SpiderMon: Towards Using Cell Towers as Illuminating Sources for Keystroke Monitoring

Kang Ling, Yuntang Liu, Ke Sun, Wei Wang, Lei Xie and Qing Gu
State Key Laboratory for Novel Software Technology, Nanjing University
{lingkang,yuntangliu,kesun}@smail.nju.edu.cn, {ww,lxie,guq}@nju.edu.cn

***Abstract*—Cellular network operators deploy base stations with a high density to ensure radio signal coverage for 4G/5G networks. While users enjoy the high-speed connection provided by cellular networks, an adversary could exploit the dense cellular deployment to detect nearby human movements and even recognize keystroke movements of a victim by passively listening to the CRS broadcast from base stations. To demonstrate this, we develop SpiderMon, the first attempt to perform passive continuous keystroke monitoring using the signal transmitted by commercial cellular base stations. Our experimental results show that SpiderMon can detect keystroke movements at a distance of 15 meters and can recover a 6-digits PIN input with a success rate of more than 51% within ten trials when the victim is behind the wall.**

I. Introduction

Keystroke inference attacks are extremely dangerous since the attacker could infer the content or even passwords typed by the user through side-channels that can hardly be detected. Existing works have used videos [1], [2], Inertial Measurement Units (IMU) [3], [4], and sound signals [5]–[9] in side-channel attacks that effectively infer the keystroke sequence, see Table I. Recently, researchers discovered that Wi-Fi radio signals can also be used as the medium for keystroke inference attacks [10]–[13]. However, most of these existing attack models are short-ranged or requires active signal transmission.

In this paper, we first show that an attacker can passively listen to the commercial 4G/5G signals and infer the keystroke sequence of a victim at a distance of 15 meters (Figure 1). As cellular network operators are using high-density deployments to improve radio signal coverage for 4G/5G networks, such attacks could be pervasive in the near future. Currently, for outdoor areas, macro/micro Base Stations (BSs) are deployed with a high density of more than 0.3 BS/km^2 in urban regions [14]. For indoor areas, radio repeaters and femtocells are deployed in most buildings to improve the radio signal quality [15]. As envisioned by the Ultra-Dense Networks (UDN) in 5G networks, the distance between cellular access points could be a few meters for indoor deployments and 50 meters for outdoor deployments [16]. While users enjoy the high-speed connections provided by 4G/5G cellular networks, such dense cellular deployment leads to severe information leakage issues that most users are unaware of.

The cellular signal is a new type of side-channel attack medium that could be more harmful than Wi-Fi signals. First, cellular-based attackers are passive listeners. They use the signal transmitted by commercial cellular BSs as the "illuminating sources". Therefore, it is harder to detect these attackers since they do not transmit any signal. Second, cellular signals have larger coverage areas than Wi-Fi signals. Compared to Wi-Fi APs that are mostly installed in buildings, cellular signals cover both outdoor and indoor areas. Third, cellular BSs provide highly stable reference signal sources. Cellular BSs use GPS-regulated oscillators and low-noise amplifiers to generate Cell-Specific Reference Signal (CRS) at a regular rate of up to 4,000 times per second, which are more stable in both the phase and the amplitude than the signals generated by low-end Wi-Fi devices. Finally, Wi-Fi transmissions could be easily blocked since they use Carrier-Sense Multiple Access (CSMA) protocols. However, it is against FCC regulations to interfere with cellular transmissions. Thus, users cannot protect themselves by transmitting an interfering signal, as suggested in PhyCloak [17].

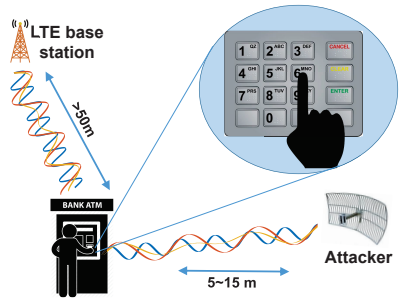


Figure 1. SpiderMon leverage cellular base stations as illuminating sources for passive keystroke monitoring.

We develop SpiderMon[1], a system that performs long-range keystroke monitoring using the signal transmitted by commercial cellular BSs. The design of SpiderMon faces three technical challenges. First, capturing the subtle changes caused by the keystroke movements at a distance of 15 meters is challenging. To address this challenge, we first use a directional antenna to amplify the signal reflected by the victim, as well as reducing the interferences of nearby movements. We then design a block Principal Component Analysis (PCA) algorithm that further amplifies the signal by combining signals in different subcarriers. Second, it is challenging to infer the keystroke sequence of a continuous typing process, where the victim types in a natural manner by continuously moving from one key to the next. Existing works treat each keystroke

[1]We name the system as SpiderMon because it monitors the victim by the small disturbance of a time-frequency grid formed by LTE CRS as shown in Figure 2(d), just as a spider that uses its web to detect the prey.

Table I
COMPARISON AMONG SIDE-CHANNEL BASED KEYSTROKE INFERENCE METHODS.

System	Attack Distance	Side-Channel Signal	Passive Listening	Continues Typing	NLOS
Owusu *et al.* [3]	On device	IMU (Smartphone)	Yes	Yes	/
Liu *et al.* [4]	Wearable	IMU (Smartwatch)	Yes	Yes	/
Shukla *et al.* [1]	5 meters	Video	Yes	Yes	No
Sun *et al.* [2]	2 meters	Video	Yes	Yes	No
Asonov *et al.* [8]	1 meter	Acoustic	Yes	Yes	/
Zhu *et al.* [6]	40 centimeters	Acoustic	Yes	Yes	/
Wikey [10]	30 centimeters	Wi-Fi	No	No	Yes
WindTalker [12]	1.5 meters	Wi-Fi	No	No	Yes
SpiderMon	**5∼15 meters**	**LTE**	**Yes**	**Yes**	**Yes**

separately by assuming that the user always returns to a given posture after each keystroke [10]. To handle continuous typing, we model the process as a Hidden Markov Model (HMM) and use the LTE signal to infer the transition between subsequent keystrokes. Third, the LTE signal contains both data transmission and reference signals so that the raw data rate is 122.88 MBytes per second, which makes real-time data processing and logging a challenge. To enable long-term monitoring, we build a signal processing frontend running on a workstation that compresses the measurements to a rate of 800 kBytes per second so that the results can be efficiently processed and stored in real-time for hours.

Our experimental results show that SpiderMon can detect 95% keystrokes at a distance of 15 meters. When the victim is behind the wall at a distance of 5 meters, SpiderMon can recover a 6-digits PIN input with a success rate of more than 51% within ten trials and this accuracy is above 36% at 15 meters with line-of-sight.

In summary, we have made the following contributions:

- To the best of our knowledge, we are the first to show that commercial 4G/5G cellular signals can be used for fine-grained human activity monitoring.
- We build a real-time cellular signal analysis system with Commercial Off-The-Shelf (COTS) USRP devices and workstations. Our system can process commercial LTE signals with a bandwidth of 20 MHz and extract $4,000 \times 200$ CRS samples per second in real-time.
- We propose to leverage the HMM to infer continuous keystroke sequences. Our extensive evaluations on keystroke sequence inference show that this method outperforms the traditional individual keystroke recovery scheme.

II. RELATED WORK

We divide the existing related work into the following four areas: LTE physical layer measurements, Radio Frequency (RF) based activity monitoring systems, keystroke inference attacks, and protection against RF-based attacks.

LTE Physical Layer Measurements: Existing LTE physical layer measurement tools mainly focus on the networking or ranging problem. LTE physical layer information, such as the Channel Quality Indicator (CQI), can be used in cross-layer design to improve TCP throughput of the cellular network [18], [19]. The real-time LTE radio resource monitor (RMon) extracts the PHY-layer resource allocation information to help LTE video streaming [20]. LTEye uses USRP N210 to decode LTE signal with a bandwidth of 10 MHz and to perform user localization [21]. Soft-LTE uses the Sora software-radio to implement the LTE uplink with a full bandwidth but does not implement the downlink [22]. Marco *et al.* [23] proposed a method for extracting TOA information from LTE CIR signals and achieved 20 meters accuracy for vehicular position tracking. However, most of these systems [20], [21], [24] do not support real-time operations on the full 20 MHz LTE bandwidth.

RF-based Activity Monitoring Systems: Different types of RF signals, including Wi-Fi [25]–[28], FMCW radar [29], [30], 60GHz radar [31], [32], and RFID [33], [34], have been used for human activity monitoring. Most of the above RF-based attacks require an active transmitting device to be placed around the victim. There are systems that use signals transmitted by GSM BSs to perform through wall monitoring [35]. However, GSM-based systems only extract the coarse-grained Doppler shift data, while LTE-based systems can measure the signal phase with high accuracy.

Keystroke Inference Attacks: Existing keystroke inference attacks use different types of sensors to capture the keystroke signal, including sound [5]–[8], IMU [3], [4], video [1], [36], and RF signals [10], [12], [13]. Asonov *et al.* [8] first demonstrated that different keys can be distinguished by their unique typing sounds. Zhuang *et al.* [7] and Berger *et al.* [37] improved keystroke recognition accuracy by adding a language model. Liu *et al.* [4] achieved 65% inference accuracy in top-3 candidates using the IMU on a smartwatch. Sun *et al.* [36] detected and quantified the subtle motion patterns of the back of the device induced by a user's keystrokes using videos. WiPass [13] and WindTalker [12] further uses the Wi-Fi CSI to snoop the unlock patterns and PINs on mobile devices. However, these methods have their own shortcomings. Sound and Wi-Fi-based methods tend to work only in limited distances. IMU-based solutions need to crack the victim's wearables, while video-based solutions are limited by lighting conditions and obstructions such as ATM keyboard cover.

Protection against RF-based Attacks: Most of existing privacy protection systems transmit interfering signals to prevent attackers from measuring key RF parameters that are vital for activity recognition. PhyCloak [17] leverage an RF signal-relay to disturb the amplitude, delay, and Doppler shift of the signal received by the attacker so that they cannot reliably infer the activity of the user. Aegis [38] uses randomized amplifications, fan movements, and antenna rotations to distort the same set of RF signal parameters. However, these protection schemes actively

transmit signals in the targeting frequency band so that they cannot be applied to cellular-based attacks, as it is against FCC regulations to transmit interfering signals in the licensed band.

III. Attack Scenario and LTE Background

In this section, we first present the attack scenario of our system. We then introduce the background of LTE system and discuss its protocol design with a focus on downlink Cell-Specific Reference Signals (CRS).

A. Attack Scenario

We consider an attack scenario where the adversary attempt to infer the PIN code of a user when he/she inputs it on an ATM or a smart lock door. The adversary may not have direct access to the target, but can deploy equipments at a distance of 5∼15 meters, *e.g.*, from a building across the road or behind a nearby wall. We assume that there is at least one LTE base station within a distance of 150 meters to the victim. The LTE coverage could be provided by a macro-cell or an indoor small cell. This requirement usually can be fulfilled in most urban areas. By passive listening to the LTE signal reflected by the victim, the adversary may infer the PIN input by the victim using a probability model.

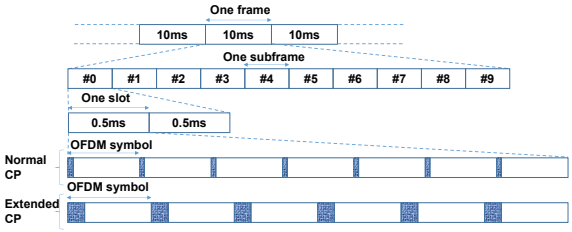
B. LTE Primer

We give a brief introduction to the LTE signal format and show how LTE signals form a time-frequency grid that can be used for human activity monitoring. Note that the 5G cellular system uses a similar OFDM modulation scheme and frame structure as in the LTE system. Therefore, most of the following discussion applies to both 4G and 5G systems.

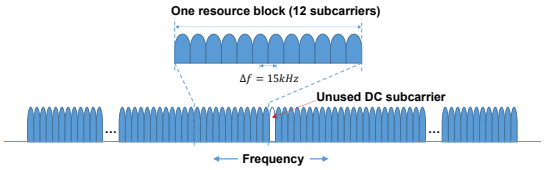
Time Domain: In the time domain, LTE BSs transmit radio frames that have a fixed duration of $10ms$. Each frame contains ten subframes with a duration of $1ms$ and each subframe contains two slots of $0.5ms$. Depending on the configuration of the BS, each slot consists of six (in case of extended cyclic prefix) or seven (in case of normal cyclic prefix) OFDM symbols which have durations of $66.67\mu s$.

Frequency domain: In the frequency domain, the OFDM symbol contains a series of subcarriers with a frequency interval of $\Delta f = 15$ kHz, as in Figure 2(b). The commonly used bandwidths for LTE signals are 5, 10 and 20 MHz, which consist of 300, 600, and 1200 subcarriers, respectively.

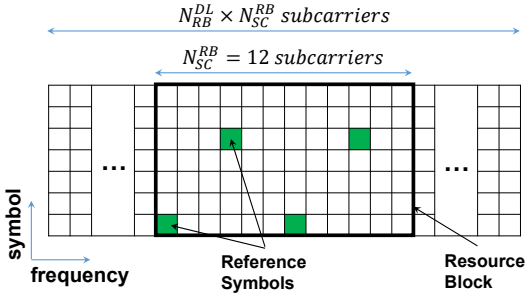
Time-Frequency Grid: The radio resources in LTE are scheduled in units called Resource Blocks (RBs), which consists of N_{SC}^{RB} =12 subcarriers in the frequency domain and lasts one slot ($0.5ms$) in the time domain, as in Figure 2(c). The LTE BS transmits the Cell-Specific Reference Signal (CRS) in all downlink RBs. The CRS is transmitted at four different locations in each RB with two CRS separated by six subcarriers in each of the two predefined symbols, as in Figure 2(c). Therefore, the CRS forms a dense time-frequency grid at fixed time and frequency intervals. For example, a Time Division Duplex (TDD) base station that has N_{RB}^{DL} =100 RBs (20 MHz bandwidth) will transmit CRS at 200 different



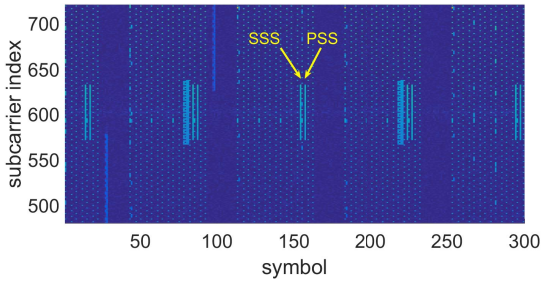
(a) Time domain: frames, subframes, slots and symbols.



(b) Frequency domain: subcarriers and resource blocks.



(c) Each slot contain N_{RB}^{DL} RB, each RB contain 12 subcarriers in the frequency domain, and 0.5 ms in the time domain.



(d) CRS (shown as small dots) and PSS/SSS for a commercial TDD base station (subcarriers around the DC subcarrier).

Figure 2. Illustration of the time-frequency grid of LTE reference signals.

subcarriers on two symbols in each slot ($0.5\ ms$). Figure 2(d) shows the CRS grid captured from a commercial TDD base station. Note that for TDD, there are some time slots reserved for uplink so that the BS does not transmit in these slots. In our experiments, the BS transmits in 14 slots in the 20 slots of each frame so that the CRS is sent in 2,800 symbols (100 frames × 14 slots × 2 symbols) per second, and 200 subcarriers (100 RB × 2 subcarriers) per symbol.

C. CRS as a Side Channel

In LTE systems, the User Equipments (UEs), *e.g.*, mobile phones, use the CRS to estimate the Channel Frequency Response (CFR) of the downlink channel. The transmitted value of CRS is predefined in the LTE protocol [39] determined by the Physical Cell ID (PCI) and slot number. Suppose that the BS transmits $S(f,t)$ on a given subcarrier f at a given time t.

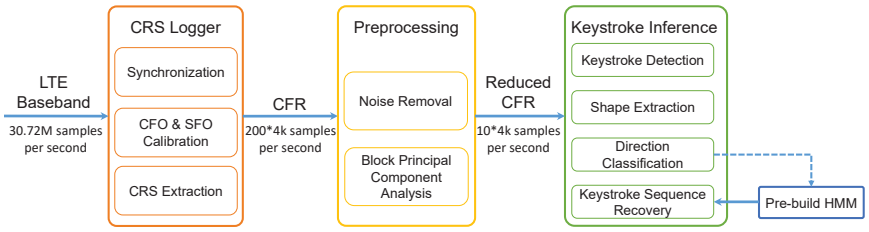


Figure 3. System overview of SpiderMon.

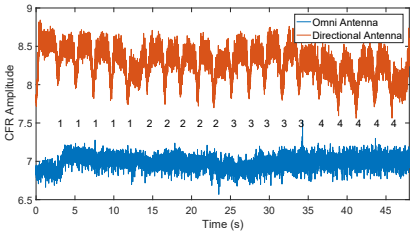


Figure 4. Omnidirectional and Directional antenna comparison.

In case that the received signal at the UE is $R(f,t)$, the CFR can be calculated by:

$$H(f,t) = \frac{R(f,t)}{S(f,t)}. \quad (1)$$

Signals received by the antenna is a superposition of transmitted signal from multiple paths [40]. Suppose a radio signal arrives at the receiving antenna through K different paths, then the CFR can be given as:

$$H(f,t) = \sum_{k=1}^{K} a_k(f,t) e^{-j2\pi f \tau_k(t)}, \quad (2)$$

where $a_k(f,t)$ represents the attenuation and initial phase offset of the k^{th} path, $e^{-j2\pi f \tau_k(t)}$ is the phase shift on the k^{th} path, and $\tau_k(t)$ is the path delay. With this model, the nearby human movements can be reflected in CFR measurement fluctuations based a similar model as in Wi-Fi systems [25], [27], [41].

IV. System Design

The structure of SpiderMon is shown in Figure 3. The LTE baseband signal is captured by USRP B210 software radio front-end and transferred to a hosting workstation using a USB 3.0 interface. We use the standard 30.72 MHz sampling rate where each sample is a complex number with two-bytes real and imaginary parts. Then we use *CRS Logger* implemented in C++ to extract CRS and CFR estimations at a rate of 4000×200 complex samples per second. At last, the CFR estimations are transmitted to a Data Preprocessing module which uses MATLAB to analyze and visualize the CRS in real-time.

A. CRS Logger

The CRS Logger consists of three components: synchronization, CFO/SFO calibration, and CRS extraction.

Synchronization: The first step for synchronization is to find the carrier frequency of a nearby LTE BS and tune the USRP to its carrier frequency. This could be done by scanning the entire LTE frequency band or using a smartphone in the engineering mode to get the U-ARFCN codes, which indicate the carrier frequencies used by neighboring BSs. The second step is searching for the Primary Synchronization Signal (PSS) to find the boundary of the subframes and symbols. This step uses a computational intensive cross-correlation operation over the whole frame to match the PSS, but it is only performed at the searching stage. After the first searching stage, we only need to perform cross-correlation within five samples of the expected PSS location to keep track of the PSS. The third step is searching for the Secondary Synchronization Signal (SSS) and extract the Physical Cell ID (PCI). We use the location of the detected PSS to capture the SSS and calculate the PCI using both the PSS and SSS.

CFO/SFO Calibration: As the transmitting BS and the receiver run at different clocks, there are both Carrier Frequency Offset (CFO) and Sampling Frequency Offset (SFO) in the received baseband signal [42], [43]. If we do not calibrate these frequency offsets, they may accumulate and the system will loose synchronization after several minutes of continuous monitoring. We first use a high-quality clock source (OctoClock CDA-2990) that has a frequency accuracy of 25 ppb to keep the CFO between our receiver and the transmitting BS to be smaller than 100 Hz. However, there are still considerable residual phase offsets caused by CFO and SFO in the CFR. The residual phase offset at a subcarrier f can be modeled as [42], [43]:

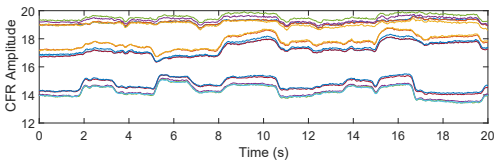
$$\varphi(f,t) = \theta(f,t) + 2\pi t \delta_{CFO} + \frac{2\pi t(f - f_c)}{f_s} \delta_{SFO}, \quad (3)$$

where $\theta(f,t)$ is the combination of the initial phase and the phase change caused by the activity. The last two components are phase offsets caused by the CFO and SFO.

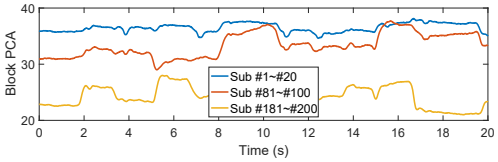
For the CFO calibration, we use the phase of the center subcarrier to estimate δ_{CFO} since it always has an SFO of zero [42]. We use linear regression over the phase history of the center subcarrier with a duration of one second to estimate the current δ_{CFO}. We then use an Exponential Moving Average (EMA) to further smooth the CFO estimation over consecutive seconds. After that, we compensate the CFO on all baseband samples using the smoothed CFO estimation by multiplying the baseband signal with an estimated phase shift.

The SFO is caused by sampling clock differences between the transmitter and the receiver. For the SFO calibration, we track the SFO by locating the cross-correlation peak of the PSS. To correct the sampling offsets, we either skip a single sample or duplicate a sample so that the sampling point of the receiver is moved by one sample in the opposite direction. With CFO/SFO calibration, we can keep the system synchronized for a long duration (several hours).

CRS Extraction: Based on the PCI obtained from the synchronization step, we can calculate in which subcarriers the CRS are transmitted as well as the value of the CRS [39].



(a) CFR signals in different subcarriers, from top to bottom: $\#1 \sim \#5$, $\#81 \sim \#85$, and $\#181 \sim \#185$.



(b) Block PCA results. The first principal components correspond to subcarriers $\#1 \sim \#20$, $\#81 \sim \#100$, and $\#181 \sim \#200$.

Figure 5. Performance of the block PCA algorithm.

After that, we calculate the CFR estimation for each symbol and subcarrier based on Eq. (1).

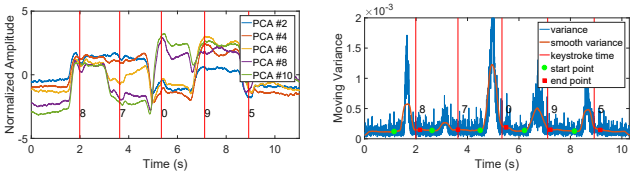
B. Data Preprocessing

The Data Preprocessing module takes the CFR values and performs the following two steps: noise removal and block principal component analysis.

Noise Removal: We first reduce the impact of multi-path interference by directional antennas. Compared to omnidirectional antennas, directional antennas amplify signals in the beam direction and reject signals in other directions. Figure 4 compares the CFR captured by a directional antenna and an omnidirectional antenna at one of the 200 subcarriers at a distance of 10 meters. Due to the high noise level, the keystroke movements are submerged in the noisy signal collected by the omnidirectional antenna. But, with the help of the directional antenna, we can easily determine the CFR variations corresponding to each keystroke event.

The raw signals captured by directional antennas are still distorted by high-frequency noises. As the hand/finger movements in keystroke input induce CRS variations with frequencies between $2 \sim 30$ Hz [12], we then use a moving-average filter to remove the high-frequency noises. Figure 5(a) shows the signal after the low-pass filter at selected subcarriers.

Block Principal Component Analysis: Most of the CFR samples are redundant, so they introduce unnecessary computational costs in the keystroke recognition stage. We use PCA (Principal Component Analysis) to extract most principal components from raw CFR signals. Figure 5(a) shows the waveform of different LTE subcarriers, we can clearly observe that signals between distant subcarriers have smaller correlations. Based on this observation, we first divide 200 subcarriers into 10 blocks, then each block performs PCA and takes the first principal component. Thus, the block PCA algorithm outputs ten principal components. Figure 5(b) shows an example of block PCA results in three blocks, where we can clearly observe the keystroke events. Compare to traditional PCA performed directly on overall 200 subcarriers, block PCA can reserve more representative information while squeezing the data size. Figure 15(b) shows that using block PCA has about 8% performance improvement over traditional PCA.



(a) Principal components (b) Keystroke detection result

Figure 6. Keystroke detection with smooth variance of the block PCAs.

V. Keystroke Monitoring

In the keystroke monitoring attack, the adversary points the antenna towards the victim (ensure that the target is within the receiving angle of the directional antenna) while he/she is typing in order to intercept the typing content. We focus on attacking the keystrokes input on numerical keypad as shown in Figure 10, which is widely used on ATM and doors for inputting the PIN number. The attack contains two steps: keystroke detection and keystroke recognition.

A. Keystroke detection

In the keystroke detection step, we use a moving variance algorithm to detect each keystroke event. Figure 6 shows the keystroke detection process. We first calculate the variance from the block PCA results. Once the variance exceeds an empirically determined threshold, the system detects a keystroke event. Sometimes one keystroke movement may introduce multiple separated variation peaks, we treat these movements as one keystroke if their time interval is less than 0.1 second. The keystroke detection result is shown in Figure 6(b). The vertical red lines are the groundtruth of the keystroke time-points provided by a key logger and the green/red dots are the detected keystrokes start/end time-points.

After detecting a keystroke movement with start and end points, we calculate the midpoint of these two points and segment the data for a period of time near the midpoint as the waveform of the keystroke (typically two seconds in our experiments). Our keystroke detection works well when there is no interference around. However, it can hardly detect a keystroke when there are objects moving around the victim. In the future, we plan to use more antenna to separate nearby objects.

B. Keystroke recognition

Existing works treat each keystroke separately by assuming that the user always returns to a given posture after each keystroke [6], [10]. In case of continuous typing, our key observation is that the CFR measurements indicate the hand/finger movements between keys, instead of the key press. We model the process as a Hidden Markov Model (HMM) to infer the transition between subsequent keystrokes. Note that existing works such as Zhuang *et al.* [7] using HMM methods to reveal text input are based on language model, which is significant different to our method, and it can not

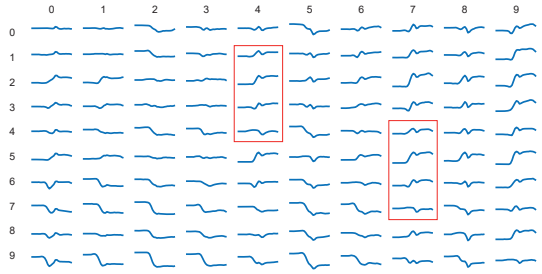


Figure 7. Keystroke movement waveforms for continuous typing.

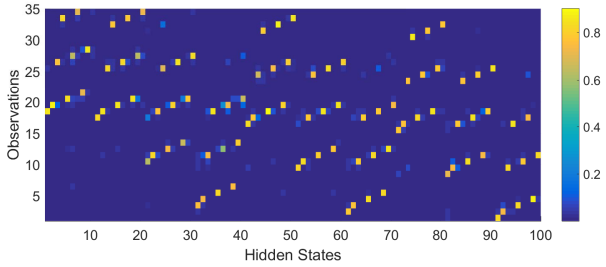


Figure 8. HMM observation probability matrix.

be applied to the PIN inference for random numbers. The keystroke recognition process has four steps: shape extraction, movement direction classification, building Hidden Markov Model, and key sequence recovery.

Shape Extraction: We first extract the waveform shapes at keystroke events to determine the movements between keys. To extract the waveform shape, we perform wavelet decomposition on each PCA component and use the level-8 approximation coefficients as the output feature. For keystroke movement with a duration of two seconds, we get a vector of length 28 for each PCA component.

Figure 7 shows the 10×10 possible movements between the numerical keys. The waveform of the i-th row and the j-th column represents the average waveform of the keystroke movement from the start position of key i to key j. From Figure 7, we can observe the following patterns: First of all, for any column, the waveforms of different rows have significant differences. As we mentioned before, the waveform of the keystroke movement is related to the starting and ending positions instead of the pressed key. Second, the farther the moving distance is, the more fluctuations shown in the waveforms, for example, '09' has large fluctuations, but '66', '99' has only one small spike corresponding to the keypress motion. Third, by comparing waveforms in two groups of boxes as we have indicated in the figure, we can find that: key pairs with same moving distance and direction have similar waveforms, which makes directly classifying these 100 keystroke movement pairs challenging.

Movement Direction Classification: We use SVM-based classifier to determine the keystroke movements. A straightforward approach is to directly classify the 100 possible movements between different keys into 100 classes. However, this method has two disadvantages. First, training a 100-category classifier requires a huge amount of training data to cover all the different cases. Second, keystroke movements with similar direction and distance induce very similar waveform

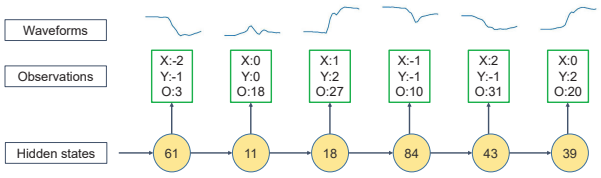


Figure 9. The state and transitions of HMM.

shapes. For example, moving from key '1' to '4' has a similar waveform as moving from '4' to '7', as they both move up by one key. We can observe this similarity in many key pairs as we indicated in Figure 7. Therefore, we use a decoupled classifier to determine the movement distance and direction. We train two classifiers, one for the "x" (horizontal) direction and the other for the "y" (vertical) direction as shown in Figure 10. As there are only five different possible movements, from -2 to 2 keys, in the horizontal direction and seven different movements, from -3 to 3 keys, in the vertical direction, the two SVM classifier are 5-category and 7-category respectively. Thus, we can use a small number of keystroke samples to train the classifiers. Figure 14 shows the resulting confusion matrix of these two classifiers. While the classifiers for keystroke movement do not provide highly accurate movement classification results, these results can serve as useful inputs to our HMM-based keystroke sequence recovery algorithm.

Building Hidden Markov Model: We model the keystroke process with a HMM indicated as $\lambda = (N, M, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$. In the HMM, N is the number of hidden states. We use the consecutive keystrokes as the hidden state, *i.e.*, a state '16' means the user moves from key '1' to key '6'. As there are 100 possible key pairs, we have $N = 100$ in our model. The parameter M is the number of possible observations for hidden states. As we use the results of the two classifier that gives five possible horizontal movements and seven possible vertical movements, we have $M = 5 \times 7 = 35$. The observation probability matrix $\boldsymbol{B}$ gives the possibility that a given observation can be observed in a hidden state. Thus, the observation matrix $\boldsymbol{B}$ is a $N \times M$ matrix with $B_{jk} = P(\text{observation } k|\text{state } s_j)$. The transition probability matrix $\boldsymbol{A}$ is the possibility a hidden state is transmitted to another hidden state, *i.e.*, the user moves to a new key. The transition probability matrix $\boldsymbol{A}$ is a $N \times N$ matrix with $A_{ij} = P(\text{state } s_j \text{ at time } t+1|\text{state } s_i \text{ at time } t)$. The initial state distribution vector $\boldsymbol{\pi}$ indicates the possibility at the start of the key sequence.

To build the HMM, we need to determine the parameters $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{\pi}$. The transmission probabilities between hidden states given in the matrix $\boldsymbol{A}$ can be predefined by the natural continuity of the typing process. For example, if we assume equal probability to type any keys, the hidden state '09' has a probability of 0.1 to transfer to states '90', '91', ..., '99', but cannot transfer to the state '87' because state '87' does not begin with the key '9'. We can also use a uniform distribution for the initial state distribution $\boldsymbol{\pi}$. The observation probability matrix $\boldsymbol{B}$ is determined through the training samples. We first collect typing waveform shapes for different key pairs. Then we use the two movement SVM classifier to calculate the

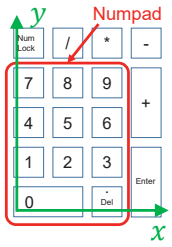


Figure 10. Numerical keypad.

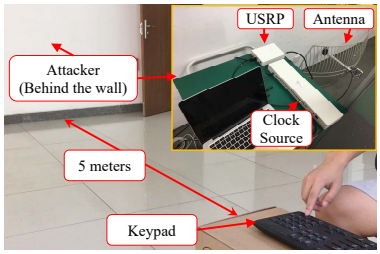
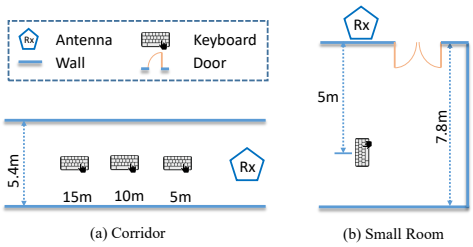


Figure 11. Experimental devices.



(a) Corridor (b) Small Room

Figure 12. Experimental environments.

probability that a given keystroke movement will emit certain observations. As our SVM classifier is not perfect, there are some keystroke states that could observe movements other than the groundtruth movement. With 2800 keystroke samples and their SVM classification results (observations), we build the observation probability matrix as we shown in Figure 8, where a brighter color means higher probability. Note that we do not need to retrain a different HMM for different user or scenarios, given the keyboard layout keeps unchanged.

Key Sequence Recovery: After building the Hidden Markov model, the key sequence recovery can be reduced to the following problem. Given the observation sequence $O = O_1O_2O_3 \cdots O_T$, find the optimal hidden sequence $Q = q_1q_2q_3 \cdots q_T$, *i.e.*, to maximize $P(Q|O, \lambda)$. This problem can be solved by the well-known Viterbi Algorithm that uses a dynamic programming approach [44].

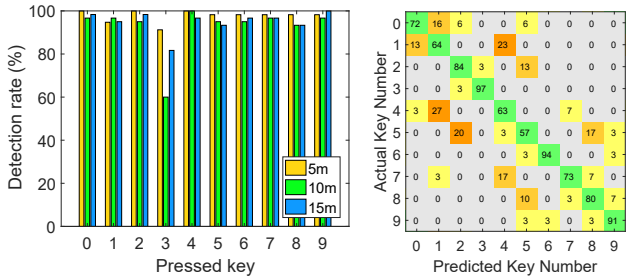
In addition to finding the most probable key sequence, we can also calculate the possibility of all key sequences given the observations. This allows the attacker to sort the candidate keystroke sequences in the descending order by their probabilities. The attacker could then try these sequences one by one and break the password within a few tries.

VI. Implementation and Evaluation

A. Implementation and Evaluation Setup

Implementation: We build SpiderMon on USRP B210 software radios with an external clock source OctoClock CDA-2990, as shown in Figure 11. The total hardware cost of the system, including the workstation, is less than 8,000 US dollars. The LTE signal is transmitted by commercial cellular BSs that are operated by one of the major cellular operators in our region. We select one of the detected BSs with the best signal qualities. The BS used in our experiments has a central frequency of 2330 MHz and a bandwidth of 20 MHz. Keystroke samples are collected in two environments as shown in Figure 12, including a corridor environment for evaluating the operational distance and keyboard orientations and a small room for evaluating the through-wall scenario.

Evaluation Setup: The volunteers are asked to type on the numeric pad of a standard keyboard (Dell Keyboard KB212-B) to simulate PIN inputs on ATM-machines and smart door locks. We perform experiments in two different input modes, one is fixed initial position as assumed by Wikey [10] and WindTalker [12], the volunteers need to return to an initial position after each key press (back to '.' in our experiments), and the other was a natural continuous input of PIN numbers.



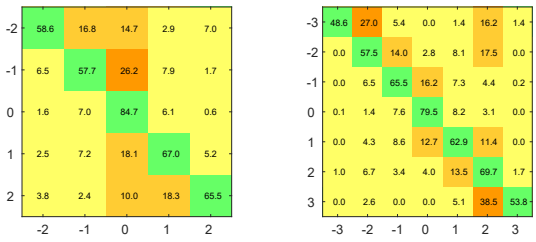
(a) Keystroke detection rate (b) Confusion matrix

Figure 13. Evaluation of keystroke inference: (a) keystroke detection rate; (b) confusion matrix of keystroke recognition with a fixed initial position.

The volunteers are requested to type in these two modes with different randomly generated digit-sequences. We only consider one-handed input, which is the way that most people enter passwords in the numeric keypad area. The volunteers are requested to type with their right hand at a limited input speed (with an interval about $1 \sim 1.5$ seconds between keystrokes). To perform the attack, one volunteer play the role of an attacker, whose data are used for training the SVM classifier (either for directly keystroke recognition in the fixed initial position mode or direction movement classification in the continuous typing mode), another volunteer play the role of victim, whose data are treated as input PIN codes. The amount of training data used in our experiments is 150 keystrokes which can be collected within five minutes. The attacker can pretend to input on the ATM or smart door keyboard to gather training data without any cooperation from the victim.

B. Performance under fixed initial position mode

We first evaluate the accuracy of keystroke detection in the fixed initial position model. We request the volunteers to type each key 60 times at three different distances ($5m$, $10m$, and $15m$ as shown in Figure 12) and count the number of detected keystrokes. Figure 13(a) shows the accuracy of our keystroke detection scheme. At distances of $5m$, $10m$, and $15m$, the average keystroke detection rates are 97.7%, 92.5%, and 95.0%, respectively. We observe that the detection rate for the key '3' is the lowest. This is because the finger moves with the shortest distance from the initial position of the key '.' to the key '3' (only $2cm$ on the numeric keypad). In our results, 51% of the missing keys are due to the key '3' and the missing rates of other keys are much smaller. For the false positive rate, *i.e.*, reporting that a key is pressed when the user is not typing, we count the number of keystrokes mis-detected under a silent environment (without surrounding movements). The false positive rate of detection is 2.38 times per hour.



(a) horizontal (b) vertical

Figure 14. Confusion matrix of SVM classifier of the decoupled horizontal and vertical direction movement distance.

Then we evaluate the recognition performance under fixed initial position mode. Figure 13(b) shows the confusion matrix for key recognition. The average recognition accuracy is 77%, and the recognition accuracy of the key '5' is the lowest (57%). The possible reason could be that the key '5' is located at the center of the numeric keypad so that it has the largest number of adjacent keys.

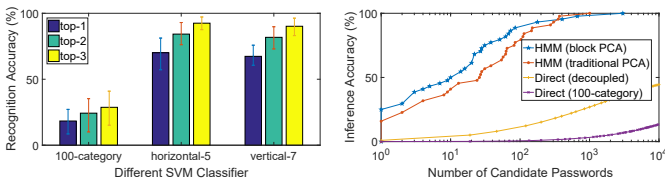
We observe that most of the errors come from adjacent keys. For example, all recognition errors of the key '1' are due to the key '0' and '4'. The key '7' has a 17% probability of being recognized as the key '4' and 7% being recognized as the key '8'. We also noticed that misidentification are more inclined to the vertical key groups like '147', '258' instead of the horizontal key groups, such as '123'. We believe that this is related to the position of the keyboard during our experiments (see Figure 12). Given our keyboard placement, for a keystroke action from the '.' key to the target key, the corresponding path length change in the horizontal direction is more pronounced than in the vertical direction.

In exiting work that use Wi-Fi CSI as side-channel, WindTalker [12] achieves comparable 80% mean accuracy at a distance of $0.75m$, but quickly drops to 40% when the distance is $1.5m$. Wikey [10] only works for scenarios where the AP is within $30cm$. Benefit from GPS-regulated oscillators and low-noise amplifiers used in commercial cellular BSs, our LTE-based approach can operate in a distance of $5 \sim 15m$.

C. Performance under continuous typing mode

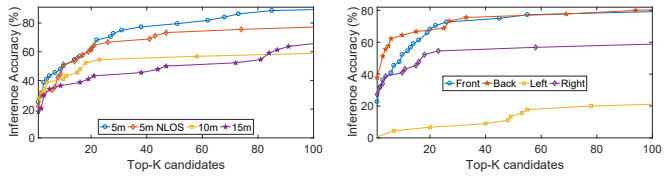
To evaluate the performance of continuous keystrokes, we first evaluate the performance of the keystroke movement SVM classifier for two different approaches: the 100-category SVM that directly estimates the possibility of the 100 possible key pair transitions and the decoupled horizontal-vertical SVM that estimates the movement in the two directions separately. The top-3 classification accuracies for different approaches are showed in Figure 15(a). We observe that the performance of the 100-category SVM is quite poor due to the much larger number of categories to be classified when compared to the decoupled SVM. For the 100-category classifier, the top-3 accuracy is less than 30% and the top-50 accuracy is still less than 90%.

We evaluate the continuous keystroke sequence inference performance as follows. For each test keystroke waveform of 6 digits, we calculate the probabilities for all possible 6-digit sequences with the HMM method. We sort the candidate key



(a) SVM classifiers performance (b) Password inference

Figure 15. Keystroke recognition performance with arbitrary initial positions: (a) SVM classifier performance for 100-category and decoupled directions SVM; (b) Password inference accuracy with different methods.



(a) Impact of different distances (b) Impact of different orientations

Figure 16. Password inference accuracy under the impact of different directions, distances, and victims.

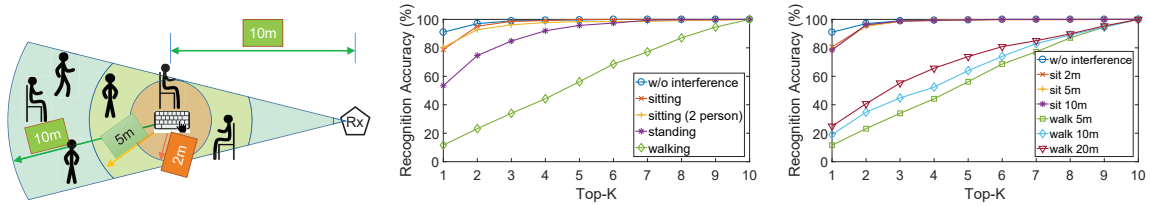
sequences based on their probability in the decreasing order and report the probability that the ground truth sequence is in the top-K candidates. For example, a top-K accuracy of 50% indicates that 50% of the true PIN codes can be found in the first K candidate sequences. Figure 15(b) shows the top-1 accuracy is 25.0%, top-10 accuracy is 54.5% for the HMM-based inference. The top-1 accuracy when directly using the output of the 100-category SVM classifier is less than 2%. We also consider the method that directly uses the horizontal and vertical SVM result to calculate key sequences probabilities. As shown with the yellow line, to achieve a success rate of 25%, the attacker may need 790 trials using the direct probability calculation without HMM.

D. Performance under different scenarios

We conducted keystroke recognition experiments in different environments, to see the impact of different distances, NLOS scenario, keyboard orientations, and different victims.

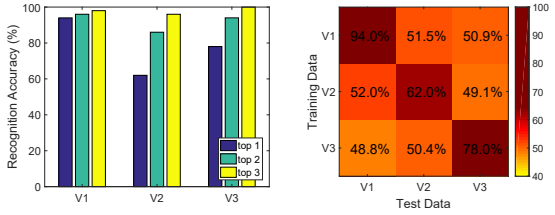
Impact of Distance and NLOS: We first evaluate the system performance when the victim was at different distances to the receiving antenna. Figure 16(a) shows the top-K password inference accuracy under a distance of $5m$, $10m$, $15m$, and an NLOS scenario (where the attack devices are blocked by a $21cm$ thick concrete wall) as shown in Figure 12. In a distance of $5m$, we can recover a 6-digits password with over 87% probability within 100 trials. Even at a distance of $15m$, SpiderMon can still achieve 36% accuracy in ten trials and over 60% in 100 trials. Because of the good penetration of LTE signals, our system can achieve 51% accuracy in ten trials in the NLOS environment with a distance of $5m$.

Impact of Keyboard Orientation: The relative direction between the victim and the attacker has serious impacts on the performance of our system, as different directions will induce different multi-path environments. We evaluate the performance of SpiderMon by placing the keyboard in four different directions (at a distance of 10 meters) so that the receiving antenna was pointed to the left, right, front, and



(a) Non-victims' interferences (b) Interference of different motions (c) Interference at different distances

Figure 17. Keystroke recognition top-k accuracy with different levels of interference from non-victims.



(a) Different victims Top-K accuracy (b) Different training sets accuracy

Figure 18. Keystroke recognition with different victims and the training set.

back of the victim. The volunteers always typed with their right hand and the keyboard was always placed in the right front of the volunteer during the experiments. From Figure 16(b) we can observe that the performance of SpiderMon is consistent for the front, back, and right orientations, while the performance on the left is considerably worse. This could be caused by the occlusion of the typing hand (right hand) by the victim's moving body when viewed from the left.

Impact of Different Victims: We evaluate the impact of different typing styles with three volunteers as the victims. The evaluation is based on the *single keystroke* setup. In Figure 18(a), we show the keystroke recognition accuracy of the three participants when training by his/her own data, where V1, V2, and V3 represent three different victims. We observe that while the top-1 accuracies for the three victims are different, *i.e.*, 94%, 62%, and 78%, all victims' top-3 accuracies are over 95%. We further evaluated the performance when the training and testing data are from different victims (one victim's data as the training set and another victim's as the testing set). The top-1 results of the accuracy are shown in Figure 18(b). In Figure 18(b), the digits in each grid mean the top-1 accuracy when the testing data is from V_a and the training data is from V_b, and the diagonal data represents the accuracy of using his/her own data with 10-fold cross-validation. We observe that when using different people's data for training, the accuracy is significantly reduced. For example, for V1, when the training data is from V2 and V3, the accuracy drops from 94.0% to 52.0% and 48.8%. However, we believe this problem can be alleviated by collecting more people's keystroke data and training with a more powerful machine learning algorithm that is less sensitive to the variance of users, *e.g.*, with a GAN [45].

Non-victims' interference: To evaluate the performance when other non-victims are in the target area, we conducted two sets of *single keystroke* recognition experiments concerning the interference of different movement intensities and different interfere distances. A volunteer plays the role of a victim to perform keystrokes at a distance of 10 meters away from the receive antenna, other volunteers are treated as non-victims in the target area. An illustration of the experiment is shown in Figure 17(a).

In the first experiment, non-victims were requested to perform different movements within 5 meters of the victim, including sitting, standing, and walking. Figure 17(b) shows the top-K accuracy of the keystroke recognition under the above interferences. We observe that as the intensity of non-victims' actions increase, the recognition accuracy decreases significantly. It is worth noting that: first, there is no significant impact on the accuracy of recognition when someone is sitting still, even if there are multiple non-victims around. Second, the standing posture has more significant influence on the performance than the sitting posture, because humans move the body involuntarily even when standing still. Third, the impact of walking on the signal is so significant that the keystroke action is completely submerged.

In the second experiment, non-victims were asked to maintain the sitting or walking state within different distances from the victim. The impact of these interferences are shown in Figure 17(c). We observe that a sitting person has nearly no effect on keystroke recognition, even if it is within 2 meters of the victim. The walking action, even at a distance of 20 meters, still has an intensity higher than the keystroke action, the top-1 accuracy rate is only about 25%, and the top-3 accuracy rate is less than 60%, barely better than a random guess.

VII. Conclusions

In this paper, we show that LTE reference signals can be used as a medium for side-channel attacks by implementing the SpiderMon system that displays and analyzes LTE CRS signals in real-time. Compared to previous attacks that use Wi-Fi CSI, LTE-based attacks can achieve comparable performance while have a longer operational distance and do not need active transmissions. Therefore, LTE-based attacks are harder to be detected and lead to more serious security breaches. We hope that our work could inspire more research in this area to protect users from such attacks.

Acknowledgment

We would like to thank our anonymous shepherd and reviewers for their valuable comments. This work is partially supported by National Natural Science Foundation of China under Numbers 61872173, 61872174, 61972192, and Collaborative Innovation Center of Novel Software Technology.

REFERENCES

[1] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha, "Beware, your hands reveal your secrets!," in *Proceedings of ACM CCS*, pp. 904–917, ACM, 2014.

[2] J. Sun, X. Jin, Y. Chen, J. Zhang, Y. Zhang, and R. Zhang, "VISIBLE: Video-assisted keystroke inference from tablet backside motion.," in *NDSS*, 2016.

[3] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "Accessory: password inference using accelerometers on smartphones," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, p. 9, ACM, 2012.

[4] X. Liu, Z. Zhou, W. Diao, Z. Li, and K. Zhang, "When good becomes evil: Keystroke inference with smartwatch," in *Proceedings of ACM CCS*, 2015.

[5] J. Liu, Y. Wang, G. Kar, Y. Chen, J. Yang, and M. Gruteser, "Snooping keystrokes with mm-level audio ranging on a single phone," in *Proceedings of ACM MobiCom*, 2015.

[6] T. Zhu, Q. Ma, S. Zhang, and Y. Liu, "Context-free attacks using keyboard acoustic emanations," in *Proceedings of ACM CCS*, 2014.

[7] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 1, p. 3, 2009.

[8] D. Asonov and R. Agrawal, "Keyboard acoustic emanations," in *IEEE Symposium on Security and Privacy*, 2004.

[9] J. Liu, C. Wang, Y. Chen, and N. Saxena, "Vibwrite: Towards finger-input authentication on ubiquitous surfaces via physical vibration," in *Proceedings of ACM CCS*, 2017.

[10] K. Ali, A. X. Liu, W. Wang, and M. Shahzad, "Keystroke recognition using WiFi signals," in *Proceedings of ACM MobiCom*, 2015.

[11] B. Chen, V. Yenamandra, and K. Srinivasan, "Tracking keystrokes using wireless signals," in *Proceedings of ACM MobiSys*, 2015.

[12] M. Li, Y. Meng, J. Liu, H. Zhu, X. Liang, Y. Liu, and N. Ruan, "When CSI meets public WiFi: Inferring your mobile phone password via WiFi signals," in *Proceedings of ACM CCS*, 2016.

[13] J. Zhang, X. Zheng, Z. Tang, T. Xing, X. Chen, D. Fang, R. Li, X. Gong, and F. Chen, "Privacy leakage in mobile sensing: Your unlock passwords can be leaked through wireless hotspot functionality," *Mobile Information Systems*, vol. 2016, 2016.

[14] L. Chiaraviglio, F. Cuomo, M. Maisto, A. Gigli, J. Lorincz, Y. Zhou, Z. Zhao, C. Qi, and H. Zhang, "What is the best spatial distribution to model base station density? a deep dive into two european mobile networks," *IEEE Access*, vol. 4, pp. 1434–1443, 2016.

[15] M. Y. Arslan, J. Yoon, K. Sundaresan, S. V. Krishnamurthy, and S. Banerjee, "FERMI: a femtocell resource management system for-interference mitigation in OFDMA networks," in *Proceedings of ACM MobiCom*, 2011.

[16] R. Baldemair, T. Irnich, K. Balachandran, E. Dahlman, G. Mildh, Y. Selén, S. Parkvall, M. Meyer, and A. Osseiran, "Ultra-dense networks in millimeter-wave frequencies," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 202–208, 2015.

[17] Y. Qiao, O. Zhang, W. Zhou, K. Srinivasan, and A. Arora, "Phycloak: Obfuscating sensing from communication signals," in *Proceedings of Usenix NSDI*, 2016.

[18] F. Lu, H. Du, A. Jain, G. M. Voelker, A. C. Snoeren, and A. Terzis, "CQIC: Revisiting cross-layer congestion control for cellular networks," in *Proceedings of ACM HotMobile*, 2015.

[19] X. Xie, X. Zhang, and S. Zhu, "Accelerating mobile web loading using cellular link information," in *Proceedings of ACM MobiSys*, 2017.

[20] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "pistream: Physical layer informed adaptive video streaming over LTE," in *Proceedings of ACM MobiCom*, 2015.

[21] S. Kumar, E. Hamed, D. Katabi, and L. Erran Li, "LTE radio analytics made easy and accessible," in *Proceedings of ACM SIGCOMM*, 2014.

[22] Y. Li, J. Fang, K. Tan, J. Zhang, Q. Cui, and X. Tao, "Soft-LTE: A software radio implementation of 3GPP long term evolution based on Sora platform," in *Proceedings of ACM MobiCom (Demo)*, 2009.

[23] M. Driusso, C. Marshall, M. Sabathy, F. Knutti, H. Mathis, and F. Babich, "Vehicular position tracking using lte signals," *IEEE Trans. Vehicular Technology*, vol. 66, no. 4, pp. 3376–3391, 2017.

[24] D. Vasisht, S. Kumar, H. Rahul, and D. Katabi, "Eliminating channel feedback in next-generation cellular networks," in *Proceedings of ACM SIGCOMM*, 2016.

[25] Q. Pu, S. Gupta, S. Gollakota, and S. Patel, "Whole-home gesture recognition using wireless signals," in *Proceedings of ACM MobiCom*, 2013.

[26] Y. Wang, J. Liu, Y. Chen, M. Gruteser, J. Yang, and H. Liu, "E-eyes: In-home device-free activity identification using fine-grained WiFi signatures," in *Proceedings of ACM MobiCom*, 2014.

[27] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, "Understanding and modeling of WiFi signal based human activity recognition," in *Proceedings of ACM MobiCom*, 2015.

[28] G. Wang, Y. Zou, Z. Zhou, K. Wu, and L. M. Ni, "We can hear you with Wi-Fi!," in *Proceedings of ACM MobiCom*, 2014.

[29] F. Adib and D. Katabi, "See through walls with WiFi!," in *Proceedings of ACM SIGCOMM*, 2013.

[30] F. Adib, Z. Kabelac, and D. Katabi, "Multi-person motion tracking via RF body reflections," in *Proceedings of Usenix NSDI*, 2015.

[31] J. Lien, N. Gillian, M. E. Karagozler, P. Amihood, C. Schwesig, E. Olson, H. Raja, and I. Poupyrev, "Soli: ubiquitous gesture sensing with millimeter wave radar," *ACM Transactions on Graphics*, vol. 35, no. 4, p. 142, 2016.

[32] T. Wei and X. Zhang, "mTrack: High-precision passive tracking using millimeter wave radios," in *Proceedings of ACM MobiCom*, 2015.

[33] L. Yang, Q. Lin, X. Li, T. Liu, and Y. Liu, "See through walls with COTS RFID system!," in *Proceedings of ACM MobiCom*, 2015.

[34] C. Wang, J. Liu, Y. Chen, H. Liu, L. Xie, W. Wang, B. He, and S. Lu, "Multi-touch in the air: Device-free finger tracking and gesture recognition via cots rfid," in *Proceedings of IEEE INFOCOM*, 2018.

[35] D. K. P. Tan, M. Lesturgie, H. Sun, W. Li, and Y. Lu, "GSM based through-the-wall passive radar demonstrator for motion sensing," in *Proceedings of IEEE New Trends for Environmental Monitoring Using Passive Systems*, 2008.

[36] Q. Yue, Z. Ling, X. Fu, B. Liu, K. Ren, and W. Zhao, "Blind recognition of touched keys on mobile devices," in *Proceedings of ACM CCS*, 2014.

[37] Y. Berger, A. Wool, and A. Yeredor, "Dictionary attacks using keyboard acoustic emanations," in *Proceedings of ACM CCS*, pp. 245–254, ACM, 2006.

[38] Y. Yao, Y. Li, X. Liu, Z. Chi, W. Wang, T. Xie, and T. Zhu, "Aegis: An interference-negligible RF sensing shield," in *Proceedings of IEEE INFOCOM*, 2018.

[39] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation*. 3GPP LTE TS 36.211, 2015.

[40] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

[41] H. Li, W. Yang, J. Wang, Y. Xu, and L. Huang, "WiFinger: talk to your smart devices with finger-grained gesture," in *Proceedings of ACM UbiComp*, 2016.

[42] D. Vasisht, S. Kumar, and D. Katabi, "Decimeter-level localization with a single WiFi access point," in *Proceedings of Usenix NSDI*, 2016.

[43] Y. Xie, Z. Li, and M. Li, "Precise power delay profiling with commodity WiFi," in *Proceedings of ACM MobiCom*, 2015.

[44] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.