mbps using sub-symbol modulation.

Recently, many researchers have focused on improving backscatter in transmission throughput and ease of use (such as compatibility with COTS radios[1], using full-duplex transceivers in backscatter system[2], designing underwater backscatter[3], [4], and enabling cross-technology backscatter[5], [6]). Throughput has always been the goal in almost all networks besides backscatter. And compatibility with COTS radios avoids the need for specialized and expensive devices such as the radio frequency identification (RFID) reader and enables backscatter to be widely deployed. They are both of great importance in backscatter systems. Combining those two aspects enables backscatter deployment in many new and interesting scenarios. For example, if we use backscatter tag to record and stream 1080p video to commercial radio, at least 9 Mbps is needed. Higher resolution requires even higher throughput.

Some backscatter systems utilize dedicated devices, such as helper radios or customized readers, to achieve higher throughput as fewer limitations will be encountered. Passive WiFi[7] leverages a dedicated helper radio for a single-tone signal and uses it to generate 802.11b WiFi packets, BLE packets, and ZigBee packets. It achieves a throughput of around 11 Mbps. LScatter[8] and TScatter[9] use LTE and OFDM WiFi signals as their backscatter excitations, respectively. Their data rates are both above 10 Mbps, but tag data can't be recovered

*Corresponding author: Wei Gong.

978-1-6654-8234-9/22/$31.00 ©2022 IEEE

by COTS radios. Dedicated devices such as the universal software radio peripheral (USRP) have to be deployed for the decoding of tag data.

The compatibility with COTS radios has attracted a lot of interest, and great progress has been made. HitchHike[1] introduces the *codeword translation* and enables backscatter to be compatible with commodity 802.11b WiFi radios. Following the idea of *codeword translation*, many backscatter systems, including LoRa Backscatter[10], FreeRider[11], MOXcatter[12], X-Tadem[13], RBLE [14], IBLE[15], and SyncScatter[16] have been designed. But as shown in Fig. 1, their throughputs are all no more than 1 Mbps, much lower than the needed 9 Mbps to transmit video.

We introduce SubScatter, which realizes a throughput of around 11X over state-of-the-art work using commodity WiFi radios. Such promotion comes from two aspects. First, we choose CCK-mode 802.11b as excitation. Its symbol duration is only $\frac{8\mu s}{11}$ instead of $1\mu s$ in 1 Mbps mode, contributing to promotion of $\frac{11}{8}$. Second, eight tag bits instead of a single are embedded in an excitation symbol, causing another 8X throughput gain. Combining those two gains, 11X is realized. The main technical contributions are summarized as follows:

- **Sub-symbol backscatter modulation.**
  We choose the complementary code keying (CCK) 802.11b WiFi signal (11-Mbps mode). Different from OFDM, CCK is a modulation focusing on the time domain. And compared to the 1 Mbps-mode 802.11b WiFi, the CCK-WiFi signal has a shorter symbol duration and each symbol contains eight bits. This helps to realize sub-symbol modulation and improve backscatter throughput. In CCK-WiFi signal, information is expressed in phase. We find that there are parallel phase items containing independent information. Those phase items can be concurrently used for backscatter transmission. The key in sub-symbol backscatter is keeping the tag modulation recoverable merely using the user payload. The Tag firstly transforms 8 tag bits into phase shifts in 8 time slots of a symbol in a way similar to CCK-WiFi, and then adds those phase-shifts to excitation in corresponding time slots by toggling the RF-switch. Eventually, tag modification can be extracted at the commodity receiver by reversing the physical-layer scrambling and CCK-mapping of standard 802.11b WiFi.
- **Minimized Hamming distance for synchronization.**
  In SubScatter, an RF rectifier with a bandwidth of 25 MHz is deployed so that the envelope of the excitation signal can be captured. Then the envelope is quantized into '0' and '1' using a comparator for ease of processing by the FPGA. SubScatter then compares the digital envelope with the pre-stored binary template to get their Hamming distance. Only at the specific instant corresponding to the template will the Hamming distance be minimized. In this way, the backscatter tag can synchronize itself to the excitation signal.
- **Prototype based on COTS devices and extensive experiments for design verification.**
  We use COTS devices including FPGA, harvesting management chip, diodes, and RF-switch to build a battery-free prototype. And experiments have been conducted to show the system effectiveness. Using only office light as the energy source, the tag is able to work normally.

Experiment results on our prototype show that SubScatter achieves a throughput of about 10.9 Mbps and keeps compatibility with commodity WiFi radios at the same time. Furthermore, BER and working range are also good enough for video streaming in an indoor environment.

II. Complementary-code-keying 802.11b WiFi signal as the excitation

We have to choose the proper excitation signal for backscatter. There are various commercial signals, such as WiFi, LTE, Bluetooth, ZigBee, and LoRa. We group and analyze them as follows.

A. Low-rate signals

Signals including Bluetooth, ZigBee, DSSS-802.11b WiFi, and LoRa all have a throughput well below 2 Mbps. If we use them as a carrier for backscatter and deploy a corresponding radio as the receiver, the throughput will be limited below the excitation data rate and the 9Mbps for 1080p video will never be achieved. Therefore, those signals are not our choice.

B. OFDM signals

OFDM technology is used in many radios, including WiFi, LTE, and 5G. The OFDM waveform has a lot of subcarriers in the frequency domain. Every subcarrier is modulated using BPSK, QPSK, or quadrature amplitude modulation (QAM). But because subcarriers are too close to each other and the tag is unable to separate them. If the tag modulates different data on those subcarriers, there will be serious self-influence. Traditional OFDM receivers eliminate such influence using the orthogonality of OFDM itself, which unfortunately cannot be utilized in backscatter tag. The RF filter made up of capacitors and inductors cannot do this job, either, because of the limitation in Q value[17][1]. Taking 802.11n as an example, the frequency band is 2.4 GHz, while the bandwidth of a single subcarrier is about 312.5 KHz. The needed Q-value to separate subcarriers is: $Q = \frac{2.4GHz}{312.5KHz} = 7680$, which is too high. As a result, the backscatter tag is unable to take advantage of the subcarriers in OFDM signals. This limits the backscatter transmission and makes the target throughput of 10 Mbps unachievable. OFDM signals are not our choice either.

C. CCK-modulated 802.11b WiFi signals

The remaining choice is the CCK-modulated 802.11b WiFi (we call it CCK-WiFi in the following discussion for simplicity). This signal supports a throughput as high as 11 Mbps. And unlike OFDM signals, the CCK-WiFi signal focuses on the time domain. In the CCK-WiFi signal, a symbol of $\frac{8}{11}\mu s$ is equally divided into 8 time slots. And in every time

[1]The Q value of a filter is defined as: $Q = \frac{f_c}{f_B}$, where f_c, f_B are the center frequency and the bandwidth, respectively.

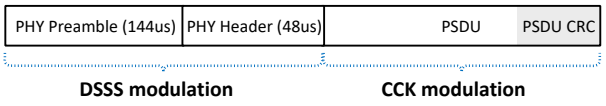


Fig. 2. The structure of CCK-modulated 802.11b packet.

slot, the signal uses modified QPSK modulation. With this excitation, the backscatter tag only needs to map bits to phase shifts according to CCK and then embed them to excitation at corresponding time slots. In this way, the tag is able to effectively use CCK-WiFi's high data-carrying capacity. The potential backscatter throughput reaches 11 Mbps, which is high enough for video streaming.

In summary, popular OFDM signals, Bluetooth signal, ZigBee signal, DSSS-WiFi signal, and LoRa signal are all unable to support a backscatter throughput of around 10 Mbps. CCK-WiFi is eventually chosen in SubScatter.

III. Sub-symbol Backscatter Modulation

A. Parallel transmissions in 802.11b

We first give a brief introduction to the physical layer model in CCK-WiFi. As shown in Fig. 2, the packet is composed of a $144\mu s$-long physical preamble used for detection and synchronization, a physical header of 48 μs containing necessary information such as data rate and the packet length, and the PSDU field protected by the cyclic redundancy check (CRC). The modulation for the preamble and header is the direct sequence spread spectrum (DSSS), while for the PSDU field it is CCK. The backscatter tag can modulate the preamble and header parts following the way introduced in SyncScatter[16], and modulate the PSDU using our sub-symbol modulation.

To generate the PSDU, there are scrambler, CCK mapping, and waveform generation in the transmitter[18]. The Scrambler is a serial convolutional encoder. It is used to whiten the input data so that the scrambled data will not be all '0' or all '1', which will cause a high peak-to-average ratio in the physical waveform. The scrambled PSDU bits are then segmented into octets, and every octet is mapped to four phase items: $\vec{\phi} = [\phi_1, \phi_2, \phi_3, \phi_4]^T$[18]. ϕ_1 is encoded by the first two bits d_1d_2 in a scrambled octet by differential quadrature phase shift keying (DQPSK). It should be noted that odd-numbered symbols shall be given an extra phase rotation of π in addition to the standard DQPSK. And the following three phase items ϕ_2, ϕ_3, ϕ_4 are quadrature phase shift keying (QPSK) encoded by the following bits in an octet d_3d_4, d_5d_6, and d_7d_8, respectively.

Those four phase items are then used to generate a $8/11\mu s$-long excitation symbol. A symbol is specifically divided into eight $11mus$-long time slots. In those time slots, the excitation phases are denoted by $\vec{\psi} = [\psi_1, \psi_2, ..., \psi_8]^T$ are generated

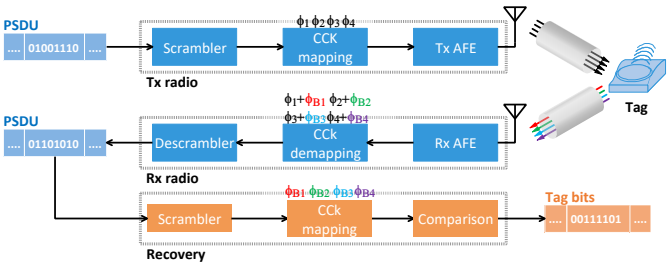


Fig. 3. The 802.11b transceiver and tag data recovery.

using four phase items as:

$$\begin{aligned}\vec{\psi} &= \mathbf{A}\vec{\phi} + \vec{r} \\ &= \sum_{i=1}^{4} \phi_i \mathbf{A}(:,i) + \vec{r}\end{aligned} \tag{1}$$

where $\mathbf{A}$ is the generator matrix and $\vec{r}$ stands for additional phases in $\vec{\psi}$. They are both defined in the WiFi standard[18] to be fixed for all WiFi devices. $\mathbf{A}(:,i)$, (i=1,2,3,4) is the i-th column of $\mathbf{A}$. Specifically, $\mathbf{A}(:,1) = [1,1,1,1,1,1,1,1]^T$, $\mathbf{A}(:,2) = [1,0,1,0,1,0,1,0]^T$, $\mathbf{A}(:,3) = [1,1,0,0,1,1,0,0]^T$, and $\mathbf{A}(:,4) = [1,1,1,1,0,0,0,0]^T$. It can be seen that the map from $\vec{\phi}$ to $\vec{\psi}$ is a linear transformation, which can be reversed as:

$$\vec{\phi} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T(\vec{\psi} - \vec{r}) \tag{2}$$

The waveform corresponding to a PSDU octet will be $\vec{S} = e^{j\vec{\psi}} = [e^{j\psi_1}, e^{j\psi_2}, ..., e^{j\psi_8}]^T$. At the receiver, $\vec{\psi}$ can be extracted and $\vec{\phi}$ can be calculated in a way similar to Equation 2. Eventually, PSDU bits can be decoded after conducting CCK-demapping and descrambling on $\vec{\phi}$.

In the waveform generation and receiving procedure described in Equation 1 and Equation 2, ϕ_1, ϕ_2, ϕ_3, and ϕ_4 can be seen as four parallel transmissions, each carries its unique data. The orthogonality between $\mathbf{A}(:,i), (i = 1,2,3,4)$ helps reduce self-interference.

B. Sub-symbol backscatter modulation

The whole working procedure of the SubScatter system is shown in Fig. 3. The transmitter and the receiver are both WiFi radios, and their functional blocks are consistent with those introduced in the previous section. The tag conveys its data by modifying the CCK-WiFi signal. But with commodity radio as the receiver, we will have no access to the received signal phases $\vec{\psi}$, nor the extracted phase items $\vec{\phi}$. Only the decoded PSDU is accessible.

Tag modulation should be carefully designed so that backscattered signals can be accepted by commodity WiFi, and tag bits can be recovered from PSDU. Following the idea of "codeword translation" in HitchHike and SyncScatter[1], [16], the phase of the symbol should be rotated for $\phi_B \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$, the backscattered signal will be $\vec{S}_B = e^{\phi_B}\vec{S} = e^{\phi_B}e^{j\vec{\psi}} = e^{j[\phi_B\mathbf{A}(:,1) + \sum_{i=1}^{4}\phi_i\mathbf{A}(:,i) + \vec{r}]}$. As a result, the recovered phase items are: $[\phi_B + \phi_1, \phi_2, \phi_3, \phi_4]$. That means only one of the four phase items can be used. The achievable

throughput is $\frac{11}{4}Mbps$. That is far below our target. In this work, tag follows CCK introduced above to modulate excitation signal. Tag octets are first mapped to backscatter phase items $\vec{\phi}_B = [\phi_{B1}, \phi_{B2}, \phi_{B3}, \phi_{B4}]^T$ using QPSK modulation. We aim to convey $\vec{\phi}_B$ to the commodity receiver using $\vec{\phi}$ as the carrier. If we replace $\vec{\phi}$ with $\vec{\phi} + \vec{\phi}_B$ in Equation 1, the received signal $\widehat{\vec{\psi}}$ becomes:

$$\begin{aligned} \widehat{\vec{\psi}} &= \mathbf{A}(\vec{\phi} + \vec{\phi}_B) + \vec{r} \\ &= \vec{\psi} + \mathbf{A}\vec{\phi}_B \end{aligned} \tag{3}$$

That means the phase of the backscattered signal should be the sum of the original excitation phase $\vec{\psi}$ and the tag phase $\mathbf{A}\vec{\phi}_B$. Then the backscattered signal will be received by the commodity receiver and $\vec{\psi} + \mathbf{A}\vec{\phi}_B$ will be extracted. The four phase items $\widehat{\vec{\phi}}$ will be extracted as the way shown in Equation 2:

$$\begin{aligned} \widehat{\vec{\phi}} &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T(\widehat{\vec{\psi}} - \vec{r}) \\ &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T(\vec{\psi} + \mathbf{A}\vec{\phi}_B - \vec{r}) \\ &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T(\mathbf{A}\vec{\phi} + \mathbf{A}\vec{\phi}_B) \\ &= \vec{\phi} + \vec{\phi}_B \end{aligned} \tag{4}$$

So $\vec{\phi}_B$, in which tag data is contained, will successfully propagate to $\widehat{\vec{\phi}}$. And $\widehat{\vec{\phi}}$ can be recovered using PSDU bits provided by WiFi radios when the excitation phase items in $\vec{\psi}$ have already been inferred by deploying another commodity WiFi radio[1] or making the excitation signal pre-defined[19]. For ease of deployment, the transmitter is made to generate packets with fixed content.

The explanation above shows how tag follows CCK to embed its bits into excitation. After backscatter modulation, the signal is still a legal CCK packet. Furthermore, the preamble and header parts that contain the necessary decoding information are reserved. On the receiver side, the detection and burst synchronization can be successfully conducted. And the frequency offset estimation and compensation on the backscattered CCK signal are expected to work well[18]. The decoding will also be successful. In summary, as the backscattered signal is still a CCK-modulated one, it fits the commodity receiver and makes decoding an easy task.

By conducting CCK-mapping and scrambling again on the decoded PSDU, we can get the phase items $\widehat{\vec{\phi}}$ of the received signal. The whole procedure is shown in Fig. 3, where the **Recovery** is conducted on the decoded PSDU from commodity radio. It is composed of scrambler, CCK-mapping, and comparison. Scrambler and CCK-mapping are identical to those in traditional WiFi. The comparison is designed to extract $\vec{\phi}_B$ by simply comparing $\widehat{\vec{\phi}}$ with $\vec{\phi}$.

In real operation, we need to phase-shift excitation signal by $\mathbf{A}\vec{\phi}_B$ to add phase items $\vec{\phi}_B$ to excitation signal as Equation 3. Specifically, the tag needs to phase-rotate the excitation signal by $\mathbf{A}\vec{\phi}_B$ at corresponding time slots in a symbol, which can be intuitively seen in Fig. 4.

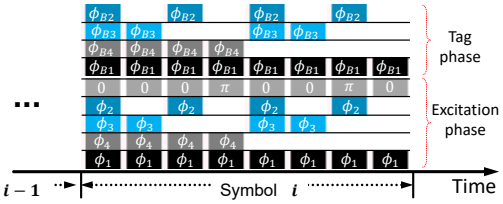


Fig. 4. The phase of the backscattered signal in eight time slots of a symbol. In every time slot, the excitation phase is the sum of the lower five rows, and the phase introduced by backscatter is the sum of the upper four rows. The blank position means the corresponding phase item is absent in the sum.

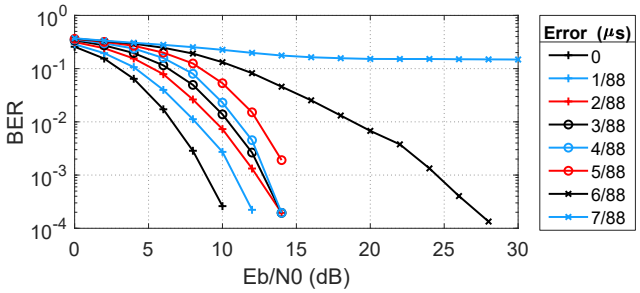


Fig. 5. The BER of tag bits with different time error.

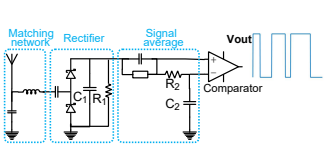
IV. Synchronization Design

We should make sure that the tag modulation is aligned to the excitation symbols, and that tag phases are added into the excitation at accurate time slots. To realize this time alignment, we first analyze the synchronization requirement and then propose a minimized-Hamming-distance based scheme that can be deployed in backscatter.

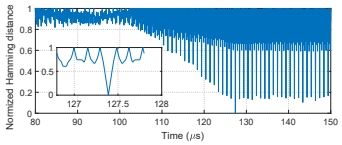
A. Synchronization requirement

We analyze the needed synchronization accuracy using the 802.11b WiFi simulation tool of Agilent advanced design system (ADS) and Matlab. We set the 802.11b WiFi end-to-end simulation model to run at 11 Mbps CCK mode. The backscatter modulation is conducted in Matlab. The data exchange between ADS and Matlab is realized by the *MATLAB script-interpreting cosimulation models* supplied by ADS. Additional time error is introduced manually to backscatter modulation and makes it misaligned to 802.11b excitation symbols in Matlab.

We give the result of modulation on ϕ_1 in Fig. 5. As can be seen that when the synchronization error is below 5/88 μs, BER is below 0.1% with E_b/N_0 greater than 15dB. Instead, when the synchronization error is 6/88 μs, we will have to increase E_b/N_0 to about 30dB to realize a similar BER. If the error is even higher, BER would not be significantly reduced by increasing E_b/N_0. Therefore, 5/88 μs accuracy is necessary. Such a requirement is more strict compared to that of 150 ns in SyncScatter[16] and the delay tolerance of over 2 μs in HitchHike[1]. That is because HitchHike uses four excitation symbols to carry one tag bit and SyncScatter modulates one-bit tag data on one excitation symbol. While

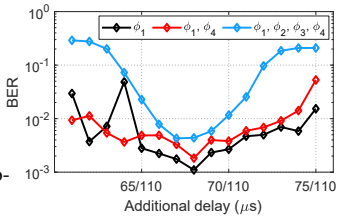


(a) The synchronization circuit.

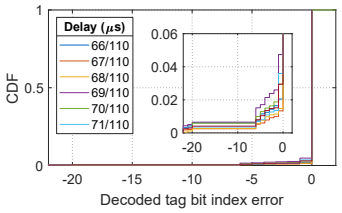


(b) The Hamming distance for synchronization.

Fig. 6. Synchronization design.



(a) BER of tag data with estimated additional time delay.



(b) Bit index error with estimated additional time delay.

Fig. 7. Synchronization performance.

SubScatter uses one symbol to carry multiple tag bits. Thus, the backscatter modulation has to be aligned more precisely to time slots in a symbol.

B. Minimized Hamming distance for synchronization

We utilize the synchronization circuit shown in Fig. 6(a). The circuit is made up of a matching circuit, a rectifier, an averaging circuit, and a comparator. It is a classic design and the effectiveness has been verified by a lot of backscatter system including WISP[20] and SyncScatter[16]. Our key design consideration focuses on the choice of the rectifier bandwidth, $f_R = \frac{1}{R_1 C_1}$, and the signal averaging circuit bandwidth, $f_A = \frac{1}{R_2 C_2}$. f_R influences the amplitude and the changing sharpness of the rectifier output[21]. When f_R is low, the amplitude will be high, but the waveform will change slowly and may lose details of the excitation envelope. On the contrary, high f_R ensures good waveform detail but with low amplitude. Considering the synchronization requirement of 5/88 μs, we choose f_R to be 25 MHz. f_A is used to limit the bandwidth of the inverting input of the comparator to provide a reference voltage to digitize the rectifier output. If it is too high, the reference will be unstable. And if it is too low, the reference voltage will rise too slowly for the comparator to work well when excitation comes. To keep this reference voltage stable and effective, f_A is chosen to be 2 MHz.

As shown in Fig. 6(a), the output of the synchronization circuit is a binary sequence, in which '1' and '0' appear only when the signal amplitude is high and low, respectively. It will be processed by the FPGA in real-time. There is a pre-stored binary template in FPGA. The template is chosen from the envelope so that when the corresponding excitation signal comes, the Hamming distance will be minimized. We take the binary envelope signal in the first $30\mu s$ of 802.11b WiFi packet as the template. We only need to compare the binary envelope sequence with the template and get their Hamming distance using a sliding window. It is worth noting that the template corresponds to a part of the 802.11b WiFi preamble and is not dependent on the PSDU at all.

Take the result of a real experiment as an example. The 802.11b packet starts at about 100 μs, but the specific time is unknown. First, the envelope signal is sensed and digitized by the tag synchronization circuit. Then, the Hamming distance between the received envelope and the template is calculated as shown in Fig. 6(b). The minimum value of Hamming distance precisely locates the time at 127.39 μs. The template is chosen as the first 30 μs in the preamble, we can infer that the packet starts at $127.39 - 30 = 97.39\mu s$. With the knowledge that the 802.11b preamble has a duration of 192 μs, we can know that the first PSDU symbol starts at about $97.39+192 = 289.39\mu s$. The accurate time of the following excitation symbols can also be located.

C. Synchronization effectiveness

The instant when Hamming distance is minimized is used as a time reference for synchronization. But there are additional delays caused by the hardware circuit and the FPGA processing. In the synchronization circuit, the matching network, the rectifier, the averaging circuit, and the comparator will cause hardware propagation delay. And the FPGA needs to calculate the Hamming distance, translate the backscatter octets to the phase shifts, and control the switch. Those tasks will take tens of clock cycles, causing the FPGA processing delay.

As it is non-trivial to directly measure those time delays, we infer them from experiments. We exhaustively search for this total delay by estimating it to be different values and compensating for it, then observing the tag data recovery performance. Fig. 7(a) shows the BER of tag data when only one phase item (ϕ_1) is modulated, when two items (ϕ_1, ϕ_4) are modulated, and when four phase items are modulated. E_b/N_0 is about 20 dB in this experiment. BER is minimized when the total delay is estimated to be 68/110 μs in all three cases. And the corresponding BERs are 0.11%, 0.18%, and 0.44%, respectively. Recall the simulation result of modulation on ϕ_1 in Fig. 5, the BER is approaching 0.1% with E_b/N_0 around 15dB only when the error is less than 5/88 μs. That proves the synchronization requirement can be satisfied.

The difference between the index of the first non-zero tag bit in expectation and that in the experiment result can also be used to evaluate synchronization performance. We define this difference as the **bit index error**. The tag is made to transmit a random binary sequence starting with a bit '1'. And to preserve the necessary fields in excitation, including the frame control field and MAC addresses fields that contain necessary information for commodity WiFi, tag data modulation should start at the 129th bit in PSDU.[2] Bit errors will make '0' and '1' change to each other, and thus the index of the first bit '1'

[2]The frame control field takes four bytes, and each of the source and destination addresses takes six bytes. So the first non-zero tag bit should appear at the index of $8 \times (4 + 6 + 6) + 1 = 129$.

will be changed. As shown in Fig. 7(b), with the estimated delay of 68/110 μs, the **bit index error** is 0 in over 99% packets. And such proportion only significantly decreases for less than 4% when the estimated delay varies between 66/110 μs and 71/110 μs. This is also evidence of the effectiveness of the synchronization design.

V. Implementation

A. 802.11b radios

Two laptops equipped with the Qualcomm Atheros AR938X NICs are used as the excitation generator and the receiver. The excitation content is controlled using the CommView for WiFi software. CommView is also used in the receiver to capture physical layer packets.

B. Tag modulation

Tag modulation is realized by controlling the toggling of the RF-switch. To frequency-shift the excitation signal by f_S, the switch needs to consecutively toggle between the open position and the closed position every half the clock cycle $\frac{1}{2f_S}$[1]. For backscatter data modulation, tag needs to add an additional phase to the excitation signal. Recall the tag modulation in Equation 3, the phase shifts in the eight time slots of a symbol are: $\mathbf{A}\vec{\phi}_B = \mathbf{A}[\phi_{B1}, \phi_{B2}, \phi_{B3}, \phi_{B4}]^T$. It is worth noting that signal phase is periodic with 2π, and $\phi_{B1}, \phi_{B2}, \phi_{B3}$, ϕ_{B4} are chosen from $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. This ensures that the phase shifts in the eight time slots are still in $\{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$. Thus, only phase rotations of $0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$ are needed in tag data modulation.

Interscatter[19] deploys four RF impedance ports for QPSK modulation in backscatter tag, one port for a phase state. When a phase is selected, the corresponding impedance port is connected to the antenna through the RF switch. We choose to translate additional time delays into phase shifts, which requires only two impedances Z_1, Z_2. The RF-switch toggles between two states periodically with the frequency of Δf. The reflected signal can be seen as multiplied by a square wave of frequency Δf. Considering two signals of the same frequency Δf, the second is delayed by half cycle $\frac{1}{\Delta f}$, then it is easy to see that there will be a phase difference of π. Similarly, phases of $\frac{\pi}{2}$ and $\frac{3\pi}{2}$ are corresponding to the time delay of $\frac{1}{4\Delta f}$ and $\frac{3}{4\Delta f}$, respectively. We can see that in this procedure Z_1, Z_2 only need to be different from each other, so that corresponding reflection coefficients are different. This requirement is very easy to satisfy. The clock signals with fixed delays can be generated in the FPGA and fed to the single-pole single-throw (SPST) RF-switch. In this way, phase shifts of $0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$ can be realized. Then they will be introduced to excitation symbols when conducting frequency shift[22].

C. Tag design

As shown in Fig. 8(a), our tag is composed of four parts: the synchronization circuit, the modulation circuit, the processing unit, and the energy harvesting unit.

The synchronization circuit extracts a binary sequence from the excitation envelope. Besides capacitors, resistors, and inductors, the circuit shown in Fig. 6(a) contains a 3dBi-glue-stick antenna, two Avago HSMS-2862 diodes, and an ON Semiconductor NCS2250 comparator.

The modulation circuit realizes sub-symbol backscatter modulation. The core component of the modulation circuit is an Analog ADG902 single-pole single-throw (SPST) RF-switch. When the switch is in the open position, its RF impedance is $Z_T = \infty$. When the switch is in the close position, it is short to GND and its impedance is $Z_T = 0$. The open and closed states of the switch correspond to the reflection coefficient of 1 and -1, respectively.

The processing unit calculates the Hamming distance between this sequence and the template for synchronization and then modifies the excitation according to the tag data by controlling the modulation circuit. It is implemented in a Microchip AGLN250 low-power FPGA. We set both the template length and the binary envelope sequence length to 1650[3]. That means when a new binary envelope sample is generated, the Hamming distance of two 1650-bit sequences is calculated. The instant in which the Hamming distance is minimized is utilized for synchronization.

The energy harvesting part supplies energy in scenarios where batteries cannot be used. The core component is a harvesting management chip, TI BQ25570, and the harvested energy is stored in a 1000 μF capacitor. When enough energy has been harvested, the power supply of 3.3V will be restored. When the voltage on the storage capacitor is too low because of discharging, the management chip will shut down the power supply and start harvesting again.

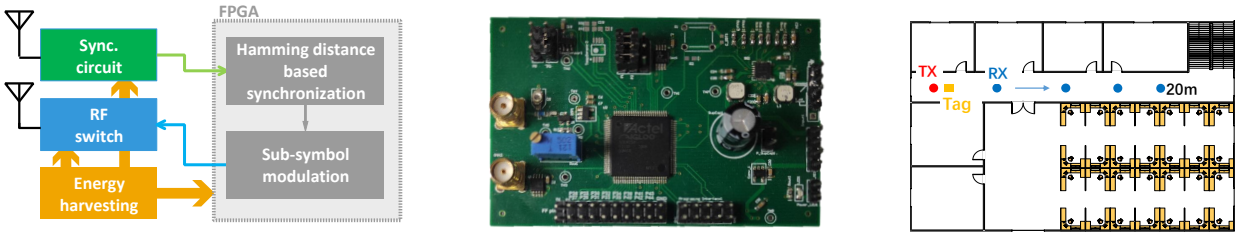
Besides the four functional parts introduced above, we also use an Analog LTC6930 oscillator to drive the FPGA and a boost converter TI TPS73615 to provide the 1.5 V power rail for the core of the low-power FPGA. The prototype is composed of COTS devices. Costs for the AGLN 250 FPGA, the BQ25570 harvesting chip, the ADG902 RF-switch, and the MP3-37 solar cell are \$26.68, \$4.75, \$3.46, \$3.21, respectively. The remaining components, including the LTC6930 oscillator, the TPS73615 boost converter, the NCS2250 comparator, the HSMS-2862 diodes, capacitors, resistors, and inductors take \$7.57 in total. The prototype cost will be less than \$45.

D. Power consumption and harvesting

The hardware prototype shown in Fig. 8(b) is built on a four-layer FR-4 substrate. As shown in Table. I, the peak power consumption is about 25.4 mW, in which the FPGA consumes 24 mW, the comparator NCS2250 in the synchronization circuit consumes 0.4 mW, the RF switch consumes about 0.1 mW, and the oscillator consumes 0.9 mW. Such power consumption is high for backscatter. We put the FPGA into the "Flash*Freeze" mode[4] when there is no excitation signal

[3]The template is chosen from $30\mu s$-long envelope, and the processing clock for synchronization is 55 MHz, so the template length is $30\mu s \times 55MHz = 1650$

[4]The "Flash*Freeze" is a low-power mode of the Microchip AGLN250 FPGA. In this mode, the FPGA stops running but resister states are kept. And the power consumption reduces to tens of micro-watts.



(a) The tag overview. (b) The tag prototype. (c) Experiment scenario.

Fig. 8. The tag is composed of a synchronization circuit, an RF switch, the FPGA processing logic, and the energy harvesting part.

TABLE I
COMPARISON OF PCB AND IC POWER CONSUMPTION (MILIWATT)

Implem.	Digital core	Osc.	Switch	Detector	Total
Prototype	24	0.9	0.1	0.4	25.4
ASIC	0.371	0.196	0.002	0.034	0.613

for power saving. And when the excitation comes, the system can be immediately waken up by the comparator output.

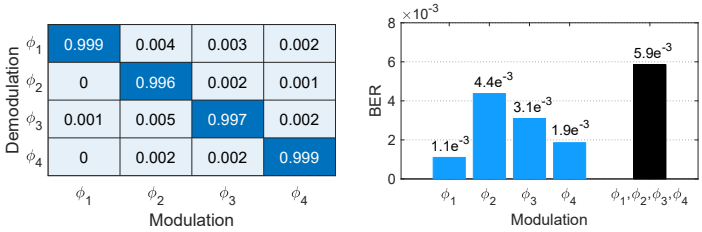
We also find that most of the power in FPGA is spent on generating the clock signal for frequency-shift and phase-modulation in the FPGA using phase-locked loop (PLL). An ASIC simulation using Cadence IC6.17 Virtuoso software and TSMC $0.18\mu m$ CMOS process design kits is conducted for power reduction. Results are shown in Table. I. A ring-oscillator that needs $196\mu W$ is used to generate clock signal of both logical operation and backscatter modulation. The powers for the RF-switch and the detector are reduced to $2\mu W$, and $34\mu W$, respectively. And in this design, the digital logic power can be reduced to about $371\mu W$ in the AGLN250 FPGA platform. So that the ASIC implementation power can be reduced to $0.613mW$. It should be noted that this is the peak power. When there is no excitation, power consumption on the digital core, oscillator, and RF-switch can be reduced, and the total power approaches tens of micro-watts. In this way, overall consumption can be reduced.

We use a solar panel to harvest light energy. Experiments show that SubScatter prototype can work well with energy harvested from office light of about 400 Lux using a 114.0x36.5 (mm) solar cell MP3-37. The prototype will need about 22 seconds to charge the storage capacitor, and then work for about 0.21 seconds. The tag is able to transmit more than 200 packets. And the prototype can work continuously when the outdoor sunlight works as the energy source. If the tag power is reduced to the ASIC level, the harvesting performance will be better. In the following sections, sufficient energy is provided to the prototype tag so that the real performance of sub-symbol modulation and synchronization design can be evaluated.

VI. EVALUATION

A. Experiment setup

The experiment is conducted in a hallway around an office area, which is shown in Fig. 8(c). Considering SubScatter's

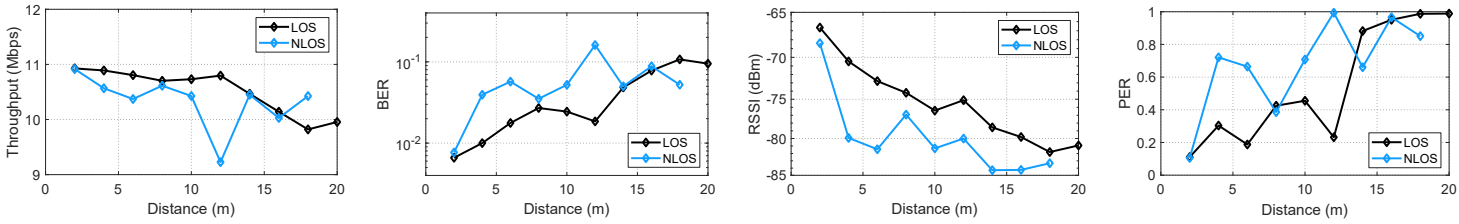


(a) The isolation between phase items. (b) The BER with different phase items.

Fig. 9. The verification of sub-symbol backscatter modulation.

ability to totally control the payload of CCK-WiFi, we want the backscattered packets to be legal and contain the correct CRC. Only in this way will they be acceptable to any commodity WiFi radio without shutting down the CRC check function as done in HitchHike[1] and SyncScatter[16]. We pay more attention to compatibility and choose to provide predefined excitation so that a single receiver is needed. In HitchHike[1], the choice is to deploy two coherent receivers to combat random excitation. But there are three drawbacks to real deployment: 1) the excitation density in time domain and the wireless channel are hard to determine in tag; 2) cooperation between two receivers is difficult to realize; 3) there is no way to keep backscattered packets accepted in WiFi radios with CRC function. Our setting trades free excitation content for the ease of real deployment and compatibility with most WiFi radios.

In our experiment, a power amplifier is utilized to boost the transmitter power to about 20dBm, and a 3dBi glue-stick antenna is used in the transmitter. The antenna embedded in the laptop is used in the receiver. The tag is placed about 0.2m from the transmitter, and the receiver is gradually moved away from the tag. The tag data recovery shown in Fig. 3 is realized in a Matlab script. Its input is the decoded PSDU of backscattered packets, which is provided by the CommView for WiFi software in the receiver. In the backscatter modulation, the tag just conducts frequency-shift on the physical preamble and header parts of the excitation so that the backscattered packets can be detected and received by the commercial radio. And the following PSDU is modulated by the introduced sub-symbol modulation to carry tag data.



(a) Throughput (b) BER (c) RSSI (d) PER

Fig. 10. Backscatter throughput, BER, RSSI and PER across distance in line-of-sight and non-line-of-sight deployments.

B. Isolation between ϕ_1, ϕ_2, ϕ_3, and ϕ_4

As introduced above, ϕ_1, ϕ_2, ϕ_3, and ϕ_4 can be treated as four parallel transmissions. It is of vital importance for us to improve the backscatter throughput to 10 Mbps. To verify such parallelism in backscatter modulation, we make the tag separately modulate each of those transmissions and keep the others unmodified. In every case, 5000 backscattered packets are analyzed. The tag data recovery result is shown in Fig. 9(a). We can see that when one of ϕ_1, ϕ_2, ϕ_3, and ϕ_4 is modulated, the corresponding phase item will be correctly recovered in over 99.5% cases, while there also exist decoding errors in other phase items. For example, when ϕ_2 is modulated by tag, ϕ_2 is correctly recovered in 99.6% cases, and 0.4% ϕ_1, 0.5% ϕ_3, 0.2% ϕ_4 are recovered to be modified by mistake. Notice that $99.6\% + 0.4\% + 0.5\% + 0.2\% \neq 1$. That's because this is not a classification or identification problem, four-phase items are processed and recovered separately. Such an error is caused by the leakage between phase items, but it occurs very rarely. The four phase items can be seen as parallel transmissions, and their isolation is good enough for the tag to concurrently use all of them. The BER when the tag modulates one or all of the four phase items is shown in Fig. 9(b). When ϕ_1, ϕ_2, ϕ_3, and ϕ_4 are separately modulated, the BERs are 0.11%, 0.44%, 0.31%, and 0.19%, respectively. Combining all four phase items increases the BER to about 0.59%. Of course, the BER can be further reduced by introducing forward error correction (FEC) code or simply repeating a tag bit several times, which is not the scope of this paper. We think such isolation is good enough and we can generate four parallel backscatter transmissions using one CCK-WiFi stream.

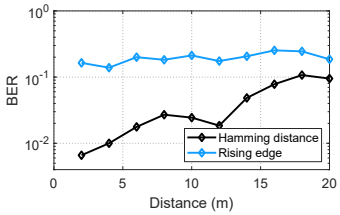
C. End-to-end results

We test end-to-end performance at different distances. Our experiment focuses on throughput, BER, received signal strength indicator (RSSI), and packet error rate (PER). We present the effective throughput in which BER is accounted for. In 802.11 WiFi, PSDU is protected by a 32-bit CRC. Its corresponding waveform is also generated using CCK modulation. So the tag can modify the CRC to any 32-bit sequence using our sub-symbol modulation. As the excitation PSDU is set to be pre-defined, the tag is aware of both the backscattered PSDU and the excitation CRC. Then the tag can first infer the CRC of backscattered PSDU and then modify the excitation CRC to fit it. So that those backscattered packets pass CRC if there are no bit errors. They are also acceptable to WiFi radios whose CRC protection is not shut down as done in the CommView software. We use PER to measure the proportion of backscattered packets with bit errors.

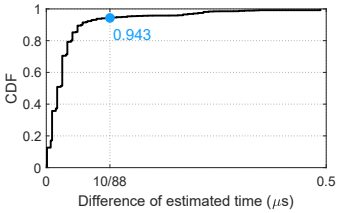
Both the line-of-sight (LOS) and non-light-of-sight (NLOS) scenarios are considered. In NLOS deployment, the tag and the receiver are separated by a wooden desk in NLOS deployment. Experiment results are shown in Fig. 10. The maximum working ranges in LOS and NLOS deployment are about 20 m and 18 m, respectively. Such distances are far enough for most indoor deployments. When the distance is less than 2 m, the LOS throughput and the NLOS throughput are both about 10.9 Mbps. Within 20m, the tag can achieve a throughput of about 10 Mbps, which is high enough to transmit high-resolution video in real-time. When the distance is closer than 2m, the BER is below 1% and the PER is about 0.1. And when the distance is below 10m, we can get a BER lower than 2.5% and a PER lower than 50%. BER and PER significantly rise when the receiver is moved away from the tag in both LOS and NLOS deployments. That's because the RSSI decreases with distance, as shown in Fig. 10(c). When RSSI is lower, the interference and the noise will have more influence and make receiving and decoding of backscattered signals even more difficult. We can also observe that in the NLOS scenario, the throughput, the RSSI, and the achievable working range are lower, and the BER and PER are higher compared with LOS deployment. That's because the desk between the tag and the receiver will reflect back a significant proportion of the backscattered signal and make the RSSI much lower, which will weaken the tag transmission capacity.

D. Hamming-distance or simply detecting signal power for Synchronization?

An important concern is whether the Hamming-distance-based synchronization can be replaced by simply detecting the power level of the incident signal, as the excitation power is expected to be stronger than noise and interference. When the excitation signal comes, there will be a rising edge in the comparator output, which can be used to indicate the beginning of a packet[1], [16]. If the answer is yes, the synchronization design can be simplified and the calculation for the Hamming distance can be avoided. In this section, we will compare two methods in terms of end-to-end performance. The circuit for Hamming-distance-based synchronization shown in Fig. 6(a) is composed of diodes, comparators, resistors, capacitors, and

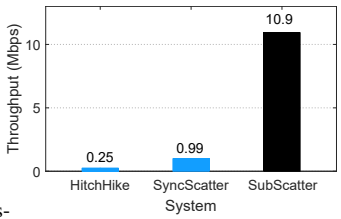


(a) BER

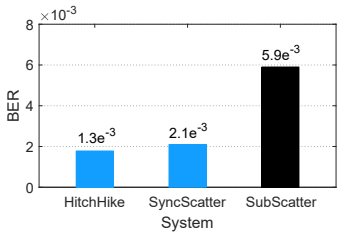


(b) The difference of estimated time using two synchronization designs.

Fig. 11. The comparison between Hamming-distance based and rising-edge based synchronization.



(a) Throughput



(b) BER

Fig. 12. The comparison of SubScatter with HitchHike and SyncScatter.

inductors. We use similar components to build the synchronization circuit to detect the rising edge of the comparator output. The difference is that a fixed voltage instead of the low-pass filtered envelope is provided to the comparator as a reference to determine high power.

The experiment scenario and deployment setup are the same as in Section. VI-C. The result is shown in Fig. 11. With the rising edge for synchronization, the BER is above 10% even when the receiver is placed only 2m from the tag. But when using the Hamming distance, the BER is below 1% at 2m. And even at 20m, the BER is still no more than 10%. Such a significant difference is related to the poor precision when the rising edge is used for synchronization. Let t_A, t_H, and t_P be the actual synchronization time, the Hamming-distance based synchronization time, and rising-edge based synchronization time, respectively. Recall that the BER rises with a synchronization error. In Fig. 7(a), the result of local traversal research using end-to-end experiments shows that the accurate synchronization instant is 68/110μs. Only when the error is below $5/88\mu s$ will the BER be below 1%. We can infer that $|t_H - t_A| \leq 5/88\mu s$. We aim to analyze $|t_P - t_A|$, but t_A is non-trivial to measure directly. So we estimate $|t_P - t_A|$ with the help of t_H:

$$\begin{aligned} |t_P - t_A| &= |(t_P - t_H) + (t_H - t_A)| \\ &\geq |t_P - t_H| - |t_H - t_A| \\ &\geq |t_P - t_H| - 5/88\mu s \end{aligned} \tag{5}$$

That means $|t_P - t_A|$ is lower bounded by $|t_P - t_H| - 5/88\mu s$. We sample $|t_P - t_H|$ in experiment and depict it in Fig .11(b). We can see that $|t_P - t_H| \geq 10/88\mu s$ in 5.7% cases, which means $|t_P - t_A| \geq 5/88\mu s$ and serious bit error will occur. Still, the given lower bound of $|t_P - t_A|$ is very loose, and the real portion when $|t_P - t_A| \geq 5/88\mu s$ is much larger.

By comparison, we can see that simply using the rising edge for synchronization will cause a significant synchronization error. That's because the transmitter radio is allowed to have a power-on ramp with a duration of no more than 2 μs at the start of a packet[18]. In this period, the transmitter power rises from 10% to 90% of its maximum power. But the specific power ramp duration is not regulated, and it's allowed to drift with time. This introduces additional errors when merely detecting rising power for synchronization. Such error has a more significant influence on SubScatter than HitchHike and SyncScatter. This is because sub-symbol backscatter modulation needs better synchronization. In conclusion, the Hamming-distance based synchronization can't be replaced by simply detecting excitation power.

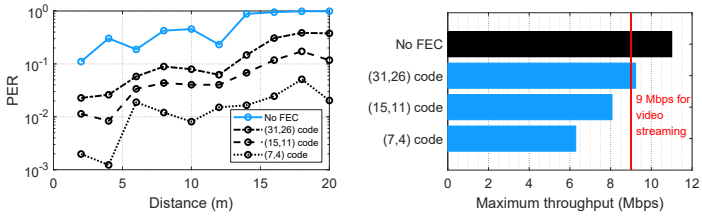
E. Comparison with state-of-the-art systems

To show the effectiveness of sub-symbol backscatter modulation and Hamming-distance-based synchronization, we compare SubScatter with the two most related systems, HitchHike[1] and SyncScatter[16]. They also utilize commodity 802.11b radios as transmitters and receivers. It's hard to fully reproduce HitchHike and SyncScatter, so we realize their backscatter modulations and synchronization designs in our platform. We use corresponding lower-bandwidth rectifier circuits and comparators to build the synchronization circuits and rely on excitation power for synchronization in those systems. For HitchHike, the tag utilizes four excitation symbols to convey one-bit tag data. And for SyncScatter, only one excitation symbol is utilized. The experimental setup is the same as that in Section VI-C, and the receiver is placed about 2 meters from the tag.

The experiment result is shown in Fig. 12. From the aspect of throughput, HitchHike and SyncScatter achieve about 0.25 Mbps and 0.99 Mbps, respectively. Benefiting from the sub-symbol backscatter modulation, SubScatter realizes a throughput of 10.9 Mbps, which is about 43× of HitchHike, and 11× of SyncScatter. The BER is shown in Fig. 12(b). In our experiment, the BERs of HitchHike, SyncScatter, and SubScatter are 0.13%, 0.21%, and 0.59%, respectively. SubScatter has higher BER, about 4.5× of HitchHike and 2.8× of SyncScatter. The high BER in SubScatter is caused by the finer-grained modulation. In SubScatter, about $\frac{1\mu s}{11}$ excitation instead of no less than $1\mu s$ in HitchHike and SyncScatter. This makes SubScatter more sensitive to noise and interference. But the BER difference is not as significant as the throughput gain. And we think it is still acceptable.

F. Effect of forward error correcting code (FEC)

To reduce the BER and PER, we investigate using forward error correction technique. The Hamming code [23] is used in the end-to-end experiments. In this code, redundancy is added to combat random errors. In the (7,4) Hamming code, four bits are encoded to seven bits and any single bit error can be corrected. Similarly, in (15,11) code and (31,26) code,



(a) PER (b) Throughput

Fig. 13. The effect of forward correction code.

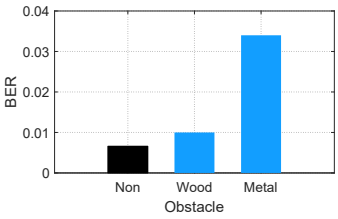


Fig. 14. Backscatter performance with obstacles.

11 and 26 original bits are encoded into 15 and 31 bits, respectively. All three Hamming codes are able to correct single-bit error. But (7,4) code has the best correction capacity, with cost of lowest efficiency. On the contrary, the (31,26) code is the most efficient, but its correction capacity is the lowest.

The PER performance with three Hamming codes is shown in Fig. 13(a). Without forward error correction, the PER exceeds 10%. While with (31,26), (15,11), and (7,4) Hamming codes, the PER drops to 2%, 1%, and 0.2%, respectively. Forward error correction effectively improves PER performance. The expense is reduced throughput. With (7,4) Hamming code, 3/7 of the bits are redundant. Thus, effective throughput drops to about 6.3 Mbps. Similarly, with (15,11) and (31,26) Hamming codes, the effective throughput is reduced to about 8.1 Mbps and 9.2 Mbps, respectively.

G. Influence of obstacle

To test SubScatter performance in more complex scenarios, we block the line between the tag and the receiver using a wooden or metal plate. The tag-receiver distance is 2 m. The BER is shown in Fig. 14. With wooden or metal obstacles, BER rises to about 1% and 3.2%, respectively. That is because obstacles, especially metal, will decrease received power and make decoding harder.

VII. Related work

Compatibility with COTS radios: The backscatter technology is first applied in RFID. And on this basis, the programable platform WISP[24], [25], [26], [20] is designed. They can interact with readers. But a specialized reader is needed in those systems to help communication. To avoid such dependence on dedicated devices, ambient backscatter[27], [28] utilizes TV transmissions as excitation to realize communication between two tags. FM backscatter[29] transmits voice information by backscattering FM radio signals. But they are still not general-purpose. Passive WiFi[7] tag transmits data to WiFi radios by modulating single-tone signal from a helper radio into WiFi packets. And then Interscatter[19] uses a COTS Bluetooth device to replace the helper radio to generate the single-tone signal. After that, the *codeword translation* introduced in HitchHike [1] brings out a lot of backscatter systems compatible with COTS radios[11], [13], [12], [30], [10], [31], [32], [15], [33], [34], [35], [36].

High throughput: Although HitchHike realizes COTS-radio-compatible backscatter, its throughput of hundreds of kilobits per second is not satisfactory. To improve the transmission data rate, SyncScatter[16], RBLE [14] and BLE-backscatter[37] push the tag modulation to a single-symbol level and improve the throughput to about 1 Mbps. LScatter[8] and TScatter[9] conduct sub-symbol backscatter modulation and further raise backscatter throughput to over 10 Mbps, but at the cost of compatibility with the COTS devices.

Codeword translation: SubScatter follows the idea of *codeword translation*[1] to transform one CCK-modulated symbol to another as many existing works[1], [11], [31], [16], [13], [12], [30], [10]. While SubScatter further digs into signal phase in the physical layer and observes the phase change caused by tag modulation more accurately, which improves the backscatter transmission capacity. Such an idea can also be used for other signals such as OFDM signals and DSSS signals to improve backscatter performance. But it doesn't contribute to realizing backscatter throughput as high as 10 Mbps, thus it is not the scope of this paper.

SubScatter in OFDM: SubScatter utilizes four parallel transmissions in CCK-WiFi for high throughput. Why cannot SubScatter be generalized into OFDM systems that have even more parallel subcarriers. In CCK-WiFi, the physical waveform is generated by multiplying four transmissions. Notice that backscatter bits are embedded by multiplying excitation with a square wave. This enables backscatter to modify transmissions without separating them. However, when OFDM subcarriers are added in the physical layer, backscatter has to separate them first and then embed different data. However, this is unachievable.

VIII. Conclusion

We present SubScatter, a novel WiFi backscatter that pushes tag modulation into sub-symbol level. We first analyze various commercial signals and choose the CCK-modulated 802.11b WiFi signal as the excitation. After that, we propose the sub-symbol backscatter modulation. Then, we analyze the synchronization requirement and design the minimized Hamming distance-based scheme. Finally, we build a prototype using commercial components. Experiment results show that SubScatter realizes a throughput of 10.9 Mbps and is compatible with commodity WiFi radios at the same time.

IX. Acknowledgement

We thank the anonymous reviewer for the helpful comments. This work was supported by NSFC Grant No. 61932017 and 61971390.

REFERENCES

[1] P. Zhang, D. Bharadia, K. Joshi, and S. Katti, "Hitchhike: Practical backscatter using commodity wifi," in *Proc. of ACM SenSys*, 2016.

[2] M. Katanbaf, A. Weinand, and V. Talla, "Simplifying backscatter deployment: Full-duplex lora backscatter," in *Proc. of USENIX NSDI*, 2021.

[3] J. Jang and F. Adib, "Underwater backscatter networking," in *Proc. of ACM SIGCOMM*, 2019.

[4] R. Ghaffarivardavagh, S. S. Afzal, O. Rodriguez, and F. Adib, "Ultra-wideband underwater backscatter via piezoelectric metamaterials," in *Proc. of ACM SIGCOMM*, 2020.

[5] W. Jiang, S. M. Kim, Z. Li, and T. He, "Achieving receiver-side cross-technology communication with cross-decoding," in *Proc. of ACM MOBICOM*, 2018.

[6] W. Jiang, Z. Li, Z. Yin, R. Liu, L. Liu, and T. He, "Cross-technology communication via phy-layer emulation," in *Proc. of ACM CoNEXT*, 2017.

[7] B. Kellogg, V. Talla, S. Gollakota, and J. R. Smith, "Passive wi-fi: Bringing low power to wi-fi transmissions," in *Proc. of USENIX NSDI*, 2016.

[8] Z. Chi, X. Liu, W. Wang, Y. Yao, and T. Zhu, "Leveraging ambient lte traffic for ubiquitous passive communication," in *Proc. of ACM SIGCOMM*, 2020.

[9] X. Liu, Z. Chi, W. Wang, Y. Yao, P. Hao, and T. Zhu, "Verification and redesign of {OFDM} backscatter," in *Proc. of USENIX NSDI*, 2021.

[10] V. Talla, M. Hessar, B. Kellogg, A. Najafi, J. R. Smith, and S. Gollakota, "Lora backscatter: Enabling the vision of ubiquitous connectivity," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, pp. 1–24, 2017.

[11] P. Zhang, C. Josephson, D. Bharadia, and S. Katti, "Freerider: Backscatter communication using commodity radios," in *Proc. of ACM CoNEXT*, 2017.

[12] J. Zhao, W. Gong, and J. Liu, "Spatial stream backscatter using commodity wifi," in *Proc. of ACM MobiSys*, 2018.

[13] ——, "X-tandem: Towards multi-hop backscatter communication with commodity wifi," in *Proc. of ACM MOBICOM*, 2018.

[14] M. Zhang, J. Zhao, S. Chen, and W. Gong, "Reliable backscatter with commodity ble," in *Proc. of IEEE INFOCOM*, 2020.

[15] M. Zhang, S. Chen, J. Zhao, and W. Gong, "Commodity-level ble backscatter," in *Proc. of ACM MobiSys*, 2021.

[16] M. Dunna, M. Meng, P.-H. Wang, C. Zhang, P. P. Mercier, and D. Bharadia, "Syncscatter: Enabling wifi like synchronization and range for wifi backscatter communication." in *Proc. of USENIX NSDI*, 2021.

[17] R. R. Mansour, "High-q tunable dielectric resonator filters," *IEEE Microwave Magazine*, vol. 10, no. 6, pp. 84–98, 2009.

[18] "Ieee standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications: Further higher data rate extension in the 2.4 ghz band," *IEEE Std 802.11g-2003*, pp. 1–104, 2003.

[19] V. Iyer, V. Talla, B. Kellogg, S. Gollakota, and J. Smith, "Inter-technology backscatter: Towards internet connectivity for implanted devices," in *Proc. of ACM SIGCOMM*, 2016.

[20] A. P. Sample, D. J. Yeager, P. S. Powledge, and J. R. Smith, "Design of a passively-powered, programmable sensing platform for uhf rfid systems," in *2007 IEEE international Conference on RFID*, 2007.

[21] A. Lozano-Nieto, *RFID design fundamentals and applications*. CRC press, 2017.

[22] P. Zhang, M. Rostami, P. Hu, and D. Ganesan, "Enabling practical backscatter communication for on-body sensors," in *Proc. of ACM SIGCOMM*, 2016.

[23] T. K. Moon, *Error correction coding: mathematical methods and algorithms*. John Wiley & Sons, 2020.

[24] A. P. Sample, D. J. Yeager, P. S. Powledge, A. V. Mamishev, and J. R. Smith, "Design of an rfid-based battery-free programmable sensing platform," *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 11, pp. 2608–2615, 2008.

[25] M. Philipose, J. R. Smith, B. Jiang, A. Mamishev, S. Roy, and K. Sundara-Rajan, "Battery-free wireless identification and sensing," *IEEE Pervasive computing*, vol. 4, no. 1, pp. 37–45, 2005.

[26] J. R. Smith, A. P. Sample, P. S. Powledge, S. Roy, and A. Mamishev, "A wirelessly-powered platform for sensing and computation," in *Proc. of ACM UbiComp*, 2006.

[27] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall, and J. R. Smith, "Ambient backscatter: Wireless communication out of thin air," in *Proc. of ACM SIGCOMM*, 2013.

[28] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith, "Turbocharging ambient backscatter communication," in *Proc. of ACM SIGCOMM*, 2014.

[29] A. Wang, V. Iyer, V. Talla, J. R. Smith, and S. Gollakota, "{FM} backscatter: Enabling connected cities and smart fabrics," in *Proc. of USENIX NSDI*, 2017.

[30] Y. Peng, L. Shangguan, Y. Hu, Y. Qian, X. Lin, X. Chen, D. Fang, and K. Jamieson, "Plora: A passive long-range data network from ambient lora transmissions," in *Proc. of ACM SIGCOMM*, 2018.

[31] W. Gong, L. Yuan, Q. Wang, and J. Zhao, "Multiprotocol backscatter for personal iot sensors," in *Proc. of ACM CoNEXT*, 2020.

[32] Y. Yang, L. Yuan, J. Zhao, and W. Gong, "Content-agnostic backscatter from thin air," in *Proc. of ACM MobiSys*, 2022.

[33] Y. Yang and W. Gong, "Universal space-time stream backscatter with ambient wifi," in *Proc. of IEEE PerCom*, 2022.

[34] L. Yuan, C. Xiong, S. Chen, and W. Gong, "Embracing self-powered wireless wearables for smart healthcare," in *Proc. of IEEE PerCom*, 2021.

[35] L. Yuan and W. Gong, "Poster:, high-throughput backscatter using commodity wifi," in *Proc. of ACM MobiSys*, 2022.

[36] Q. Wang, S. Chen, J. Zhao, and W. Gong, "Rapidrider: Efficient wifi backscatter with uncontrolled ambient signals," in *Proc. of IEEE INFOCOM*, 2021.

[37] J. F. Ensworth and M. S. Reynolds, "Ble-backscatter: Ultralow-power iot nodes compatible with bluetooth 4.0 low energy (ble) smartphones and tablets," *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 9, pp. 3360–3368, 2017.