Covert Communication with Acoustic Noise

Meng Jin, *Member, IEEE, ACM*, Yuan He, *Member, IEEE, ACM*, Yunhao Liu, *Fellow, IEEE, ACM* , Xinbing Wang, *Member, IEEE, ACM*
[1]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, P.R. China
[2]School of Software and BNRist, Tsinghua University, P.R. China
jinm@sjtu.edu.cn, heyuan@tsinghua.edu.cn, yunhao@tsinghua.edu.cn, xwang8@sjtu.edu.cn

***Abstract*—Along with the proliferation of IoT devices, people have a lot of concerns on the privacy issues brought by them. Existing solutions, employing encryption or trying to hide the communication in PHY layer, often suffer from the limited capability of IoT devices. To address this issue, we propose Rustle, an acoustic communication design which builds covert connection among IoT devices using random noise. Noise signal can be easily generated and exchanged by the widely used speakers and microphones on IoT devices. Based on a fine-grained control of signal's wave shape, Rustle generates a series of mutually uncorrelated random signals that contain "hidden patterns" to embed information. Extensive evaluations demonstrate that Rustle can achieve a lower than 1% BER while the eavesdropper's error rate on detecting the signal is higher than 80%.**

***Index Terms*—Covert communication; Acoustic; IoT security.**

I. Introduction

Internet of Things (IoT) devices are prevalent nowadays and potentially bring significant improvements to daily life. By combining rich sensors and wireless connectivity, these devices have the potential to learn and share extensive information about their users and their surrounding environment. However, along with their benefits come a potential privacy risk. First, most information that IoT devices share has major privacy implications. Second, as a result of the low cost and low power design, most IoT devices support only weak encryption or lack encryption at all. These two properties make IoT devices attractive targets for eavesdropping attack [1]. More seriously, although the information is well-encrypted, just the exposure of the communication may already leak the user's private information (e.g., his/her location) [2, 3]. So, an urgent problem now is: how to secure the communication between those weak and information-rich devices, considering that encryption is neither affordable nor effective.

A potential solution is to *hide* the communication in PHY-layer, making it *undetectable* for eavesdroppers. This is usually done by manipulating the channel that the signal propagates through. For example, friendly-jamming based methods [4–6] bury the signal under a strong artificial noise which can be removed only by the legitimate receiver. But this method cannot defend against a MIMO eavesdropper which can separate the noise and the signal with multiple antennas [7, 8]. A way to fight against MIMO eavesdropping is to perform channel randomization using moving antenna [7], which is however unavailable on most IoT devices. From the above example, and others detailed in Sec. III, we find that such signal hiding strategies will eventually create a hardware battle between the sender and the eavesdropper: the sender has to keep advancing its hardware to make sure that its ability to hide the signal is higher than the eavesdropper's ability to extract it. We can never win the battle with the low-end IoT devices.

In this paper, we propose Rustle, an acoustic communication system which can "furtively" connect IoT devices with noise-like signal. Different from existing methods which try to hide the signal with random channel, Rustle directly randomizes the signal itself. By carefully regulating the fine-grained (sample level) **wave shape** of the signal, Rustle can generate a series of noise-like and mutually uncorrelated signal sequences following certain patterns, which can be seen only by the receiver. Such "hidden patterns" is used to convey information. We term such a new form of signal modulation as *furtive modulation*.

So, what kind of signal pattern is suitable for furtive modulation? In the design of Rustle, we embed information in each noise sequence by adjusting its correlation (shape similarity) with a template sequence (a secret key). We find that due to the *weak transitivity* of the correlation between signal sequences, although a set of noise sequences all have high and the same correlation with the key, their mutual correlation can be designed to be very low, resembling a series of random noise. The receiver who knows the key can decode the signal by performing correlation, while an eavesdropper who doesn't know the key cannot even detect the signal.

Besides covert communication, furtive modulation also provides the following benefits:

- **Parallel transmission.** With the weak transitivity of correlation, one can simultaneously control the correlation between one sequence and multiple keys, enabling parallel transmission from one sender to multiple receivers.
- **High reliability.** Furtive modulation is in essence a form of spectrum spreading. So, it is resistant to jamming, fading, and interference, thus supports room-range coverage.
- **Inaudible.** Furtive modulation supports under-noise communication. By leveraging masking effects of the human auditory system, the signal can be inaudible by people.
- **Universal.** Acoustic channel is easy to build among IoT devices where speakers and microphones are widely used.

In summary, we make the following contributions:

- We propose a new form of modulation which embeds information in a series of mutually orthogonal noise sequences. We model the modulation process as a high-dimensional geometric problem and solve it with an optimization algorithm. A lightweight offline-cum-online design is proposed to make this process applicable on IoT devices.

- We address two challenges to make the modulated signal immune to the affect of wireless channel: i) accurate estimation of the receiver's channel without a known preamble, and ii) appropriate selection of channel resistant strategy without knowing the eavesdropper's channel.
- We perform theoretical and experimental evaluation of Rustle. The results show that Rustle achieves secret and efficient transmission under a full range of configurations, including different ambient noise, different transmission range and angle, and through-wall transmission.

Compared with the published IPSN version [9], we add a theoretical analysis on how Rustle's transmission mode (e.g., signal's volume and its correlation with the key) affects its transmission reliability and secrecy in Sec. VII-A, based on which we in Sec. VI-B propose an on-demand channel resistant strategy, where the sender's transmission mode is adaptively adjusted to achieve a customized requirement on reliability and security. Experimental results in Sec. IX demonstrate Rustle's quick adaptation for varying requirement. In Sec. VII, we add experiments to analyze the distribution of error bits and explain the error types clearly. According to the error characteristic, we introduce a new error correction mechanism in Sec. VII, which is designed to alleviate bursty acoustic interference. The evaluation results demonstrate the effectiveness of this method in Sec. IX. We also add discussions and analysis on Rustle's resistant to different attacks in Sec. VIII, showing that Rustle is resistant to eavesdropping, Jamming, side channel attack, and Brute Force attack.

II. Threat model

In our target scenario, the attacker is a malicious receiver (EVE) which tries to eavesdrop on the communication between legitimate transceivers (Alice and Bob). Rustle's target is to ensure reliable communications between Alice and Bob, making the signal received at EVE resembles noise.

We do not assume Bob's any advantages over EVE on either prior knowledge, hardware, algorithm, or channel quality. Specifically, we assume that EVE has complete knowledge of both the communication medium (i.e., acoustic) and the modulation and demodulation method used by the legitimate transceivers. We also assume that EVE is a powerful device, which can: i) eavesdrop on any communication between legitimate transceivers (attacker can also be a MIMO eavesdropper); ii) record any signal in high fidelity; and iii) process the signal with any signal processing techniques to detect the information therein. Besides, different from traditional covert communication, Rustle do not assume that the receiver's channel quality is better than the attacker's.

Besides eavesdropping attack, Rustle is also resistant to side channel attack, jamming attack, and Brute Force attack, which we will detail in Sec. VIII.

III. Is signal hiding effective?

In a wireless system, the sender embeds its information (bits) into signal by varying properties of the signal (e.g., its amplitude, phase, and frequency). This process is called modulation. To embed information, any modulated signal inevitably leaves structured hints on the channel. Fig. 1 shows the spectrum of the signal modulated with conventional modulation techniques. Clearly, the modulated signal is easy to detect by an eavesdropper.

To hide the hints, one method is to leverage the channel that the signal propagates through. Specifically, suppose the signal transmitted by the sender is x. After propagating through the channel, the signal received by the receiver is:

$$y = h \cdot x + W, \tag{1}$$

where h is the signal attenuation introduced by the channel and W is the ambient noise. To ensure that the eavesdropper cannot detect x, we can use h and W to hide any pattern in x, and make the signal on the air, y, looks like random noise. There are three typical ways in achieving this:

LPD communication. In low probability of detection communication, the transmitted information is encoded with redundancy codes and transmitted with a power that is lower than the noise floor. By assuming that the legitimate channel is advantageous over the attacking channel, the sender can always find an appropriate configuration of the transmission power and the coding redundancy that makes the signal decodable only for the legitimate receiver. However, this assumption is not always hold in IoT scenarios where the eavesdropper can use more advanced receiving systems than IoT devices.

Friendly jamming. Jamming-based systems let the receiver transmit a noise while receiving signals [4, 6]. So, nobody else can decode the signal except the receiver who knows the noise sequence. However, this method cannot defend a MIMO eavesdropper, which can separate n concurrent signals with n antennas [7, 8]. A way to defend MIMO eavesdropping is to generate noises with a MIMO system that has more antennas than that on the eavesdropper. But it is clearly infeasible on the low cost IoT devices.

Channel randomization. Another way to defend MIMO eavesdropping is to randomize the wireless channel h [7]. Consider that the channel is highly sensitive to the motion of the antenna, we can equip the sender with an rotating antenna. Then the channel h can be randomized by rotating the antenna randomly. However, this method also requires specialized hardware that is unavailable on most IoT devices.

Summary. The fundamental reason that makes signal hiding infeasible is that a modulated signal is usually highly *structured*. Such non-Gaussianity of a modulated signal makes it easily separated from the noise with either advanced signal processing techniques (e.g., ICA) [10] or advanced hardware (e.g., an antenna array). To defense advanced eavesdroppers, the sender has to use more advanced hardware (e.g., a larger antenna array or a rotating antenna) to randomize the channel [7]. So, eventually we face a hardware game between the eavesdropper and the sender. We can never win the game with the low-cost IoT devices.

IV. Random noise communication

To change the hardware game, we do not try to hide the signal under randomized channel h. Instead, we directly randomize the signal x. But how to embed information in random

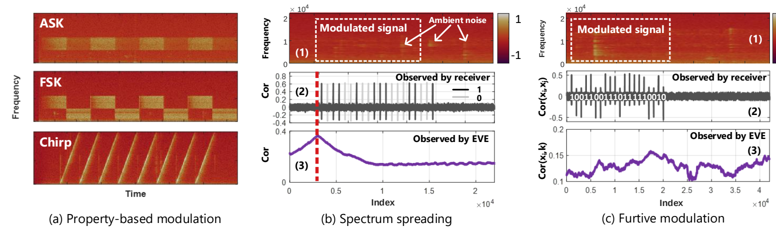


Fig. 1: Feasibility of shape-based modulation.

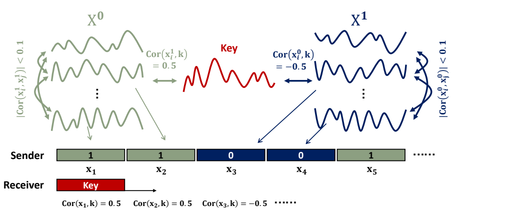


Fig. 2: Furtive modulation.

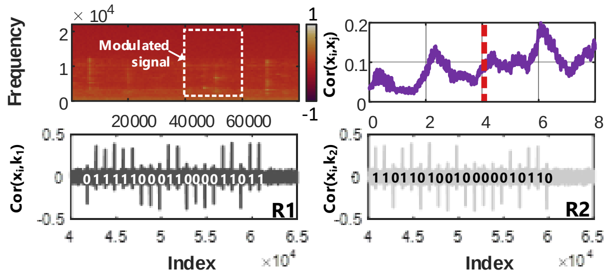


Fig. 3: Parallel transmission.

signal? The idea is to control signal's specific **waveform shape** rather than its property. Specifically, we use noises to represent signal symbols. The shape of the sequence is controlled to carry information while resembles a random noise for the eavesdroppers. Compared with property-based modulations, shape-based modulation provides two advantages:

- It enables more *fine-grained* (sample level) control of the signal, making it possible to design more complex and unstructured signal pattern to secretly embed information.
- Different from randomizing the channel h, randomizing signal x requires no additional hardware.

Now, the missing piece is a way to secretly embed information in the shape of noise signal.

A. A naive solution: spectrum spreading

A naive solution for shape-based modulation is to borrow the idea from spectrum spreading. For example, we can just use two different noise sequences x_0 and x_1 to represent bits 0 and 1. Then the receiver who knows x_0 and x_1 can retrieve the information by performing cross correlation.

We perform an experiment to see the feasibility of this method. Specifically, we transmit a set of such spreading sequences from a Samsung S7 to a Microsoft surface laptop. The signal volume is controlled lower than the ambient noise to make the signal inaudible to human ear. Fig. 1(a-1) shows the spectrogram of the received signal, where we do not find any structure hints that can be identified as modulated signal. Fig. 1(a-2) shows the correlation result, telling that the receiver can correctly retrieve the contained information.

However, there is a problem in this method: *the sender always encode a bit to the same sequence, which leaves repeated signal segments on the channel.* So, an eavesdropper can easily detect the communication by performing *mutual correlation* on the received signal. Specifically, the eavesdropper can search the received signal with a moving window which covers multiple signal sequences. In each window, it calculates the mutual correlation between each two sequences. A high mutual correlation in a window indicates the occurrence of modulated signals. Fig. 1(a-3) shows the mutual correlation result of the signal in Fig. 1(a-1), where we can see a obvious peak which indicates the modulated signal.

B. Overview of furtive modulation

We solve the above problem with a new form of modulation, named furtive modulation. Different from the spectrum spreading based method, **furtive modulation maps same bit(s) to different and uncorrelated noise sequences.** Sequences representing the same bit are designed to share a common property that is unseen by an eavesdropper. Such property is used to secretly embed bit data.

Fig. 2 shows the idea to generate such signal sequences. Specifically, the sender will generate two sets of noises sequences, $\mathbf{X^0} = \{x_1^0, ..., x_N^0\}$ and $\mathbf{X^1} = \{x_1^1, ..., x_N^1\}$, representing bits "0" and "1", respectively. Each sequence $x_i^\bullet$ ($\bullet$ = 0 or 1) embeds data in its *correlation* with a template sequence (a key) k. We term it as the signal's *key correlation*, denoted as $cor(x_i^\bullet, k)$. Sequences in the same set are designed to have the same correlation to k, meanwhile have low mutual correlation. For example, we can use $cor(x_i^1, k) = 0.5$ and $cor(x_i^0, k) = -0.5$ to represent bits "1" and "0", respectively, while control the mutual correlation between each two sequences $x_i^\bullet$ and $x_j^\bullet$ as $|cor(x_i^\bullet, x_j^\bullet)| < 0.1$. If the sender can generate a large quantity of such sequences, it can always encode messages into series of nearly **uncorrelated** noise sequences $\{x_1, x_2, ...\}$ (where $x_i \in \mathbf{X^0}$ or $\mathbf{X^1}$).

The above idea is proposed based on the *weak transitivity* of correlation coefficient [11], described as follows:

Observation 1. *Two signal sequence x_i and x_j, which both exhibit high correlation with a third sequence k, can be nearly uncorrelated with each other.*

Furtive modulation can be seen as a PHY-layer encryption. The receiver who knows the key can decode the signal by performing correlation. While for an eavesdropper who does not know the key, the signal resembles random noises. Design details of furtive modulation is introduced in Sec. V.

Figs. 1(b-1) and (b-2) show the spectrogram and the mutual correlation of a series of furtive modulation signal, where the key correlation is controlled at ± 0.7. As can be seen, the signal leaves no obvious hints on both results. Fig. 1(b-3) shows the key correlation result, where the obvious signal peaks indicates the embedded bits. Compared with the spectrum spreading based method, furtive modulation can significantly increase the secrecy, without sacrificing the reliability.

Besides covert communication, we can also use the weak

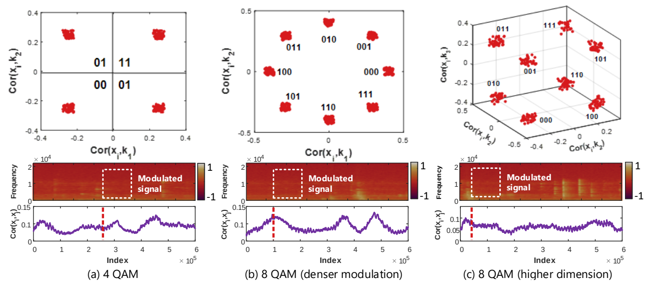


Fig. 4: Higher order modulation.

transitivity of correlation to improve modulation density or to achieve parallel transmission.

Observation 2. *Given M orthogonal keys $\mathbf{K} = \{k_1, ..., k_M\}$, one can simultaneously control the correlation between a sequence x_i and the M keys.*

Parallel transmission. In furtive modulation, one key can be considered as an identity of one channel related to one transceiver pair. So, by simultaneously controlling the correlation between one sequence and multiple keys, one can embed information for multiple receivers in that sequence. Different receivers can obtain their respective bitstreams from the same signal sequence by correlating the signal with their own keys. Fig. 3 shows a case with two parallel transmissions, where the key correlation $cor(x_i, k_1)$ and $cor(x_i, k_2)$ are controlled at ± 0.4 and ± 0.3, respectively, and the mutual correlation $cor(x_i, x_j)$ is controlled at ± 0.15. We can see that both the two receivers can correctly extract their own bits from the same modulated signal.

Higher order modulation. Besides parallel transmission, a simultaneous control of key correlation enables higher-order signal modulation. For example, if we use two keys k_1 and k_2 to encrypt one channel, simultaneous controlling of $cor(x_i, k_1)$ and $cor(x_i, k_2)$ resembles a 4 QAM modulation (as shown in Fig. 4(a)). We can further achieve higher-order modulation by either increasing the modulation dimension (e.g., using more keys as shown in Fig. 4(c)) or using more levels of key correlations (as shown in Fig. 4(b)).

V. Furtive Modulation

Problem define. We in this section address the key challenge in furtive modulation: given a key set $\mathbf{K} = \{k_1, ..., k_M\}$, how to find a set of sequences $\mathbf{X} = \{x_1, ..., x_N\}$ that satisfies:

1) The correlation between each sequence x_i and each key k_m is controlled as $|cor(x_i, k_m)| = c_{k_m}$.
2) The mutual correlation between each pair of sequences x_i and x_j is controlled as $|cor(x_i, x_j)| \leq c_{mu}$.

In the following, we first model the above problem as a *high-dimensional geometric problem* and then solve it with an optimization algorithm. At last, we discuss the secrecy and reliability of the furtive modulation method.

A. Modeling the correlation

We start from the case with only one key. In furtive modulation, each bit is represented by a noise sequence $x_i = \{x_i(1), ..., x_i(L)\}$, which in essence can be modeled

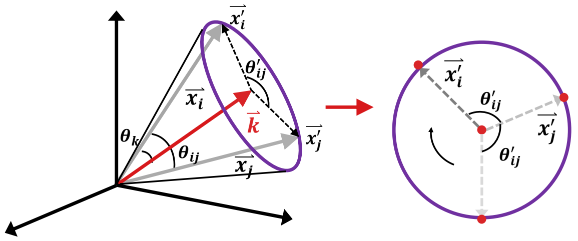


Fig. 5: One-key modulation.

as a vector in a L-dimensional space. The correlation between two sequences x_i and x_j is defined as the cosine of the angle between the corresponding two vectors $\vec{x_i}$ and $\vec{x_j}$ as:

$$cor(x_i, x_j) = \frac{\vec{x_i} \cdot \vec{x_j}}{||x_i|| \cdot ||x_j||} = \frac{\sum_{l=1}^{L} x_i(l) \cdot x_j(l)}{||x_i|| \cdot ||x_j||} \tag{2}$$

Since furtive modulation requires the key correlation of each sequence x_i to be controlled at $cor(x_i, k) = c_k$, so all the signal vectors $\vec{x_i}$ have to diverge from the key vector $\vec{k}$ with the same angle $\theta_k = \arccos(c_k)$. Fig. 5 shows an example in a 3D space (i.e., when $L = 3$). We can see that this constraint confines all the signal vectors on a conical surface around $\vec{k}$. The field angle of the cone is $2\theta_k$.

To further control the mutual correlation between signal sequences as $|cor(x_i, x_j)| \leq c_{mu}$, we should select a set of vectors on the conical surface, making their pairwise angles θ_{ij} larger than $\theta_{mu} = arccos(c_{mu})$. Without loss of generality, we assume all the signal vectors are of equal length, then their end points will form a 2D circle, as shown in Fig. 5. We denote the projections of x_i and x_j on that 2D space as x_i' and x_j'. Then, given certain c_{mu} and c_k, the problem of finding the signal set $\mathbf{X}$ in a 3D space that satisfy $\theta_{ij} > \theta_{mu}$ can be transformed to the problem of finding a set of vectors $\mathbf{X'}$ on that circle (a 2D space) where the angle between any two vectors x_i' and x_j' is larger than a certain angle θ_{mu}'. We have $\theta_{mu}' = \arccos(\frac{c_{mu} - c_k^2}{1 - c_k^2})$ based on the law of cosines.

When extended to a L-dimensional space, furtive modulation can be modeled as a $(L-1)$-dimensional *spherical code* problem [12]: finding a set of vectors $\mathbf{X'} = \{x_1', ..., x_N'\}$ pointing from the origin to a (L-1)-dimensional hypersphere, where the angle between any two vectors x_i' and x_j' satisfies:

$$\theta_{ij}' > \theta_{mu}' = \arccos(\frac{c_{mu} - c_k^2}{1 - c_k^2}) \tag{3}$$

Given certain c_k and c_{mu}, the number of sequences that satisfy the above requirement will increase with the dimension L [12]. The proof of Eq. (3) can be found in [9].

Then we discuss the case with M orthogonal keys. In this case, the furtive modulation problem can be modeled as a $(L-M)$-dimensional spherical code problem. Again, we use the 3D-space example to illustrate this problem, as shown in Fig. 6(a). When there are two keys, the signal vectors with controlled correlations to the two keys will respectively form two cones. The end points of the vectors form two circles. Then, to find the vectors which have controlled correlation with both the two keys, we should intersect those two circles, and the intersections give the candidates of the vectors. Such

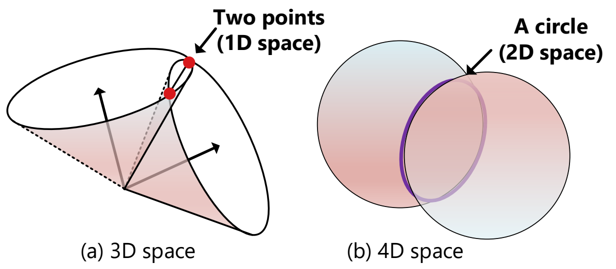


Fig. 6: M-key modulation.

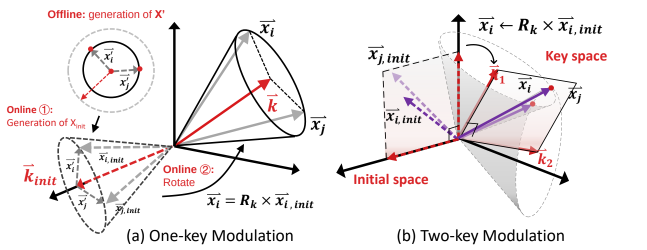


Fig. 7: The offline-cum-online $\mathbf{X}$ generation.

intersections exist as long as the field angles of the two cones θ_{k_1} and θ_{k_2} satisfy $\theta_{k_1} + \theta_{k_2} > \pi/2$. Obviously, in this case we can at most find two candidate vectors and their mutual correlation cannot be controlled.

When extended to a 4D space, the end points of the candidate vectors are given by the intersections of two 3D spheres, which is a 2D circle, as shown in Fig. 6(b). Now the furtive modulation problem is again transformed to a 2D spherical code problem. By that analogy, a M-key furtive modulation problem can be transformed to a $(L-M)$-dimensional spherical code problem as:

Finding a set of vectors $\mathbf{X}' = \{x'_1, ..., x'_N\}$ pointing from the origin to a (L-M)-dimensional hypersphere, where the angle between any two vectors satisfies:

$$\theta'_{ij} > \theta'_{mu} = \arccos\left(\frac{c_{mu} - \sum_{m=1}^{M} c_{k_m}^2}{1 - \sum_{m=1}^{M} c_{k_m}^2}\right) \tag{4}$$

The proof of the above statement can be found in [9].

B. Signal generation

The above section tells that to generate the signal set $\mathbf{X}$, we should first find a set of spherical codes $\mathbf{X}' = \{x'_1, ..., x'_N\}$ with a required minimum pairwise angle θ'_{mu}. However, a solution to the spherical code problem with a determined θ'_{mu} is still a unsolved problem in mathematics [12]. So, in this section, we find the code set $\mathbf{X}'$ in a best-effort manner: finding N spherical codes in a $(L-M)$-dimensional space where the minimal pairwise angle of the vectors is maximized. We will show in Sec. V-D that such an approximation method is sufficient to guarantee transmission secrecy.

Specifically, the task of the above searching algorithm is to find N unit vectors (the spherical codes $\mathbf{X}'$) in a $(L-M)$-dimensional space with maximum minimal pairwise angle. To do so, the basic idea is to start with a random set of unit vectors $\mathbf{X_0}$, and then iteratively splay the vector pairs with small pairwise angels. Specifically, in each iteration, we find the two vectors x'_i and x'_j with the smallest pairwise angle in $\mathbf{X_0}$. Then we expand their pairwise angle by updating these two vectors as:

$$x'_i = norm(x'_i - \varepsilon x'_j),\ x'_j = norm(x'_j - \varepsilon x'_i)$$

where ε is an adjustable parameter which controls the searching rate. Iterating this step for new vectors with smallest pairwise angle will stabilize the signal set $\mathbf{X}'$ for the maximum minimal pairwise angle. More details of the searching algorithm are omitted in this paper. Readers are suggested to refer to [12].

While the above searching process is conceptually simple, it is still difficult to perform it online, especially for the low-end IoT devices. So we perform signal generation in a hybrid *offline-cum-online* manner. In the offline phase, we generate a library of spherical codes $\mathbf{X}' = \{x'_1, ..., x'_N\}$ in a $(L-M)$-dimensional space using the above searching algorithm. The generated $\mathbf{X}'$ is stored on IoT devices. At runtime, for different key sets $\mathbf{K} = \{k_1, ..., k_M\}$ and the required key correlations, we can immediately generate the corresponding $\mathbf{X}$ from $\mathbf{X}'$ based on a transformation function determined by $\mathbf{K}$. An example of this process is shown in Figure 7.

In the one-key modulation case (Fig. 7(a)), we first generate an initial set of signal $\mathbf{X_{init}}$ with an initial key k_{init}. Specifically, k_{init} is set as an arbitrary unit coordinate vector. We scale the length of vectors $\mathbf{X}'$ to $(1-c_k^2)/c_k^2$ and add them to k_{init} to generate $\mathbf{X_{init}}$. Then we calculate the transfer matrix R_k between $\vec{k}_{init}$ and $\vec{k}$, and rotate the vector set $\mathbf{X_{init}}$ with R_k to generate $\mathbf{X}$ [13].

When uisng M keys, we first generates an initial signal set $\mathbf{X_{init}}$ with an initial key set $\mathbf{K_{init}}$ where each key is a unit coordinate vector. We term the M-dimensional space determined by $\mathbf{K_{init}}$ as the initial space and the space determined by $\mathbf{K}$ as the key space. Then we calculate the transfer matrix R_k between the initial space and the key space [13] and rotate the vector set $\mathbf{X_{init}}$ by R_k to generate $\mathbf{X}$. As an example, Figure 7(b) shows a two-key modulation case in 3D space, where the initial space and the key space are 2D surfaces. Note that $\mathbf{X}$ is generated only once for each $\mathbf{K}$. Signals used for each transmission are selected randomly from $\mathbf{X}$.

Now we have the sequence set $\mathbf{X}$, where all the sequences have positive correlations to the keys. In the transmission process, the transmitter embeds information in these sequences by flipping the sign of their key correlations. For each sequence x_i, to flip its key correlation $cor(x_i, k_m)$ from c_{k_m} to $-c_{k_m}$, we just need to rotate the vector $\vec{x_i}$ by $2 \cdot \arccos(c_{k_m})$ around the vertical of the surface determined by $\vec{k_m}$ and $\vec{x_i}$. Note that this process do not change either x_i's correlation with other keys or signals' absolute mutual correlation.

Computation overhead. Due to the offline-cum-online design, the computation overhead of $\mathbf{X}$ generation lies mainly on the vector rotation process, whose computation complexity is $O(L^2)$. Note that $\mathbf{X}$ generation is performed *only once* for each pair of transceivers, after the key set $\mathbf{K}$ between them is determined. In the transmission process, the sender encodes the bits by simply selecting signal sequences randomly from $\mathbf{X}$, whose computation complexity is $O(L)$.

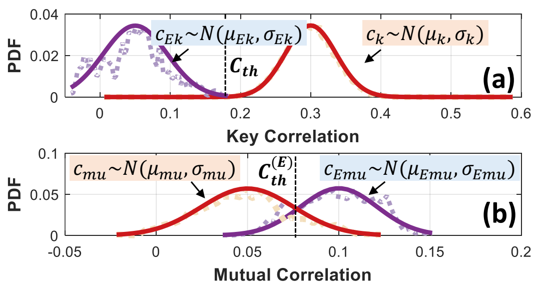


Fig. 8: Distributions of the key correlation and mutual correlation of the modulated signals.

C. Key agreement

For furtive modulation, Rustle requires the transceivers to share the key set K. Such a key agreement process should also be undetectable by the attackers. In the design of Rustle, we use the key agreement method proposed in an existing work [14], which can generate a symmetric key between two devices without relying on any communication between them. So, the key generation process is undetectable by the attacker. Note that the key can be also updated periodically to ensure that the key cannot be learned over a long period.

D. Secrecy and reliability

The secrecy and reliability of furtive modulation are sensitive to c_k and c_{mu}. To understand how c_k and c_{mu} affect the transmission reliability and secrecy, we perform an experiment where the sender generates $N = 500$ modulated signals with the key correlation of $c_k = 0.4$. A receiver locates 2m away from the sender receives the signals and calculates their key correlations and mutual correlations. The PDF of the key correlation and mutual correlation of the received signal are shown in Fig. 8(a). As can be seen, due to the channel effect, the key correlation of the received signal is attenuated to around 0.3. As a comparison, we also collect a sequence of ambient noise in a cafe. The mutual correlation and key correlation of the ambient noise (denoted as $c_k^{(E)}$ and $c_{mu}^{(E)}$, respectively) are plotted in Fig. 8(b). To make sure that the receiver can reliably decode the signal while the eavesdropper cannot detect the signal, the signal's key correlation should be higher than that of the ambient noise, meanwhile its mutual correlation should be no higher than the ambient noise.

We assume that the key correlation and mutual correlation of both the received signal and the ambient noise follow Gaussian distribution, i.e., $c_k \sim N(\mu_k, \sigma_k^2)$, $c_{mu} \sim N(\mu_{mu}, \sigma_{mu}^2)$, $c_k^{(E)} \sim N(\mu_{Ek}, \sigma_{Ek}^2)$, and $c_{mu}^{(E)} \sim N(\mu_{Emu}, \sigma_{Emu}^2)$. Then, if we set the decoding threshold as C_{th}, (i.e, the signal satisfying $|c_k| > C_{th}$ is identified as modulated signal), the receiver's misdetection rate PM_R (the probability that a modulated signal is identified as a noise) and the false alarm rate PF_R (the probability that a noise is identified as a modulated signal) in detecting the modulated signal are estimated as:

$$PM_R = \int_{-\infty}^{C_{th}} f_k(x)dx = 1 - \frac{1}{2}erfc(\frac{C_{th} - \mu_k}{\sqrt{2}\sigma_k})$$
$$PF_R = \int_{C_{th}}^{\infty} f_{Ek}(x)dx = 1 - \frac{1}{2}erfc(\frac{\mu_{Ek} - C_{th}}{\sqrt{2}\sigma_{Ek}}) \quad (5)$$

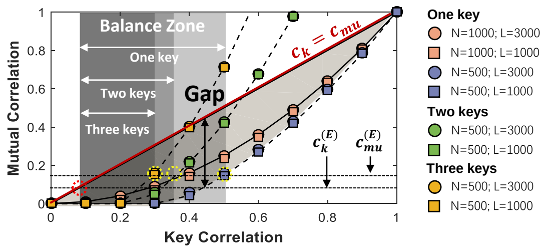


Fig. 9: c_k v.s. c_{mu}.

where $f_k(x)$ and $f_{Ek}(x)$ are the probability density of c_k and $c_k^{(E)}$, respectively. Then the decoding error is calculated as:

$$PE_R = PE_R + PF_R$$

For the eavesdropper, if it uses a threshold $C_{th}^{(E)}$ to detect the communication (i.e, the signal satisfying $|c_{mu}| > C_{th}^{(E)}$ is identified as modulated signal), its misdetection rate PM_E and the false alarm rate PF_E are estimated as:

$$PM_E = \int_{-\infty}^{C_{th}^{(E)}} f_{mu}(x)dx = 1 - \frac{1}{2}erfc(\frac{C_{th}^{(E)} - \mu_{mu}}{\sqrt{2}\sigma_{mu}})$$
$$PF_E = \int_{C_{th}^{(E)}}^{\infty} f_{Emu}(x)dx = 1 - \frac{1}{2}erfc(\frac{\mu_{Emu} - C_{th}^{(E)}}{\sqrt{2}\sigma_{Emu}}) \quad (6)$$

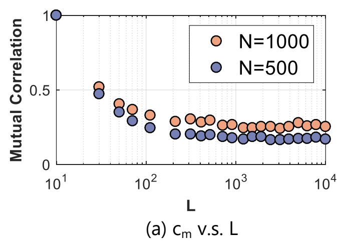
where $f_{mu}(x)$ and $f_{Emu}(x)$ are the probability density of c_{mu} and $c_{mu}^{(E)}$, respectively. The detection error is calculated as:

$$PE_E = PM_E + PF_E$$

.

Equations 5 and 6 tell that to achieve simultaneous secrecy and reliability, we need both high c_k and low c_{mu}, which are however two contradictory requirements. Recall that in the signal generation process shown in Fig. 6, c_k determines the size of the space (e.g., the field angle of the cone in a 3D space) that confines the N signal vectors. A higher c_k confines the vectors in a smaller space, which inevitably leads to a smaller pairwise angle and thus a higher c_{mu}. In the case with M keys, the confined space will be further compressed to the intersection of the M spaces determined by the M keys, as shown in Fig. 6. Besides, N and L, which respectively determines the number of vectors and the dimension of the confined space also affect the relationship between c_k and c_{mu}. In the following of this section, we generate sequence sets with different configurations of those parameters (i.e., c_k, c_{mu}, L, and N), and observe how these parameters affect the secrecy and reliability of furtive modulation.

$\mathbf{c_k}$ v.s. $\mathbf{c_{mu}}$. Fig. 9 shows c_{mu} of the signals modulated with different c_k, under different L and N. As can be seen, *there is always a gap between c_{mu} and c_k, leveraging which we can achieve simultaneous transmission reliability and secrecy*. Recall that, to achieve both reliability and secrecy, we should ensure that the key correlation of the modulated signal is higher than that of the ambient noise while the mutual correlation is lower than that of the ambient noise. Fig. 9 shows the averaged mutual correlation and key correlation of ambient noises obtained in different environments (e.g., cafe, resident,



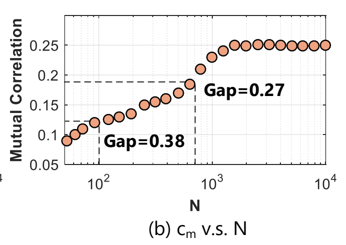


Fig. 10: Impact of different factors on c_{mu}.

office, cabin, and etc.). As can be seen, we can achieve simultaneous secrecy and reliability when $0.1 < c_k < 0.5$. We term such a range as the **balance zone** of c_k.

We also show c_{mu} v.s. c_k in the two-key and three-key cases in Fig. 9. We can see that although the gap between c_k and c_{mu} is narrowed in these cases, there still exists balance zone for secret and reliable transmission.

Impact of L. Fig. 10(a) shows the achieved c_{mu} under different sequence lengths L when $c_k = 0.5$. As expected, c_{mu} decreases with a increased L and converges when $L > 500$. On another hand, the experiment result in Sec. IX further shows that a longer sequence also introduces high transmission reliability. So we empirically set $L = 1000$.

Impact of N. Fig. 10(b) shows how c_{mu} changes with different library size N when $c_k = 0.5$ and $L = 1000$. Intuitively, we can achieve a lower c_{mu} by decreasing the library size. However, a smaller library size compels the sender to encode data with repeated signal sequence within a short period of time. This allows the eavesdropper to detect the modulated signal by performing mutual correlation. So, we set $N = 500$ in the implementation of Rustle, which can thwart the eavesdropper from learning the sequence set $\mathbf{X}$.

Specifically, when $N = 500$, if the sender transmit 20-bit packets with a period of 1s (a typical packet period for IoT devices), the eavesdropper has to capture repeated signal sequences with a longer than $N/20 = 25$ sec window. However, even a 25s signal can compel the eavesdropper to cost a computation overhead of $L^2 \cdot \binom{25\times f/L}{2} = 10^{12}$ to learn the sequence set $\mathbf{X}$ from it (where $f = 48KHz$ is the sampling rate of the microphone), which is computationally intractable and inefficient for most devices.

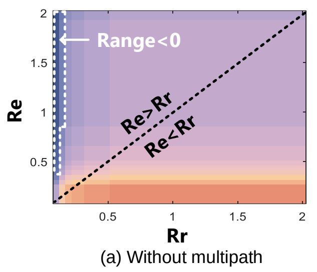
VI. Contend with channel effect

In this section, we evaluate how channel effects (e.g., path loss, noise, and multipath) affect the transmission secrecy and reliability of Rustle and introduce how we contend with them.

A. Understanding the acoustic channel

In an indoor environment, signals transmitted from a sender are usually reflected by multiple objects, which cause them to traverse different, say P, propagation paths before reconvening at the receiver, known as the multipath effect. Each path p will cause an attenuation in signal strength, denoted as $\alpha(p)$. So, the received signal can be expressed as:

$$y(t) = \sum_{p=1}^{P} \alpha(p) \cdot x(t - d_p) + w(t) \tag{7}$$



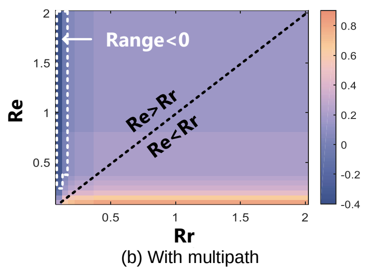


Fig. 11: Range of the zone under different R_e and R_r.

where $w(t)$ is the ambient noise, and d_p is time delay of the p^{th} path. Applying Eq (7) to Eq. (2) obtains the mutual correlation and key correlation of the received signal y_i:

$$cor(y_i, k) = \frac{\alpha(1) \cdot c_k}{\sqrt{\sum_{p=1}^{P} \alpha^2(p) + \alpha_W}} \tag{8}$$

$$cor(y_i, y_j) = \frac{\sum_{p=1}^{P} \alpha^2(p) \cdot cor(x_i, x_j)}{\sum_{p=1}^{P} \alpha^2(p) + \alpha_W} \tag{9}$$

where $\alpha(1)$ is the path loss of the LoS (line of sight) path and α_W is the amplitude of the noise w.

Now a key question is: does the channel undermine the achieved balance between reliability and secrecy? Or more specifically, will the channel shrink the balance zone of c_k? We then provide an in-depth analysis on three channel factors to see how they affect the balance zone of c_k.

Path loss and ambient noise. We first consider the channel with only path loss and ambient noise. In this case, we have $\alpha_W > 0$, $\alpha(1) > 0$, and $\alpha(2) \sim \alpha(P) = 0$. The range of the balance zone is determined by the signal-to-noise power ratio of both the legitimate channel and the eavesdropper's channel, denoted as $R_r = \alpha_r(1)/\alpha_W$ and $R_e = \alpha_e(1)/\alpha_W$. Fig. 11(a) shows the range of balance zone under different R_e and R_r obtained based on Eqs. (8) and (9). We can see that even when the eavesdropper's channel is better than the receiver's, there still exists balance zone for c_k (i.e., $Range > 0$). Only when the eavesdropper's channel is much better (e.g., when $R_e/R_r > 4$), we have to make a tradeoff between reliability and secrecy. Generally, *in most times the legitimate transceivers can always strike a proper balance between transmission reliability and secrecy with a careful control of c_k.*

Multipath. In the cases with multipath, we have $\alpha(1) \sim \alpha(P) > 0$. Fig. 12 shows the measured α in a multipath-rich corridor, where the peaks indicate α of the LoS and the reflection paths. Fig. 11(b) shows the range of balance zone obtained with multipath effect, where α is set according to the measurement result in Fig. 12. Compared with the result in Figs. 11(a), we can observe a shrink in balance zone, which means that *multipath will impose an unfair effect on the receiver and the eavesdropper, which should be compensated.*

B. Dealing with multipath effect

The typical way in dealing with multipath effect is to estimate signal's attenuation on each path (i.e., α) using a known preamble signal [15], and then recover $x(t)$ from $y(t)$

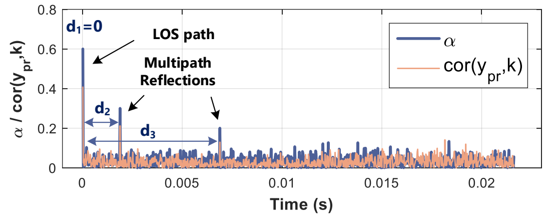


Fig. 12: Signal attenuation α on different paths.

with α. This is however infeasible in our case where signal with definite and known shape should not be transmitted for the purpose of secret transmission. In the following of this section, we first introduce the design of Rustle's preamble signal, then we show how to estimate α based on the preamble.

Preamble design. To ensure the randomness of the signal, the preamble in Rustle is also modulated using furtive modulation. To achieve a reliable preamble detection, we use different settings in modulating preambles and data. Specifically, considering that the required number of preamble signals is much smaller than bit signals, we modulate preamble and data with different keys and use a much smaller library size N ($N = 100$) in modulating the preambles. This significantly increases the gap between c_k and c_{mu} as discussed in Sec. V-D, so Rustle can use a higher c_k to modulate the preamble without sacrificing the secrecy.

Channel estimation. The challenge in estimating α is that the receiver does not know the exact shape of the preamble signal, instead, it just know that the preamble x_{pr} has a certain correlation (c_k) with the key. We solve this issue based on the fact that we do not need to know the exact shape of the preamble. In fact, knowing the expected key correlation (c_k) and the practical key correlation of the received preamble $cor(y_{pr}, k)$ is sufficient to estimate signal's attenuation on each path α. Specifically, Eq. (8) tells that α is proportional to $cor(y_{pr}, k)$, with a scaling factor C, where

$$C = \frac{c_k}{\sqrt{\sum_{p=1}^{P} \alpha^2(p) + \alpha_W^2}}$$

Fig. 12 shows $cor(y, k)$ and α obtained on the same channel, which verifies their proportional relation. So, if we can estimate C, we can then obtain $\alpha(t)$ from $cor(y_{pr}, k)$. However, there is an unknown term $\sum_{p=1}^{P} \alpha(i)$ in C.

Fortunately, since different copies of the signal usually arrive sequentially, we can actually extract α in a recursive manner. Specifically, we denote the delay of the i-th signal path as d_i. So, the first $d_2 \cdot f$ samples of the signal actually involves only the signal traveling along the shortest path, based on which we can extract $\alpha(1)$ as:

$$\alpha(1) = \sqrt{\frac{cor^2(k^{(d_1)}, y_{pr}^{(d_1)}) \cdot \alpha_W^2}{c_k^2 - cor^2(k^{(d_1)}, y_{pr}^{(d_1)})}} \tag{10}$$

where $k^{(d_2)}$ and $y_{pr}^{(d_2)}$ denotes the first $d_2 \cdot f$ samples of the key and the preamble. With the calculated $\alpha(1)$, we can then extract $\alpha(2)$ from the first $d_3 \cdot f$ samples, and then iteratively extract $\alpha(3)$, ..., $\alpha(M)$. Then with α, we can compensate the channel attenuation and recover $x(t)$ from $y(t)$.

C. Dealing with ambient noise

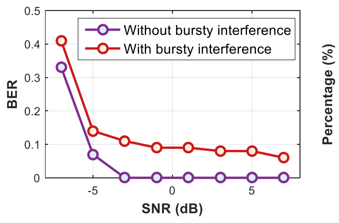
The impact of ambient noises can be mitigate by either increasing signal's key correlation c_k or increasing its volume v_x. However, how to select appropriate c_k and v_x, so that the modulated signal can be detect reliably by the legitimate receiver while cannot be detected by either the eavesdropper or the human ear. Here, v_x can be selected based on the masking effect of human's auditory system. Specifically, due to the masking effect, the modulated signal is unobtrusive when the energy of the ambient noise is higher that of the modulated signal. So, to select an appropriate v_x, we can first measure volume of the ambient noise, and then adjust v_x accordingly, making it lower than the volume of the ambient noise.

The key correlation c_k is selected in an on-demand manner: given a required transmission reliability and secrecy (p_{Rec}, p_{Eve}), we select appropriate c_k so that the receiver's decoding error PE_R is lower than p_{Rec} while the eavesdropper's detection error PE_E is higher than p_{Eve}. This is achievable by solving the following inequality:

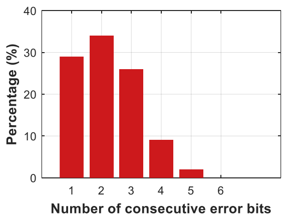
$$\begin{aligned} PE_R &= 2 - \frac{1}{2}erfc(\frac{C_{th} - \mu_k}{\sqrt{2}\sigma_k}) - \frac{1}{2}erfc(\frac{\mu_{Ek} - C_{th}}{\sqrt{2}\sigma_{Ek}}) < p_{Rec} \\ PE_E &= 2 - \frac{1}{2}erfc(\frac{C_{th} - \mu_{mu}}{\sqrt{2}\sigma_{mu}}) - \frac{1}{2}erfc(\frac{\mu_{Emu} - C_{th}}{\sqrt{2}\sigma_{Emu}}) > p_{Eve} \end{aligned} \tag{11}$$

In the above inequalities, (μ_{Ek}, σ_{Ek}) and $(\mu_{Emu}, \sigma_{Emu})$ characterize the key correlation and mutual correlation of the ambient noise, respectively, which can be easily obtained by measuring the ambient noise. (μ_k, σ_k) characterizes the key correlation of the signal on the receiver side, and (μ_{mu}, σ_{mu}) characterizes the mutual correlation of the signal on the eavesdropper side. By solving the above two inequalities, we can obtain the minimal required μ_k for high transmission reliability and the maximal tolerable μ_{mu} for high transmission secrecy. Then, if the sender knows the channel of both the receiver and the eavesdropper, it can then deduce the minimal required c_k and the maximal tolerable c_{mu} on the sender side based on Eqs (8) and (9).

However, one problem in the above method is that the eavesdropper's channel state is unknown to the sender. Although the knowledge of the receiver's channel state is reasonable (since the legitimate transceivers can cooperate to characterize the channel [16]), the knowledge of the eavesdropper's channel state is questionable. To solve this problem, our first idea is to use a conservative strategy: choosing c_k assuming that the eavesdropper has the best-case channel in the environment. So, the secrecy can be guaranteed no matter where the eavesdropper locates. However, considering that the channel state varies significantly across locations, how can the sender knows the best-case channel in the environment? Our solution is to leverage other legitimate receivers in the network. Specifically, in a typical wireless system, each sender will maintain a list of candidate receivers (i.e., its neighbors), and information about the channel state of each receiver is included in the list. So, the sender can capture a rough distribution of the channel state. It selects the best one as the eavesdropper's channel, based on which it estimates the required c_k using



(a) BER with and without bursty interference



(b) Number of consecutive error bits

Fig. 13: Error characteristics.

Eq. 11. Our experiment result in Sec. IX-D shows that even using such a conservative strategy, we can still achieve a reasonable transmission reliability – it achieves a more than 85% coverage in a $48m^2$ apartment while the risky area (where eavesdropper's error rate on detecting the signal is lower than 0.9) takes less than 5% of the apartment.

We also propose a risking strategy: estimating the required c_k assuming that the eavesdropper's channel is no better than the receiver's channel. Our evaluation result shows that the risking strategy still achieves a high secrecy – the risky area takes less than 15% of the apartment even when the coverage of legitimate transmission is higher than 95%. Of course, one can also choose an intermediate strategy. We leave the design of such strategy as our future work.

VII. Reliability Enhancement

Although we have introduced strategies to contend with channel effects, error bits can still appear. We in this section first analyze the error characteristics and reveal the dominate error type in Rustle's transmission and then introduce a new error correction mechanism to enhance transmission reliability.

A. Analysis of bit errors

Poor channel condition (e.g. low signal SNR) and bursty interferences are known to be the two dominating factors that lead to bit errors. To observe how they affect the performance of Rustle, we conduct two experiments to observe Rustle's bit error rate (BER) under different SNR. In the first experiment, we remove all the sound sources which can produce bursty interferences (e.g., the TV set) in the environment and manually generate persistent white noises as interference. The SNR of the signal is adjusted by changing the volume of the noise. In the second experiment, besides the white noise, we also turn ON the TV set to produce bursty interference. All other settings are kept the same as the first experiment.

The experiment results in Fig. 13 demonstrate that for SNR$\geq$ -3dB, Rustle achieves an almost 0 BER if there is no interference, showing its resilience to a wide range of SNR. In contrast, when there is an interference, error bits still appears even at high SNR. This is due to the fact that the furtive modulation is in essence a form of spectrum spreading, which supports even under-noise communication. This demonstrates the potential to enhance Rustle's immunity against interference by balancing between the overprotection to low SNR and under-protection to bursty interference.

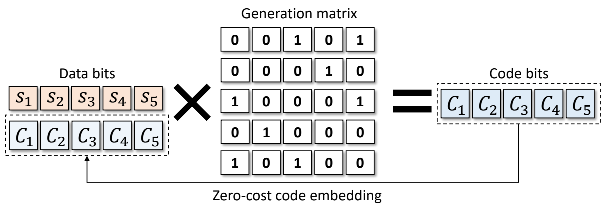


Fig. 14: An example of the coding mechanism.

B. Error correction

According to the characteristics of the data errors, we introduce an coding mechanism which is resistant to bursty interference, with zero additional cost on packet length. In the following of this section, we first introduce our coding mechanism, which is designed based on the coding mechanism introduced in [17]. Then we show how we can embed the code into the packet with zero cost on packet length.

Code bit generation. In our coding mechanism, the bit sequence $\mathbf{S} = \{s_1, ..., s_N\}$ (denoted as *data bits*) are divided into blocks where each block holds the same number of bits. We denote the block length as B_d (we have $B_d = 5$ in our implementation). Now, to encode the data bits into code bits with the block length of B_c (we have $B_c = 5$ in our implementation), we multiply the data bits with a $B_d \times B_c$ generation matrix G, as shown in Fig. 14. Each code bit C_i is generated via the following equation:

$$C_i = \sum_{j=1}^{B_d} s_i \cdot G_i^j$$

Code bit embedding. In our zero-cost coding mechanism, the code bit C_i and the data bit s_i are simultaneously embedded in the noise sequence x_i. This is feasible due to Rustle's ability to embed two different bits in one noise sequence, as we have shown in Secs. IV-B and V-B. However, embedding more bits in one noise sequence may release Rustle's resistant to low SNR channel (e.g., it may shrike the balance zone as shown in Fig. 9). Since the results shown in Fig. 13 tell that Rustle achieves almost 0 BER when SNR$\geq$-3dB, Rustle only embeds code bits if SNR is above -3dB.

Optimize the resilient to interference To be resilient to the bursty inter, the coding mechanism needs to optimize the generation matrix G_i. To reflect the interference resilience of G_i, we define its intra distance $dist(G_i)$ as the minimum distance between data bits/code bits involved in computing C_i. For example in Fig. 14, $dist(G_1) = 2$ is the distance between G_1^3 and G_1^5. Rustle's resilience to the bursty interference is determined by $dist(G_i)$. Specifically, when the interference duration is less than $dist(G_i)$, then it can lead to one of the following two results: i) It at most corrupt one data bit used for computing C_i, while C_i is correct. ii) It corrupts C_i, while the data bits used for computing C_i are all correct. For either case, we can correct the errors, meaning that G_i is resilient to interference whose duration is less than $dist(G_i)$.

We define the distance of the generation matrix G as $dist(G) = min\{dist(G_1), ..., dist(G_{B_c})\}$, which is the minimum value of all the column distances. To offer better resilience, we have to optimize the generation matrix so

TABLE I: Configurations.

sampling rate	d_R	d_E	L	N	packet length	M
44.1KHz	2m	2m	1000	500	24 symbols	1

that $dist(G)$ is maximized. Details on the optimization of the generation matrix are omitted in this paper. Readers are suggested to refer to [17].

VIII. Security of Rustle

Besides the eavesdropping attack, Rustle is also resistant to the following three types of attacks.

Side channel attack. In performing side channel attack, the attacker tries to obtain additional information (e.g., user's activity) by observing the features of the wireless signal in the scene (e.g., signal's amplitude and phase). Rustle is resistant to such attack because the legitimate transceivers in Rustle communicate with each other using random noise, where the features of the signal are randomly distorted. So the attacker cannot infer any information from the observed signal.

Jamming attack. In jamming attack, the attacker produces jamming signal once a communication is detected. That is to say, one important prerequisite in performing jamming attack is that the attacker should first detect the communication process. However, in the design of Rustle, the legitimate transceivers communicate with noise-like signal. As a result, the attacker cannot even detect the communication, not to mention jamming the process. Of course, the attacker can constantly produce jamming signal no matter whether a communication is detected or not. However, since the furtive modulation method used in Rustle is in essence a form of spectrum spreading, which supports even under-noise communication, the jamming signal should be strong enough to interfere the legitimate communication. This makes the jamming signal easily detected by the legitimate devices.

Brute Force attack. Since the library size of the generated noise signal is limited (i.e., $N = 500$), theoretically the eavesdropper can detect the transmission (i.e., capture repeated signal sequences) by performing mutual correlation over a sufficiently long acoustic signal. Actually, however, this is computationally intractable and inefficient for most devices. Specifically, if the sender transmit 20-bit packets with a period of 0.1s (a typical packet period for IoT devices), the eavesdropper has to capture repeated signal sequences with a longer than $N/(20 \cdot 10) = 2.5$ sec window. However, even a 2.5s signal can compel the eavesdropper to cost a computation overhead of $L^2\binom{2.5 \cdot f - L}{2} \geq 6 \times 10^{16}$ to detect repeated signal sequences (where $f = 48KHz$ is the sampling rate of the microphone). Furthermore, even though the attacker can support such a high computation overhead, the sender and the receiver can periodically update the key (and thus the signal library). For example, they can generate a new key K_{new} by hashing the previous key K_{pre} using the same function as $K_{new} = Hash(K_{pre})$.

IX. Evaluation

A. Experimental Setup and Methodology

Settings. We implement a prototype of Rustle using commodity hardware. Specifically, we use a Microsoft Surface Laptop 2 as a sender and two Samsung Galaxy S7 phones as the receiver and the eavesdropper. The sender is connected to a JBL PS2200 loudspeaker. Table I lists default configurations of the experiment, where d_R and d_E denote the sender-receiver and sender-eavesdropper distances.

Secrecy metric. An eavesdropper's task is to detect wireless transmission by checking the mutual correlation and the volume of the received signal with pre-defined thresholds. We use the eavesdropper's error rate (denoted as P_E) to quantify the secrecy of Rustle. Here, the error rate includes both the *miss detection rate* (P_M) and the *false alarm rate* (P_F). We have $P_E = P_M + P_F$. In the evaluation results, all the reported P_E are obtained in the case where the eavesdropper selects the best thresholds for signal detection.

Efficiency metrics. We use two metrics, BER (bit error rate) and throughput (the received bits per second), to evaluate the transmission efficiency between legitimate transceivers.

B. Reliability v.s. Secrecy

■ **Impact of c_k and v_x.**

Then we evaluate Rustle's transmission reliability (BER of the legitimate receiver) and secrecy (P_E of the eavesdropper) under different c_k and v_x. We perform experiments in two environments, an apartment and a cafe, where the average volume of the ambient noises are $33dB$ and $72dB$, respectively. The results are shown in Figs 15 and 16.

As expected, in both environments, the increase in either c_k or v_x leads to an increased transmission reliability (lower BER) while a decreased secrecy (lower P_E). In a noisier environment (the cafe), Rustle requires higher c_k or v_x to guarantee reliable transmission. Overall, we can always find cases which can simultaneously satisfy $BER < 0.1$ and $P_E > 0.9$, in both environments. Even in the case where the BER is controlled below 1%, Rustle can still achieve a high security, where the eavesdropper's error rate on detecting the signal is higher than 70%. The above results verify that Rustle can achieve simultaneous transmission secrecy and reliability if appropriate c_k and v_x are selected.

Fig. 17 summarizes the relationship between transmission reliability and secrecy under different configurations of c_k and c_{mu}. The result tells that: i) most cases are crowded in the bottom right corner of the figure (we can find that more than 30% cases in the apartment and more than 35% cases in the cafe locate in the area where $P_E > 0.9$ and $BER < 0.1$), indicating an appropriate balance between secrecy and reliability; and ii) although Rustle can simultaneously achieve high reliability and high secrecy, there is still an conflict between these two factors – to achieve a even lower BER (e.g., $BER < 1\%$), Rustle has to, to some extent, sacrifice the secrecy, resulting in a lower than 85% error rate in detecting the transmission (e.g., $P_E < 0.85$). Another observation is that Rustle's secrecy in a cafe is better than that in an apartment. The reason is that in a cafe where the volume of ambient

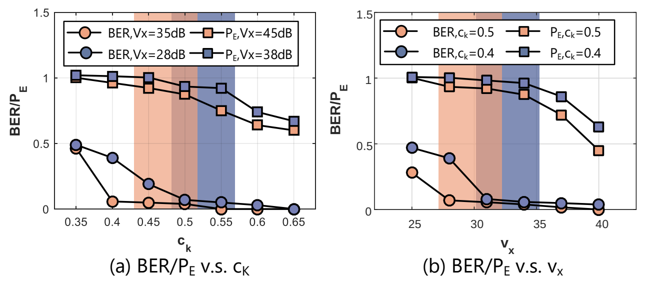


Fig. 15: BER and P_E in an apartment.

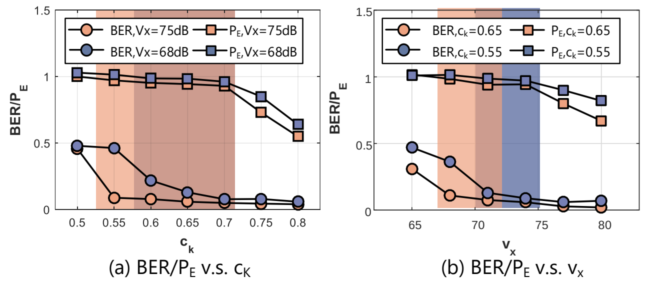


Fig. 16: BER and P_E in a cafe.

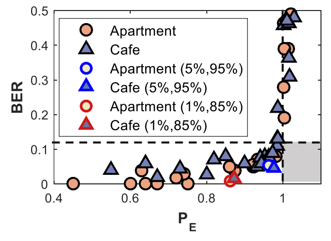


Fig. 17: BER v.s. P_E.

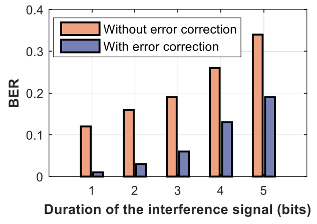


Fig. 18: Bursty interference.

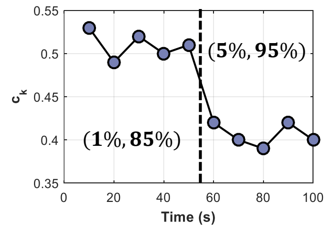


Fig. 19: Varying requirement.

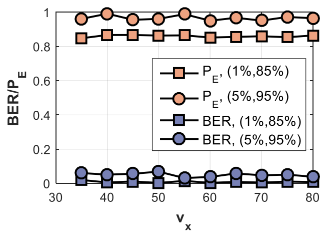


Fig. 20: Impact of noise.

noise is high and fluctuant, the modulated signal can hardly be captured by checking the signal volume.

■ **Performance under bursty interference.**

We further evaluate how the error correction method benefits the performance of Rustle, by observing Rustle's performance under bursty interference. In the experiment, we use a laptop to generate bursty interference signals with the power of 80dBm. We set the transmission interval of interference signals at 500ms and the duration of each interference signal varies from 1 bit to 5 bits long. Our proposed interleave operation can theoretically correct 3 consecutive error bits.

Figure 18 shows the BER obtained with and without the error correction method, under different duration of interference signal. We can see that: i) the BER increases with the increasing of the duration of interference signal. For example, the BER is up to 0.34 when the duration of interference signal is 5 bits long. ii) The proposed error correction method can effectively reduce the BER when there is transient jamming or interference. We find that the BER is reduced from 0.19 to 0.06 with the error correction method when the duration of interference signal is 3 bits long.

■ **Performance under varying requirement.**

Rustle features quick adaption for varying requirement on secrecy and reliability. This is achieved by adaptively selecting c_k and v_x using the method introduced in Sec. VI-C. In this experiment, we emulated a scenario that at the first stage Rustle is required to have a high reliability (i.e., with the requirement $(1\%, 85\%)$); after that, Rustle is required to maintain a high secrecy (i.e., with the requirement $(5\%, 95\%)$).

Fig. 19 plots the selected c_k during the experiment. We can see that at the first stage, Rustle set the key correlation around $c_k = 0.5$ to achieve a lower than 1% BER. Upon the requirement change, Rustle swiftly shifted to set a lower key correlation for higher secrecy.

■ **Other practical considerations.**

We now evaluate the impact of other practical factors. We test both the conservative and the risking strategies under different noises and on different devices.

Fig. 20 shows Rustle's performance under different levels of ambient noises As can be seen, not matter which strategy Rustle selects, it always simultaneously achieve high P_E (> 0.92) and low BER (< 0.08) under different noise levels. This indicates that Rustle can **automatically strike an appropriate balance between secrecy and reliability.** To further evaluate the secrecy of Rustle when a higher BER (i.e., $BER < 1\%$) is required, we extend the upper bound of the balance zone by 50% and evaluate the performance of the risking strategy. Fig. 20 shows that in this case, Rustle still achieves high secrecy ($P_E > 0.8$) when the BER is controlled under 1%.

Besides the impact of noise levels, we also evaluate the impact of different types of ambient sounds. Fig. 21(a) shows Rustle's performance under three types of ambient noises, e.g., music (74dB), human voice (66dB), and machinery sound (81dB). As can be seen, the type of the ambient noise does not have apparent impact on the performance of Rustle.

The previous experiments focus on the impact of audible sounds. We further evaluate the impact of inaudible sounds. Fig. 21(b) shows Rustle's performance under noises on three different bands. As can be seen, Rustle simultaneously achieves high secrecy and reliability under noises on both audible and inaudible bands.

Finally, to validate the feasibility of Rustle on various IoT devices, we evaluate its performance on smartphones with different brands (GALAXY S10, Xiaomi Note9, GALAXY S9, and Xiaomi Pro10) and on resource-limited IoT devices (Arduino Uno board). Since there is no default speaker and

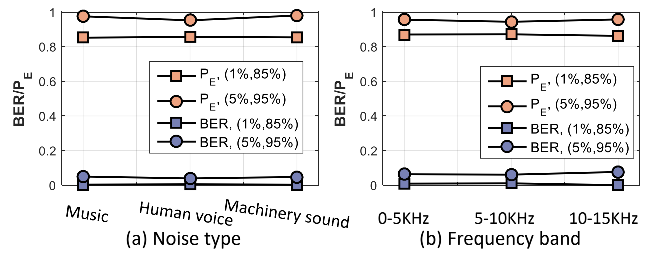


Fig. 21: Impact of different noises.

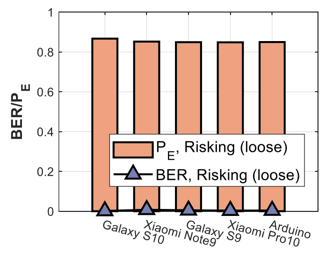


Fig. 22: Performance on various IoT devices.

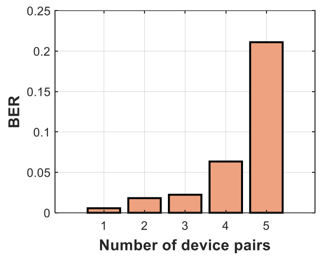


Fig. 23: Performance under interference.

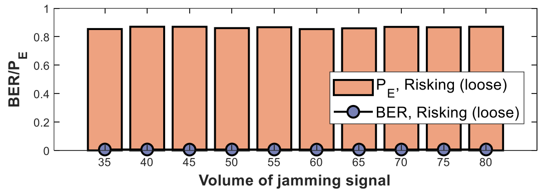


Fig. 24: Performance under jamming attack.

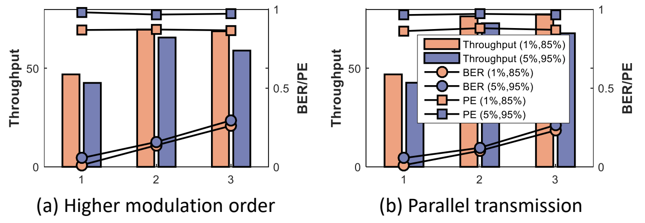


Fig. 25: Performance of M-key modulation.

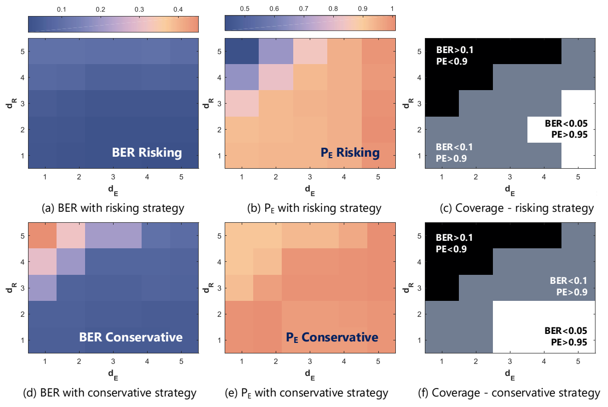


Fig. 26: Coverage.

microphone on Arduino board, we connect additional speaker module and microphone module to it. The results are shown in Fig. 22. As can be seen, the type of the device does not have apparent impact on the performance of Rustle.

■ **Performance under heavy interference.**

In the previous experiments, we only consider the cases with a single pair of transmitter and receiver. Consider that one of the key advantages of our furtive modulation technique is its resistant to the interference, we further evaluate its performance in the case where multiple pairs of devices operating at the same time. In the experiment, we let 1-5 pairs of transceivers operate simultaneously, the average BER obtained under different degree of concurrency are shown in Fig. 23. As can be seen, Rustle can still deliver a decent performance when four pairs of devices operating simultaneously. The performance crashes when there are 5 devices transmit simultaneously. So, in such a crowding scenario, the transmitter needs to avoid the interference by performing carrier sensing. We leave this to our future work.

■ **Resistant to jamming attack.**

We also perform an experiment to observe the performance of Rustle in resisting jamming attack. Fig. 24 shows the BER of Rustle obtained under different levels of random noise generated by the attacker. As can be seen, Rustle can always achieve a lower-than 0.05 BER under different noise levels. This is because that Rustle can automatically adjust the volume of the modulated signal based on the volume of the ambient noise, so that the signal can be decoded by the legitimate receiver and at the same time cannot be detected by an eavesdropper or the human ear due to the masking effect.

C. *Transmission efficiency*

Rustle's transmission efficiency is the modulation order (the number of bits per symbol). So, in this experiment, we test Rustle's performance when it uses M keys to achieve M-order modulation and a M-device parallelism. Fig. 25(a) shows the performance with higher-order modulation. As can be seen, the BER of Rustle increases with the modulation order. This is because that a higher modulation order will lead to a narrow balance zone (as shown in Fig. 9). Thus Rustle has to sacrifice the reliability for secrecy.

Fig. 25(b) shows Rustle's performance in parallel transmission. Due to the similar reason in higher-order modulation, the BER increases with higher parallelism and reaches to 0.26 when the parallelism increases to three.

D. *Coverage*

To evaluate the coverage of Rustle, we first perform a group of experiments to evaluate Rustle's BER and P_E under different combinations of sender-receiver distance (d_R) and the sender-eavesdropper distance (d_E). The performance of both the risking and the conservative strategies are evaluated in the experiments. Figs 26 show the results.

We find that under both strategies, Rustle can simultaneously achieve low BER and high P_E in most cases. The result in Figs 26 (c) and (f) tells that for both strategies, there are 72% cases achieve $BER < 0.1$ and $P_E > 0.9$ simultaneously, in which there are $22\% \sim 33\%$ cases achieve $BER < 0.05$ and $P_E > 0.95$ simultaneously. Only in the particularly unfair cases (e.g., $d_E = 1m$ and $d_R = 5m$), Rustle has to make a tradeoff between reliability and secrecy. Specifically, in the risking strategy, Rustle trades low secrecy for high transmission reliability, and thus the error rate of the eavesdropper P_E decreases to 0.45 when $d_E = 1m$ and $d_R = 5m$. But even though, Rustle can still achieve $P_E > 0.9$ in more than 80% cases. Similarly, in the conservative strategy where Rustle trade low reliability for high transmission secrecy, BER increases to 0.41 when $d_E = 1m$ and $d_R = 5m$. But it can still achieve $BER < 0.1$ in more than 80% cases.

We further conduct an in-the-wild experiment under a practical indoor setting to understand the coverage of Rustle in real-world applications. Specifically, we deploy one sender in an one-bedroom apartment as shown in Fig. 27(a). We place

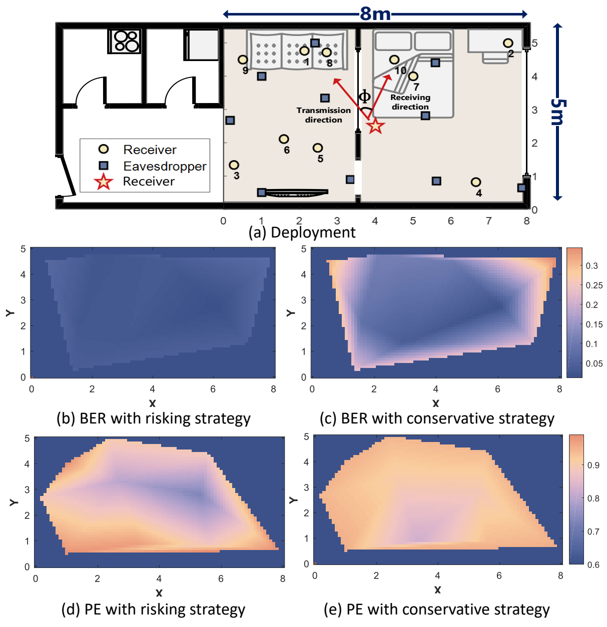


Fig. 27: Coverage in real-world applications.

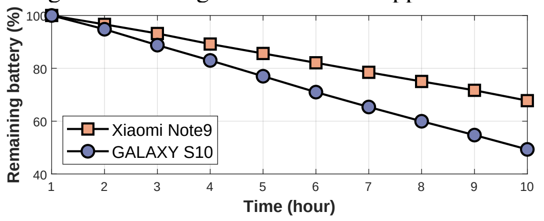


Fig. 28: Energy consumption of Rustle.

the receiver on 8 different locations with different distance and direction to the sender, the locations of the receivers are also marked in Fig. 27(a). As can be seen, some locations suffer through-obstacle transmission. On each location, the sender transmits message to the receiver with both the risking and the conservative strategies. Experiments on each location are repeated for 8 times, and each time we place an eavesdropper at different locations in the apartment. Locations of the eavesdroppers are also marked in Fig. 27(a). For both the two strategies, we obtain the BER of the receivers and P_E of the eavesdroppers on different locations.

Figs. 27 (b)-(e) show the heat maps of BER and P_E, which are obtained by applying interpolation (using the matlab function `griddata()`) on the results of different locations. As can be seen: i) with the conservative strategy, the risky area (where $P_E < 0.9$) takes less than 5% of the apartment, while the coverage of legitimate transmission is higher than 85%; ii) even in the risking strategy, the risky area can be still controlled lower than 15%, while the coverage for the legitimate transmission is higher than 95%.

E. Energy consumption

We also measure the battery consumption of Rustle on different smartphones (GALAXY S10 and Xiaomi Note9). In the experiment, we let two smartphones send packets continuously and observes the remaining battery percentage after 6 hours. The results shown in Fig. 28 tells that on both smartphones, the remaining battery percentage are larger than 50%. Considering that IoT devices usually transmit only small amount of information in a very low frequency (e.g., once per hour). Rustle is able to run for a very long period of time.

X. Related work

Covert communication via side channel. Besides signal hiding, another way to support secret communication is to build side channel leveraging physical characteristics like vibration [18], heat [19], screen light [20, 21], electromagnetic [22, 23], and etc. These methods, however, require either physical contact or specialized equipment, which make them unsuitable for the IoT scenario. Besides, they assume that the attackers do not know the channel that the transceivers use. While, none of the above is required in Rustle.

Acoustic communication. Acoustic communication has been extensively studied in [15, 24]. With a variety of modulation methods, they build reliable acoustic channel for long-range, short-range, and under water communication. However, most existing acoustic-based communication techniques are not designed for secure communication. They simply use conventional modulation methods, which makes the modulated signal easy to detect by an eavesdropper. Although there are some works designed for secure communication [4, 5], they can only defend weak eavesdroppers, under the assumptions that: i) the channel of the legal receiver is advantageous than that of the eavesdropper; and ii) the eavesdropper is a single-antenna (or single-microphone) device. These two assumptions are not always hold in practical scenarios. Different from those method, Rustle can defend a MIMO eavesdropper, and it does not assume that the receivers channel quality is better than the eavesdroppers channel. Thus, it provides a higher degree of security than the existing acoustic-based communication.

Key agreement. key agreement between transceivers is the important first step in Rustle. Many key agreement protocols have been proposed for IoT devices, where legitimate transceivers exchange secret keys by exploiting the RF or acoustic channel between them [25–29]. Rustle can use the acoustic based method to generate secret keys [26].

XI. Conclusions

Rustle aims to build covert connections among IoT devices with acoustic signal. By controlling the signal's fine-grained shape, Rustle can embed information into random signal, making the modulated signal resembles random noise to the eavesdropper. Our evaluation results tell that Rustle can achieve a lower than 1% BER while keep the eavesdropper's error rate on detecting the signal is higher than 80%.

Acknowledgment

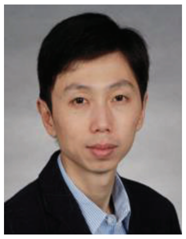
This work was supported by National Natural Science Fund of China (62272293,42050105,62061146002).

References

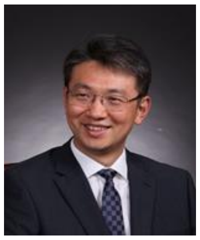
[1] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi. Information exposure from consumer iot devices. In *ACM IMC*, 2019.

[2] Y. Qiao, O. Zhang, W. Zhou, K. Srinivasan, and A. Arora. Phycloak: Obfuscating sensing from communication signals. In *NSDI*, 2016.
[3] Y. Zhu, Z. Xiao, Y. Chen, Z. Li, Max Liu, B. Y. Zhao, and H. Zheng. Et tu alexa? when commodity wifi devices turn into adversarial motion sensors. In *NDSS*, 2020.
[4] R. Nandakumar, K. K. Chintalapudi, V. N. Padmanabhan, and R. Venkatesan. Dhwani: Secure peer-to-peer acoustic nfc. In *ACM SIGCOMM*, 2013.
[5] B. Zhang, Q. Zhan, S. Chen, M. Li, K. Ren, C. Wang, and D. Ma. Priwhisper: Enabling keyless secure acoustic communication for smartphones. *IEEE Internet of Things Journal*, 1(1):33–45, 2014.
[6] S. Goel and R. Negi. Guaranteeing secrecy using artificial noise. *IEEE Transactions on Wireless Communications*, 7(6):2180–2189, 2008.
[7] H. Hassanieh, J. Wang, D. Katabi, and T. Kohno. Securing rfids by randomizing the modulation and channel. In *Usenix NSDI*, 2015.
[8] A. Chaman, J. Wang, J. Sun, H. Hassanieh, and R. R. Choudhury. Ghostbuster: Detecting the presence of hidden eavesdroppers. In *ACM MobiCom*, 2018.
[9] M. Jin, Y. He, Y. Liu, and X. Wang. Furtively connecting iot devices with acoustic noise. In *ACM/IEEE IPSN*, 2022.
[10] J. Cardoso. Blind signal separation: statistical principles. *Proceedings of the IEEE*, 86(10):2009–2025, 1998.
[11] A. Thangaraj, S. Dihidar, A. R. Calderbank, S. W. McLaughlin, and J. Merolla. Exploiting transitivity of correlation for fast template matching. *IEEE TIP*, 19(8):2190–2200, 2010.
[12] S. Gautam and D. Vaintrob. A novel approach to the spherical codes problem. Technical report, MIT, 2013.
[13] A. Aguilera and R. Perez-Aguila. General n-dimensional rotations. In *WSCG*, 2004.
[14] M. Jin and X. Wang. Pairing iot devices with spatial keys. In *ACM/IEEE IPSN*, 2022.
[15] Q. Wang, K. Ren, M. Zhou, T. Lei, D. Koutsonikolas, and L. Su. Messages behind the sound: Real-time hidden acoustic signal capture with smartphones. In *ACM MobiCom*, 2016.
[16] Q. Wang, K. Ren, G. Li, C. Xia, X. Chen, Z. Wang, and Q. Zou. Walls have ears! opportunistically communicating secret messages over the wiretap channel: from theory to practice. In *ACM CCS*, 2015.
[17] Z. Yin, W. Jiang, R. Liu, S. M. Kim, and T. He. Safetynet: Interference protection via transparent phy layer coding. In *IEEE ICDCS*, 2020.
[18] N. Roy, M. Gowda, and R. R. Choudhury. Ripple: Communicating through physical vibration. In *Usenix NSDI*, 2015.
[19] M. Guri, M. Monitz, Y. Mirski, and Y. Elovici. Bitwhisper: Covert signaling channel between air-gapped computers using thermal manipulations. In *ACM CSF*, 2015.
[20] K. Zhang, C. Wu, C. Yang, Y. Zhao, K. Huang, C. Peng, Y. Liu, and Z. Yang. Chromacode: A fully imperceptible screen-camera communication system. In *ACM MobiCom*, 2018.
[21] A. Wang, C. Peng, O. Zhang, G. Shen, and B. Zeng. Inframe: Multiflexing full-frame visible communication channel for humans and devices. In *ACM MobiSys*, 2015.
[22] Z. Yang, Q. Huang, and Q. Zhang. Nicscater: Backscater as a covert channel in mobile devices. In *ACM MobiCom*, 2018.
[23] M. Gao, F. Lin, W. Xu, M. Nuermaimaiti, J. Han, W. Xu, and K. Ren. Deaf-aid: Mobile iot communication exploiting stealthy speaker-to-gyroscope channel. In *ACM MobiCom*, 2020.
[24] Y. Bai, J. Liu, L. Lu, Y. Yang, Y. Chen, and J. Yu. Batcomm: Enabling inaudible acoustic communication with high-throughput for mobile devices. In *SenSys*, 2020.
[25] W. Xi, C. Qian, J. Han, K. Zhao, S. Zhong, X. Li, and J. Zhao. Instant and robust authentication and key agreement among mobile devices. In *ACM CCS*, 2016.
[26] P. Xie, J. Feng, Z. Cao, and J. Wang. Genewave: Fast authentication and key agreement on commodity mobile devices. In *ICNP*, 2017.
[27] H. Liu, Y. Wang, J. Yang, and Y. Chen. Fast and practical secret key extraction by exploiting channel response. In *IEEE INFOCOM*, 2013.
[28] J. Han, A. Chung, M. Sinha, M. Harishankar, S. Pan, H. Noh, P. Zhang, and P. Tague. Do you feel what i hear? enabling autonomous iot device pairing using different sensor types. In *IEEE S&P*, 2018.
[29] W. Xu, G. Revadigar, C. Luo, N. Bergmann, and W. Hu. Walkie-talkie: Motion-assisted automatic key generation for secure on-body device communication. In *ACM/IEEE IPSN*, 2016.

Meng Jin is currently an assitant professor at School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University. She is previously a post-doctoral researcher at School of Software and BNRist, Tsinghua University. She received the B.S., M.S. and Ph.d. degrees in Computer Science from Northwest University, Xian, China, in 2012, 2015, and 2018, respectively. Her main research interests include Backscatter Communication, Wireless Network Co-existence at 2.4GHz, mobile sensing, and Clock Synchronization.



Yuan He is an associate professor in the School of Software and BNRist of Tsinghua University. He received his B.E. degree in University of Science and Technology of China, his M.E. degree in Institute of Software, Chinese Academy of Sciences, and his PhD degree in Hong Kong University of Science and Technology. His research interests include Internet of Things, wireless and sensor networks, mobile and ubiquitous computing. He is a senior member of the IEEE and a member of ACM.



Yunhao Liu received his BS degree in Automation Department from Tsinghua University. He received an MS and a Ph.D. degree in Computer Science and Engineering at Michigan State University, USA. Yunhao is now MSU Foundation Professor and Chairperson of Department of Computer Science and Engineering, Michigan State University, and holds Chang Jiang Chair Professorship at Tsinghua University. He is an ACM Distinguished Speaker and now serves as the Editor-in-Chief of ACM Transactions on Sensor Networks. His research interests include sensor network and pervasive computing, peer-to-peer computing, IOT and supply chain. Yunhao is a Fellow of the IEEE and ACM.

Xinbing Wang received the B.S. degree (Hons.) from the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, in 1998, the M.S. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2001, and the Ph.D. degree, majoring in the electrical and computer engineering and minoring in mathematics, from North Carolina State University, Raleigh, NC, USA, in 2006. He is currently a professor with the Department of Electronic Engineering, Shanghai Jiao Tong University. He has been an Associate Editor for the IEEE/ACM Transactions on Networking and the IEEE Transactions on Mobile Computing, and a member of the Technical Program Committees of several conferences including the ACM MobiCom 2012, the ACM MobiHoc 20122014, and the IEEE INFOCOM 20092017.