XR Devices Send WiFi Packets When They Should Not: Cross-Building Keylogging Attacks via Non-Cooperative Wireless Sensing

Christopher Vattheuer Justin Feng Hossein Khalili Nader Sehatbakhsh Omid Abari
University of California, Los Angeles (UCLA)

***Abstract*—As Extended Reality (XR) technology continues to integrate into diverse fields, various security vulnerabilities—such as keystroke inference (keylogging)—have become a growing concern. Several keylogging attacks demonstrate the feasibility of exploiting this vulnerability using different modalities, including voice and vision. However, these attacks are often constrained by the need for line of sight (LoS) and/or close proximity (<10 meters). We propose a novel keylogging attack on XR devices leveraging WiFi wireless sensing. Unlike prior methods, our attack does not require LoS and is effective across various scenarios, including long-distance, cross-building settings (up to 30 meters). Our attack requires only a single, cheap, pocket-sized receiving setup to collect the victim's WiFi packets. Compared to previous keylogging attacks leveraging WiFi, our approach is the first to eliminate the need for a separate transmitter and receiver or a fake hotspot. As a result, unlike prior methods, our attack is effective even at large distances. The core idea hinges on exploiting a security vulnerability in WiFi chipsets. This vulnerability allows an attacker to send a fake, unencrypted packet to the victim's device, where, in response, the victim's device involuntarily and automatically transmits an acknowledgment ("ACK") packet. By leveraging this mechanism, we can continuously force the headset's WiFi chipset to transmit packets and therefore harvest large volumes of Channel State Information (CSI) data from the victim's headset. We then develop a novel unsupervised signal processing algorithm to exploit CSI data to perform pose estimation and locate the victim's hands and fingers, ultimately enabling keystroke inference. We evaluate our attack on *Meta Quest 2* and *Meta Quest 3* [1], [2] headsets under diverse conditions, including distances ranging from 1 meter to 30 meters, angles spanning from -90° to +90°, multiple users, and through-wall scenarios, demonstrating its robustness and effectiveness across a wide range of environments. Our attack achieves 78.6% top-25 accuracy across a building on passwords up to 15 characters long.**

I. Introduction

Virtual and Augmented Reality, collectively known as Mixed/Extended Reality (XR), are transformative technologies that immerse users in customizable, interactive, and simulated environments. These advancements have revolutionized numerous sectors, making XR devices increasingly prevalent in many personal and public spaces [3], [4], [5], [6], [7], [8], [9].

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240926
www.ndss-symposium.org

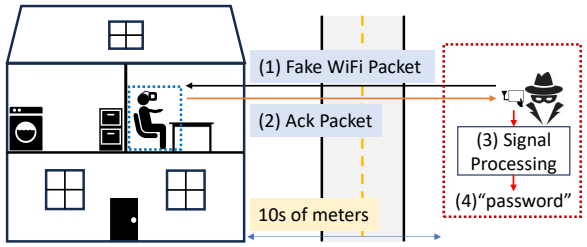

Fig. 1: High-Level Attack Overview. The adversary sends a fake WiFi packet that the user involuntarily acknowledges. The adversary applies signal processing to reveal what was typed. The benefit of TwiST is extended range and NLOS scenarios.

As a result, these devices are increasingly gaining access to sensitive data, such as passwords and PIN codes, and are being used to control security-critical applications, raising significant privacy and security concerns.

Similar to prevailing computing models, XR devices are vulnerable to a wide range of software, network, mobile, and sensor-based attacks. In fact, an extensive array of attacks has already been launched against various aspects of XR devices [10], [11], [12], [13], [14], [15], [16]. This emerging category of attacks is highly domain-specific, exploiting unique characteristics of XR devices, such as the seamless integration of sensors and computational capabilities, as well as their use of diverse sensor modalities.

Among the various attack schemes, ***external physical attacks*** are becoming increasingly prevalent. The key concept behind this class of attacks is the extraction of sensitive information from the device—such as passwords—primarily through keystroke inference (keylogging) using physical side channels. These attacks often involve leveraging physical signals, such as using a microphone to eavesdrop on the clicking sounds generated by the victim's hand controller during keystrokes [17]. Another example includes analyzing the victim's hand movement via a (spy) camera [18]. A key characteristic of these external attacks is that they do not require the attacker to infiltrate the XR device or install any application on it. The attack is entirely independent of the device, making it a non-invasive, highly stealthy, and powerful approach to extracting sensitive information.

Despite their success, the common limitation on prior external XR attacks [18], [17], [14], [13], [19], [20], [21],

[22], [16], [23], [24], [25] is that they operate in short distances (less than 10 meters) and often require line of sight (LOS) for measuring the physical signal. This limitation in range is especially seen in prior CSI-based XR attacks, which fail at larger distances due to their attack setup. Furthermore, these attacks have limited generalizability across environments due to their choice of algorithm, which ultimately requires attackers to physically modify and collect training data in a victim's environment before an attack.

Why do previous CSI-based setups fail at further distances? To be accurately sensed using CSI in an environment with mobility, users must be in close proximity to a transmitter (TX) or receiver (RX) used in sensing. Prior methods, such as VRSpy [19] use an attacker-owned TX and RX pair, and as a result, they must place attacker equipment next to a user to have a high signal-to-noise ratio (SNR) and accurate predictions. This requirement for proximity is due to the Near Field Domination Effect [26], explained later in Section II.
To overcome this limitation, we use only an attacker-owned RX, facilitated by forcing the victim XR headset to play an active role as the TX in sensing its user. Given that the user wears the headset, this ensures that a TX is always very close to a user, guaranteeing high SNR even at long range. Our approach enables significantly more robustness to motion in the environment. Further comparisons and theoretical background are elucidated in Section II.

To force a user headset to act as a TX, we exploit an existing vulnerability in *all* WiFi chipsets, called Polite WiFi vulnerability [27], to extract a large corpus of CSI data (180-220 packets per second) from the WiFi signals of a user's headset. This vulnerability allows an attacker to send a fake, unencrypted packet to a specific victim device where, in response, the victim's device involuntarily and automatically transmits an acknowledgment ("ACK") packet. The victim device that the attacker sends a packet to will be the only device to send a response back, and will respond even if it is connected to a password-protected network that the attacker is not on. As a result, the attacker can continuously send fake packets and receive CSI information through received "ACK" packets. This eliminates limitations in prior work on exploiting WiFi signals for keylogging. This makes the headset's WiFi device an ideal transmitter for the WiFi sensing attack since it is naturally very close to the body of the victim. Furthermore, our approach does not need the victim device to be on the same network as the attacker nor requires the devices to be connected to a fake and/or attacker-controlled access point [28], [29], [21].

Why do prior CSI XR sensing methods fail to generalize environments? Prior CSI-based sensing XR attacks rely only on CSI data and machine learning models trained on real typing data to make predictions. However, because signal propagation characteristics vary significantly across different environments, models are typically trained and evaluated within the same setting. This requires collecting new training data for each deployment, making such attacks difficult to perform and unscalable.
To overcome this limitation, we propose a novel signal processing scheme called Transition Web in Spring Tension (TwiST). TwiST operates on the fly and can infer movements across users and environments by relying on properties of movement within a constrained region, providing a one-size-fits-all approach to wireless sensing. Notably, it achieves this using only a single receiver equipped with a single antenna.

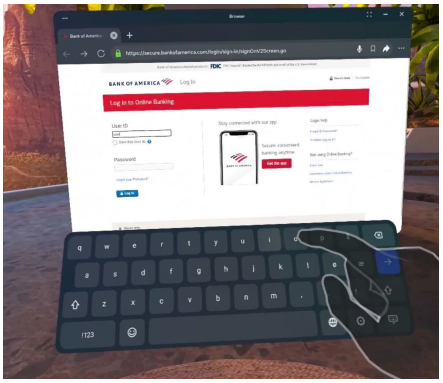

Fig. 2: Typing Passwords in VR. When logging into Bank of America, a virtual touch keyboard that is approximately 40cm wide is presented to the VR User. This keyboard can be interacted with by physically reaching out and touching it.

An overview of our attack is shown in Figure 6. Our attack is simple and can be performed using inexpensive hardware and limited computational resources. We evaluate it under different conditions and scenarios, including line-of-sight, cross-building (up to 30 meters), and through-the-wall. Our attack achieves 78.6% top-25 accuracy across a building on passwords up to 15 characters. This paper makes the following contributions:

- We propose the first XR Keylogging attack, which works in long-range and Non-Line-of-Sight (NLOS) scenarios.
- We show how an attacker can turn the headset into a transmitter for a WiFi-based keylogging attack by forcing the headset to continuously transmit WiFi packets without having any access to the headset or its network.
- We introduce a novel unsupervised wireless-based sensing method capable of capturing the motion of users across environments with no training data.
- We perform extensive evaluation in several LOS and NLOS scenarios. We will open-source our implementation and experimental data.

II. Background

A. Virtual Reality Keyboards

Virtual reality headsets are used across various domains, many of which increasingly require frequent text input, as shown in Figure 2. These tasks include web browsing, signing into accounts, entering passwords or PINs, sending messages, and making purchases. In current XR systems, text entry is supported through several interaction styles. Most platforms provide virtual keyboards that users operate either with hand

tracking or with controllers via raycast selection [30], [2], [31]. Emerging devices such as the Apple Vision Pro [31] also support gaze-based input. Our work focuses on the widely deployed hand tracking input method.

Headset-based hand tracking for typing is seamless on the Meta Quest 2, which achieves an average positional error of only 1.1 cm [32]. The keys on virtual keyboards are significantly larger than on physical keyboards, typically around 3cm wide, to accommodate user motion and tracking error. By tracking the user's head, these keyboards remain fixed in position during typing, simplifying alignment and use. VR headsets are data-intensive and rely on Wi-Fi as their primary method of transmitting and receiving data. The virtual keyboard in VR is approximately 40cm wide and contains around 39–42 keys, each about 3cm in width. As technology improves, adoption of XR devices continues to grow, especially in public settings. While this paper focuses on the Meta Quest 2, the vulnerabilities described apply broadly to XR wearable devices that use floating virtual keyboards.

B. WiFi-Protocol

WiFi is a widely used wireless communication protocol that enables devices to exchange data over a network using the IEEE 802.11 standard. Each device in a WiFi network is uniquely identified by its Media Access Control (MAC) address, a 48-bit identifier. The MAC address ensures that devices can distinguish themselves on the network, enabling data to be directed to the correct recipient. MAC addresses are not encrypted as they are essential for packet routing and identifying devices in a network. Encrypting MAC addresses would prevent packets from being properly forwarded, so they must remain exposed. This allows attackers to sniff and identify devices even in networks they are not connected to.

WiFi communication relies on a structured packet format that includes a header, payload, and trailer. The header contains metadata, including the sender's and receiver's MAC addresses, which allow the recipient device to recognize packets meant for it and enable intermediate devices, such as access points, to forward the packet to the appropriate destination.

The payload is the main body of the packet and contains the actual data being transmitted. In secure WiFi protocols like WPA2 or WPA3, the payload is encrypted to ensure confidentiality. The header of packets is not encrypted, leaving the details of the header visible to those outside of the network.

The trailer includes the Frame Check Sequence (FCS), a 32-bit value used for error detection. The FCS is computed using a Cyclic Redundancy Check (CRC) algorithm and allows the receiver to verify the integrity of the packet. This structured format ensures proper routing, reliable delivery, and robust error detection within a wireless network.

WiFi communication includes mechanisms such as acknowledgments (ACKs) to ensure reliable data transmission. When a device sends a packet, the receiving device sends an acknowledgment back only if the packet is received without errors, as verified by the FCS. If the sender does not receive an ACK within a brief window, it assumes the packet was not successfully received and retransmits it. This acknowledgment system provides immediate feedback and helps mitigate the effects of interference and packet loss, both of which are common in wireless environments.

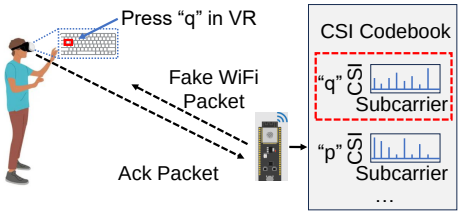

Fig. 3: In CSI sensing, different motions have different effects on CSI. Pressing ”q” corresponds to a different CSI vs. Carrier response compared to pressing ”p”.

While critical for ensuring reliable communication, this acknowledgment mechanism has highly limited security. Acknowledgments will be sent for any received packet where a device is the intended receiver and the FCS passes. This allows for other devices, even those outside an open or even a password-protected network, to spoof packets to a target device, prompting ACKs from any device. Prior work has demonstrated that this behavior (known as Polite WiFi) exists in all WiFi chipsets[27]. This behavior exists as acknowledgments must be sent immediately after the FCS passes, as it significantly improves the throughput of a network because performing an FCS check is significantly faster than validating the integrity of a packet's encrypted payload.

C. Wireless-Sensing

In WiFi communication, packets are transmitted wirelessly between devices acting as transmitters and receivers. The transmitted signals, denoted by x_{sent}, are complex numbers characterized by an amplitude A and a phase ψ ($x_{\text{sent}} = Ae^{j\psi}$). These amplitude and phase components are carefully modulated to encode data.

WiFi uses a technique called Orthogonal Frequency-Division Multiplexing (OFDM), where data is transmitted across different frequencies called subcarriers. As signals propagate through the environment, they undergo distortions caused by factors such as path loss, fading, and multipath reflections. In OFDM systems, these distortions are collectively represented by the channel h, where $x_{\text{received}}(t) = x_{\text{sent}}(t) * h(\tau, t) + \epsilon(t)$, for propagation delay τ and noise $\epsilon(t)$. Considering that the noise is negligible, the channel in the frequency domain can be presented as $H(f) = \frac{X_{\text{received}}(f)}{X_{\text{sent}}(f)}$.

Channel State Information (CSI) represents the channel response for each subcarrier, providing a detailed view of how the wireless channel affects transmitted signals. It is typically represented as $H(k,n) = |H(k,n)|e^{j\phi(k,n)}$. $|H(k,n)|$ denotes the amplitude response, and $\phi(k,n)$ denotes the phase response of the channel at subcarrier k and time index n.

CSI serves as a descriptor of the wireless environment across the subcarriers, capturing the way signals interact with objects and materials in their path. By analyzing the amplitude and phase changes across subcarriers, CSI can

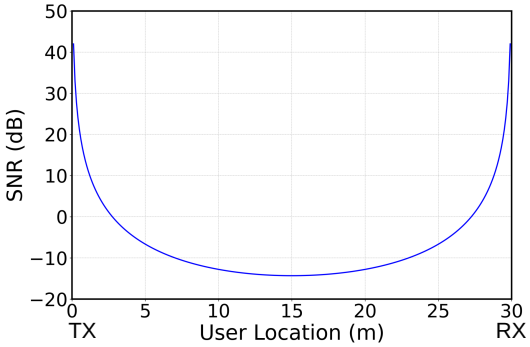

Fig. 4: Signal of interest's SNR versus user's position between a TX and RX spaced 30m. SNR is the ratio of the power of channel variations caused by the user to variations caused by other environmental mobilities. High SNR occurs when a user is in close proximity to a TX or RX.

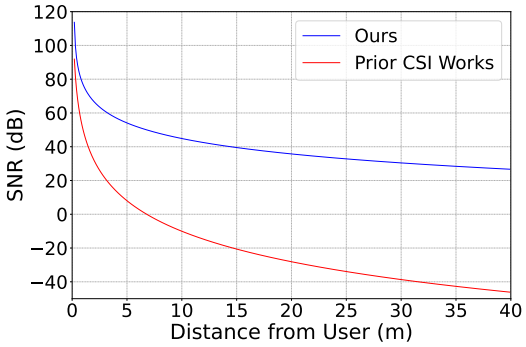

Fig. 5: SNR of the signal of interest versus distance of victim from attacker equipment for both TwiST and Prior CSI Work. Effective attacks require SNR of at least 5-10 dB. TwiST enables much higher SNR (and longer range) since it utilizes a user's XR headset as a TX, which is very close to the user, whereas prior CSI-based works utilize an attacker-owned TX and RX which are spaced far from the victim.

reveal spatial and temporal variations in the environment. This makes CSI highly suitable for sensing tasks, such as motion detection, activity recognition, and localization [33], [34], [35].

In CSI based sensing, the Near-Field Domination Effect (NFDE) [26], relates variations in the channel caused by an object to its velocity (v_S), its distance to the Tx (d_{Tx}), its distance to the user (d_{Rx}) and the path loss (α):

$$\left|\frac{\partial H(k,n)}{\partial n}\right|^2 \propto v_S^2 (d_{Tx} d_{Rx})^{-\alpha}$$

NFDE shows that objects that move faster or are closer to a TX or RX have a greater impact on the variations in the channel. Thus, as a user gets closer to a TX or RX, their contribution to channel variations increases significantly, making it easier to sense even small motions. This property applies to all mobility within an environment, where moving objects unrelated to a victim work to obscure the victim's motion.

It is for this reason that accurate sensing necessitates users to remain close to a TX or RX. Figure 4 shows the SNR of a user's hand motions given their placement between a TX-RX pair spaced 30m apart. SNR is calculated by taking the power of channel variations due to hand movements compared to variations caused by environmental mobilities. By ensuring a user is always close to a TX, the channel variations of small hand movements remain more significant relative to environmental noise even at long range.

III. Related Work

We compare some of the state-of-the-art work in software-based and external extended reality attacks. We compare each attack, along with our own work, in Table III.

A. Software-Based Extended Reality Attacks

Some works use app software to analyze user data, such as head orientation, to extract keystrokes [15], [13], [10], [12]. These attacks may be conducted via malware, a background app, a malicious website [11], or game [36], enabling remote attacks. The key idea is that permissions are not required to access XR sensor data [14]. However, these approaches have several limitations. First, they require access to the device or the user to download malware or be connected to a specific multiplayer application. Second, they rely on the absence of permission controls for sensor data. TwiST improves on these attacks by requiring no user action or physical access to the device and by leveraging properties inherent to wireless protocols.

VRSpy is a prior CSI-Based Tx-Rx Keystroke Inference Method for VR devices [19]. While VRSpy achieves high accuracy in its evaluation, it fails to generalize environments and does not work at long range. VRSpy does not involve user equipment in its sensing paradigm, instead performing all evaluations with a user typing within a 4-foot gap between an attacker-owned TX and RX pair. Requiring physical access to an environment to strategically position a Tx/Rx where a victim will later use XR significantly constrains the feasibility of real-world attacks; Such an approach requires extensive planning and coordination to determine where a victim will situate themselves to covertly transform that region into a controlled wireless sensing environment. Figure 5 shows a comparison of SNR for user motions given the distance from a user to the nearest attacker-owned equipment. The SNR of hand motions in TwiST remains much more robust to distance and environmental noise than VRSpy due to our involvement of user equipment in wireless sensing.
Furthermore, VRSpy does not create environment-invariant features for typing in VR, meaning that training data must be collected for every new environment VRSpy is evaluated in. This again requires an attacker to access a victim's environment to collect a corpus of typing data to later launch an attack. In contrast, our method does not require any training data and runs on the fly by generating generalizable features

from CSI data.

B. External Keystroke Attacks

Prior work demonstrates that keystroke inference is broadly feasible across many sensing modalities and non-XR devices. Acoustic side-channel attacks recover individual keys and even full sentences typed on physical keyboards by analyzing keystroke sound signatures [37], [38]. Wireless sensing has also been used: CSI-based attacks infer keystrokes on mechanical keyboards [20], and similar RF features have been leveraged to recover smartphone passwords through CSI or Beamforming Feedback Information [21], [28]. Additional work shows that wireless keyboards leak information through packet-timing and RSS-based fluctuations [39]. Despite their diversity, these methods share important constraints. They target physical or mobile keyboards rather than VR or XR input systems, typically require user or environment-specific calibration, and generally operate only at short distances. This limits their practicality in immersive settings where input is often unconstrained. In contrast, TwiST requires no training data, generalizes across users and environments, and functions over substantially longer ranges.

C. Attacks on XR Devices

There are various hardware-based attacks beyond analyzing the network. For example, some attacks leverage sensors (camera, accelerometer, etc.) to reveal information [14], [13], [18]. More recent XR-focused work has explored alternative sensing channels, such as infrared (IR) arrays [40], but these methods require dedicated IR hardware, careful physical placement, and maintain effectiveness only within short line-of-sight ranges (roughly 2–4 m). Their reliance on specialized equipment and close proximity make it difficult to deploy in realistic adversarial settings, especially compared to passive CSI approaches.

Cameras are a powerful tool because they allow an adversary to estimate pose and perform hand tracking, which can be used to infer typing. Camera-based approaches have several limitations. First, line of sight is required between the attacker and the victim. Second, performance degrades in harsh environments (for example, low light). TwiST improves on camera-based approaches by enabling attacks even through walls. We also measure network activity, which is orthogonal to lighting conditions. Sound-based attacks are another option [17]; typing clicks can reveal key presses, but these attacks still fail across rooms or buildings, unlike TwiST.

IV. Threat Model

The goal of our attack is to reveal what an XR user is finger-typing on a virtual keyboard by analyzing CSI data extracted through wireless sensing. The adversary uses commercial off-the-shelf hardware to measure WiFi signals. Via a signal processing pipeline, the adversary can then extract sensitive key press information.

In the attack, the victim is using an XR headset and is interacting with the headset by pressing virtual keys. The victim may need to enter sensitive information, such as passwords, as part of this interaction. To use the keyboard, the victim uses physical pressing motions to interact with the virtual keyboard. Such key presses are desired to be detected by the adversary. The victim headset does not need to be on the same network as any of the adversary's devices and does not intentionally transmit data to the adversary.

Other devices may be situated in the environment and have their own impact on the wireless channel. They may be transmitting their own data for purposes related or unrelated to the victim. Furthermore, the environment may have objects, barriers, and other structures that impede the line of sight view between the adversary and the victim. Other people may be present nearby in the environment. The attack is intended to work in environments typical of XR use cases, such as homes, office spaces, and labs.

The adversary may be situated within or outside the victim's environment. The adversary does not have physical access to the victim's headset and cannot tamper with the headset's hardware or software. The adversary can place a sniffer/injector device nearby. These devices may be in the same room, a different room, or even a different building. The injector is able to send fake packets into the environment, stimulating the virtual reality headset to automatically acknowledge them via acknowledgment packets. The sniffer can measure the packets in the environment (including the acknowledgment packets), specifically to measure channel state information and network activity. The sniffer and injector can be either the same physical device and/or two devices placed next to each other. To send and receive packets, the adversary does not need any special privilege and/or rely on any information other than the existence of a "polite WiFi" vulnerability that commonly exists in a wide variety of WiFi chipsets [27], [41]. Finally, the adversary has compute resources that can be used offline.

V. Attack Design

Our attack consists of several phases and modules to address several challenges in realizing this attack. Figure 6 outlines the steps. First, (a) *XR devices are identified* where XR devices are identified by their WiFi packets, then there is a (b) *Online Phase* where the adversary collects wireless information. Next, the (c) *Location Estimation* module and (d) *Press Estimation* module are essential for processing and analyzing the data. Finally, the keys pressed are revealed via the (e) *Keystroke Inference* module. An important aspect of our attack is that our signal analysis scheme, TwiST algorithm, ***does not require any pre-training, re-training, and/or fine-tuning***. This unsupervised adaptability allows it to perform effectively across different environments and configurations without the need for prior calibration or specific setup conditions.

In the remainder of this section, we will describe the TwiST modules and the challenges faced in realizing these modules.

A. Device Identification

The first step in our attack is to find the MAC address of a WiFi chipset. This is a challenging task since many modern

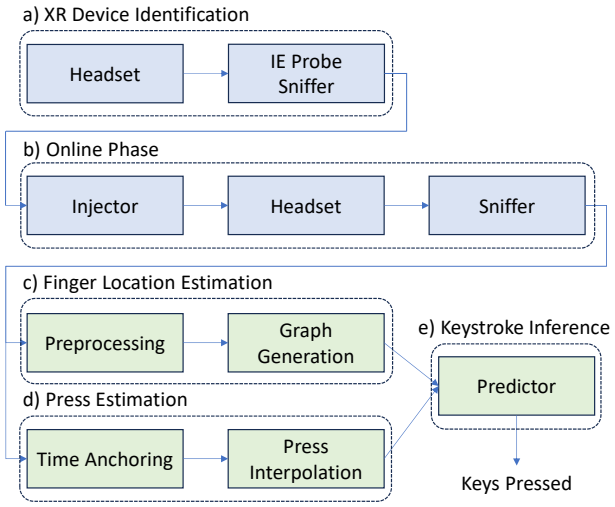

Fig. 6: The attack consists of five major steps. a) XR Devices are identified b) CSI data is collected from the environment. c) The receiver uses CSI data to generate graphs. d) The adversary estimates the time of key presses. e) The adversary infers the keys pressed.

WiFi chipsets perform MAC Address Randomization to limit tracking. This technique replaces a device's real MAC address with a randomized one. Whereas a device's real MAC address can be used to look up a device type, a randomized one cannot. The specifics of MAC address randomization vary across XR device models. The Meta Quest 2 does not implement randomization, making it persistently identifiable across networks. The Meta Quest 3 uses per-network MAC randomization; however, this randomization is done such that each WiFi network it connects to is assigned a consistent randomized address. Finally, the Apple Vision Pro also uses per-network randomization, with an additional layer of privacy where the randomized MAC address rotates, but this rotation is done just every two weeks [42].

Even with MAC randomization and rotation, device types can still be easily identified using findings from recent work [43], [44]. Prior work found that device types can be identified by producing fingerprints on their Information Element (IE) probes. These probes are packets that WiFi devices send out to identify nearby networks, even while already connected to a network, that include detailed information about a device and its capabilities. This MAC Address De-Randomization allows attackers to create fingerprints for XR device models to later identify them in the wild. This approach of producing a fingerprint for an XR device to identify them in the wild is evaluated in Section VI.

B. Online Phase

In the *Online Phase*, the goal of the adversary is to collect relevant information from the environment that may reveal information on what the victim typed. To realize this, several challenges need to be addressed.

First, a consistent and generally high-rate stream of packets from a target device is needed to increase the accuracy of wireless sensing. To facilitate this behavior, the adversary continuously sends well-formed "fake" WiFi packets to the victim's device (injection), forcing it to respond with acknowledgment packets. These acknowledgment packets are measured by the adversary (sniffing). To inject and sniff packets, the adversary first obtains the MAC address of the target headset using tools like Wireshark. It is important to note that the attacker only needs to extract Channel State Information (CSI) data using their own receiving setup. This approach does not require the decryption of packet contents, such as headers or payloads, simplifying the attack process while maintaining its effectiveness.

Second, the attack must be stealthy and not interfere with the existing operation of the victim's device. We found that IP spoofing is not required for the source address of the packets the adversary sends (null packets), as target devices will continue to send acknowledgments to packets that pass the CRC even after de-authentication frames are sent. An issue arises via a property of XR headsets, where they turn off their radio and enter power saving mode at the onset of these null packets. To prevent the headset from entering power-saver mode, fake beacon frames are sent using beacon stuffing. These frames are modified beacon frames that alert devices that they should stay awake, as they have more frames waiting for them. No IP spoofing was required for the false beacon frames. While this attack can be detected via snooping traffic on a network, it has negligible effects on the victim's XR headset, and the network experiences only minor decreases in speed, making it unlikely to trigger an investigation into low-level network traffic.

The final consideration is deciding how many null packets to send to get a sufficient response from the victim. Injecting null data packets and intermittent beacon frames was sufficient for inducing over 180 responses per second from target devices. Many wireless devices, including the Meta Quest 2 and Apple Vision Pro, tested in this work, are susceptible to this vulnerability. As described in Section 2.2, Polite Behavior is integral to WiFi protocol and cannot be disabled on XR devices.

A key advantage of our setup compared to prior approaches is that we force the victim's headset to participate as a transmitter in a wireless sensing scenario. Previous methods rely on a dedicated transmitter to send packets, which are then measured by a receiver for wireless sensing. This approach inherently limits both the measurement distance and situations in which the attack can be utilized, as accurate measurements rely on proximity to a Tx or Rx device.

In contrast, our setup leverages the automatic acknowledgment (ACK) packets sent by the victim's headset in response to crafted 802.11 frames. This allows us to always be capable of precise wireless sensing by turning the victim's headset into an (involuntary) transmitter. Compared to using a separate transmitter, this greatly simplifies the setup for the attack, as the location of the XR user no longer needs to be physically

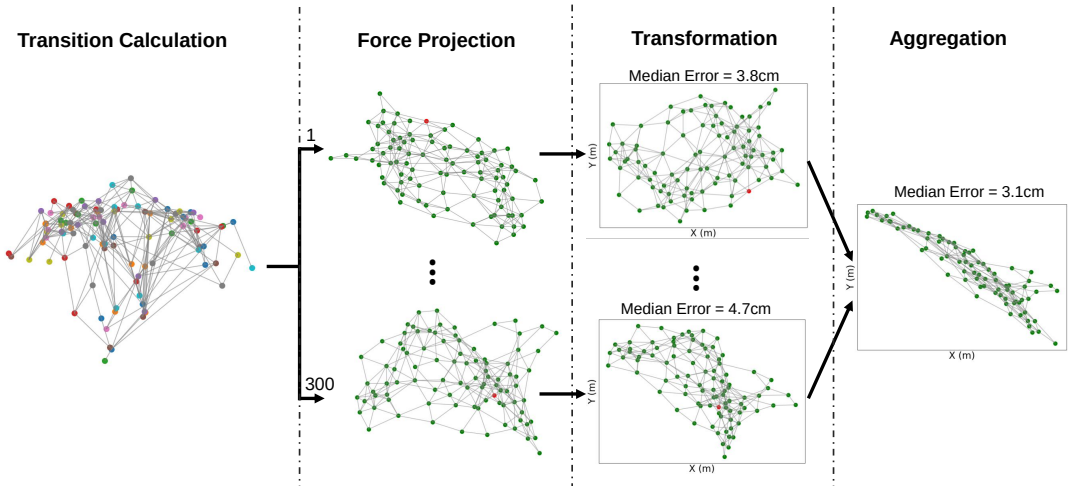

Fig. 7: Visual Overview of TWiST. Transitions over time are calculated over clusters generated from CSI data, forming a graph. The constructed graph then undergoes a force projection to capture the relative proximity of clusters. Using information gleaned from network traffic, these projections are rotated, scaled, and finally translated. To reduce the variation that naturally exists in force projections, this process is repeated 300 times as part of an ensemble, reducing variance and improving accuracy.

accessed to place a nearby transmitter, and the channel is guaranteed to be highly sensitive to their fine-grained motions, allowing for high accuracy. Furthermore, this allows for the attack to be performed at long distances, as well as in private and restricted areas where the wireless channel is less consistent or inaccessible to the attacker. The attacker does not need to know exactly where the victim is; they only need to be close enough to have the victim acknowledge the null packets sent to them.

In practical XR environments, users can freely move, but text entry is generally performed while stationary. Modern XR systems anchor virtual keyboards to the user's head or body frame, making accurate selection difficult while in motion. Our attack operates during these periods of user stability.

C. Location Estimation

The *Location Estimation* module serves to identify which key has been pressed. To enable this, we utilize a *Preprocessing* submodule and a *Graph Generation* submodule.

1) Preprocessing: Before generating positional predictions based on the CSI data, hardware noise must first be removed. One significant source of noise is the Automatic Gain Control (AGC), which alters the amplitude of the CSI measurements across antennas. To correct for this, the amplitude of each antenna's CSI is normalized by the L2 norm of the corresponding channel measurements. Let $H(k, n, a)$ represent the CSI for subcarrier k, time index n, and antenna a. The power for each antenna is calculated as:

$$P(n, a) = \sqrt{\sum_{k} |H(k, n, a)|^2}.$$

The AGC-corrected CSI, $\tilde{H}(k, n, a)$, for each antenna is then computed as:

$$\tilde{H}(k, n, a) = \frac{H(k, n, a)}{P(n, a)}.$$

2) Graph Generation: In this stage of the attack, a novel unsupervised learning-based technique, the Transition Web in Spring Tension (TwiST) Network, is used to project CSI values onto the virtual keyboard. TwiST is a person and environment-invariant CSI sensing method based on graph generation designed to capture motion in environments with consistent and repeating states. Below, we outline each step of the algorithm in detail.

1) **Sliding Window-based CSI Stabilization.** To further smooth the preprocessed CSI, we compute a mean over a sliding window of size $W = 25$ packets, centered on i. Specifically, for each $n \in \{1, \ldots, N\}$, we define

$$\bar{\mathbf{H}}(n) = \frac{1}{W} \sum_{b=n-\lfloor W/2 \rfloor}^{n+\lfloor W/2 \rfloor} \mathbf{H}(b).$$

This produces a stable mean vector $\bar{\mathbf{H}}(n)$ for each index n.

2) **Agglomerative Clustering.** Cluster $\{\bar{\mathbf{H}}(n)\}_{n=1}^{N}$ into $M = 100$ clusters using agglomerative clustering. In a stable wireless environment, the measured CSI value when a finger is at a given location will remain constant, including when the location is later revisited. This one-to-one mapping of finger positions to CSI values means that CSI naturally encodes unique regions of a keyboard. Clustering formally identifies these keyboard regions, where cluster C_m identifies some real position, p_m, on the keyboard. This step is critical for generalization as it allows for later analysis based on the properties of constrained movement between regions of the keyboard, rather than on CSI alone.

3) **Graph Construction.** Construct a weighted graph $G = (V, E)$ by initializing an adjacency matrix $\mathbf{A} \in \mathbb{N}^{M \times M}$ with all entries set to zero. Each cluster $\mathcal{C}_m$ is represented by a node $v_m \in V$. The goal of the remaining steps are to populate the values of G and perform a projection f which minimizes $\sum_{m=1}^{M} \|f(v_m) - p_m\|_2$.
4) **Transition Mapping.** To populate the weighted edges of G, we iterate over the sequence of CSI values $\{\bar{\mathbf{H}}(n)\}$ to identify cluster memberships at each timestamp. Suppose i is the cluster index at time t, and j is the cluster index at time index $n+1$. We record this transition in $\mathbf{A}(i,j) \leftarrow \mathbf{A}(i,j) + 1$.
   Hence, $\mathbf{A}(i,j)$ accumulates the frequency of transitions from region i to region j. These transitions have two main interpretations.
   Firstly, they capture the proximity of nodes to each other. Because nodes correspond to actual regions of the keyboard and movements are continuous, nodes that represent physically adjacent locations on the keyboard will have direct transitions to each other. Consequently, the length of the shortest path between two nodes in G can be used as a proxy for their physical distance.
   Secondly, these transitions capture how close a node is to the center of the keyboard. If we assume each consecutive typed position is chosen uniformly in the rectangular keyboard area and the movement between those positions are straight lines, then nodes closer to the center of the keyboard will be crossed more frequently. This arises because there are many more possible trajectories connecting uniformly distributed endpoints passing through the central region than the outer regions. As a result, nodes corresponding to centered regions tend to accumulate higher degrees and larger weights than peripheral nodes.
5) **Force-Directed Graph Projection.** We embed G in a 2D plane using a force-directed graph layout. Each node v_m receives a 2D coordinate (x_m, y_m), and the transition weights act like springs between nodes. This spring-like model preserves the physical adjacency implied by the transitions of G producing $\mathbf{P} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}$, a coordinate set.
   A force-directed layout is used because it naturally captures the relative placement of nodes of G. Central nodes tend to have more transitions and larger weights $\mathbf{A}(i,j)$) causing them to also exert larger attractive forces on each other, naturally pulling them to the center. Simultaneously, the less-connected peripheral nodes experience weaker forces and tend to remain on the outside. This projection also captures the relative distance between nodes as nearby nodes have shorter paths and will be pulled closer together than further ones. While this representation will capture the relative placement of nodes, it has no concept of global orientation or scale.
6) **Keyboard Alignment Using Enter Press.** Let t_e be a known timestamp when the "Enter" key is pressed, and let v_e be the corresponding node in the 2D layout (i.e., the cluster active at t_e). We rotate all positions in $\mathbf{P}$ about the layout's centroid so that the coordinate of v_e ends up on the right side, consistent with the keyboard layout of the Meta Quest. This step gives an orientation to $\mathbf{P}$, capturing the physical orientation of the keyboard. The "Enter" key is used as an anchor, as it can be easily detected and is found on the same spot on the right side of the keyboard.
7) **Vertical Flip.** Although the previous alignment ensures the enter key is on the right, the layout $\mathbf{P}$ may still be inverted top-to-bottom. By examining additional known press events, we use the heuristic of checking whether most keypress events occur at positions below the enter key to check if $\mathbf{P}$ is upside down. If this is the case, we flip $\mathbf{P}$ across the horizontal axis. This operation preserves inter-node distances while restoring a natural top-to-bottom ordering for the keyboard. This heuristic relies on the physical layout of the keyboard, where fewer keys exist below the enter key and are generally used less frequently.
8) **Keyboard Resizing.** Once oriented, we scale all (x_m, y_m) coordinates to fit into a $10\,\mathrm{cm} \times 30\,\mathrm{cm}$ bounding box. Specifically, define scale factors s_x, s_y such that 90% of predictions occur within this box and multiply all positions by these factors. Because positional predictions are relative, the mapping remains valid under uniform scaling or resizing of user keyboards. As long as key placements preserve their relative spacing and geometry, the same mapping can be applied across many resizes.
9) **Keyboard Offset.** Finally, we translate the scaled layout to align with the physical keyboard's placement. For instance, if we measure the real-world location of the "Enter" key or a reference corner, we can shift all coordinates so that the layout coincides with the actual physical positions. Because the top row of the keyboard features some of the most frequently used keys, this can bias the graph generation to create more nodes corresponding to the top of the keyboard, so the graph is offset upwards to account for this.
10) **Ensembling.** Steps 5 onward are repeated 300 times to generate an ensemble of TwiST networks, and the final prediction is obtained by averaging the positions across all networks in the ensemble. This is done to reduce variance across variations of graph generations in Step 5, which can be subject to randomness, where accuracy among individual generations can vary by several centimeters.

We utilize a graph-based method over a typical formulation of using a deep neural network on CSI values for several reasons. First, a machine learning approach requires extensive training data. Obtaining a sufficient amount of training data for machine learning is non-trivial. Second, many CSI-based sensing methods face severe issues in the generalization of environments. Minor positional differences in train and test data can result in a significant loss of accuracy. This weakness in generalization is overcome with TwiST algorithm by

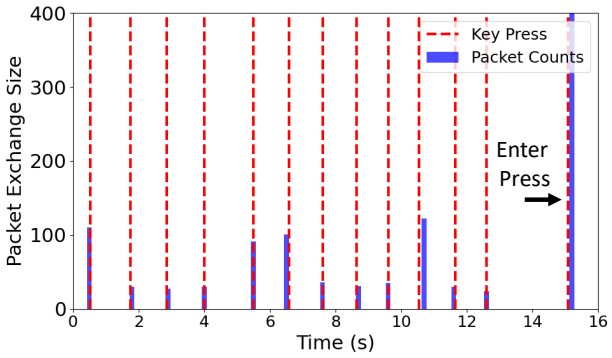

Fig. 8: Keystrokes during Google Search vs Network Activity. When a user presses a key (dashed red line) during a Google Search, a network request is sent to Google's Autocomplete Endpoint, resulting in an immediate exchange of packets. When enter is pressed, a search is completed and a much larger exchange of packets occurs as a page is loaded.

generalizing on properties of constrained motion in a manner that is highly explainable and interpretable.

Via the TwiST algorithm, the adversary can recover a coherent 2-dimensional keyboard layout from CSI alone, without the need for any training data.

D. Press Estimation

The purpose of the *Press Estimation* module is to detect when a press has occurred. Obtaining the timestamp of a press is non-trivial and requires several innovations. *Press Estimation* consists of two steps: *Time Anchoring* and *Press Interpolation*.

1) Time Anchoring: With just CSI information, it is difficult, if not impossible, to detect the timestamp of presses at a time resolution suitable for an attack. To determine the keystroke times of users in XR with high accuracy, a novel exploit involving the examination of a user's wireless activity is used. Through examination of the network traffic on an XR headset, keystrokes on the web can be identified. This weakness exists because many cloud-based services, including Google Search [45], Gmail [46], and Google Docs [47] prompt an exchange of packets every time a user inputs a keypress, often for logging or recommendations. While useful to users, this leads to an exchange of packets corresponding exactly to when a user presses a key. A larger burst of packets can be seen when the enter key is pressed as the user completes their search and loads a new page. Figure 8 illustrates this in more detail.

The attacker leverages this characteristic to find the timestamp of a key press. The attacker analyzes the network activity for bursts of packets spaced at least half a second apart, where the start of the bursts indicates a key press.

2) Press Interpolation: While many websites exchange packets per key press, it was found that login pages, such as those for banks, do not. While packets aren't sent per keypress, a key behavior can be used to detect the keypress timestamps. When the user loads the login page and when the user submits their login information, a large exchange of packets occurs, allowing an adversary to localize the start and end of a typing sequence.

Just the start and end times of a typing sequence are insufficient for detecting the timestamps of every key press. To counter this, the adversary can utilize the typing cadence analyzed when the user searches for their desired page in a search engine to interpolate the key presses, given the start and end times. For example, on the web, the user must use a search engine to access a login page. The typing activity here can be used to profile the typing cadence of the victim. Using the start and end times of their activity, the adversary can interpolate between these points to estimate keypress timestamps.

This approach has merit over other approaches. While machine learning or graph generation could be used to uncover key presses, we found that the key press motion itself is relatively small and is harder to detect than the actual key being pressed. Analyzing the network activity overcomes these limitations by providing an accurate and repeatable method of detecting key press timestamps.

E. Keystroke Inference

Finally, the *Keystroke Inference* module combines the information gathered from the *Location Estimation* module and the *Press Estimation* module to output which keys were pressed, and when. For each known key press time, we extract the corresponding predicted position in the 2D layout. These positions are then mapped to the known layout of the keyboard to determine the most likely key associated with each press.

This process allows us to rank possible keys based on their proximity to the predicted position. The ranking can be used in two ways:

1. **Uniform Key Estimation:** Each key press is uniformly evaluated by taking the highest-ranked key for each predicted position. This provides a straightforward estimation of the pressed key.
2. **Dictionary-Aided Word Ranking:** The rankings can also be combined with a dictionary attack, where the candidate words are scored based on the average ranking of each letter in the word. This approach incorporates language constraints, improving accuracy when a predefined vocabulary or context is available.

Both approaches have merit, and we provide the results for both in Section VI.

VI. Evaluation

A. System Setup

We conducted experiments in a dense metropolitan environment with no environment or interference control, including overlapping WiFi networks, other active devices, and continuous foot traffic. NLoS tests were performed in publicly open areas where people walk by, and across both rooms and buildings. A sample NLoS setup, with the victim and adversary separated by walls and corridors, is shown in Figure 9.

Fig. 9: Attack Setup. A user types in XR (Meta Quest 2) while the attacker sends fake packets to the headsets and analyzes the headset's ACK packets received through the wall.

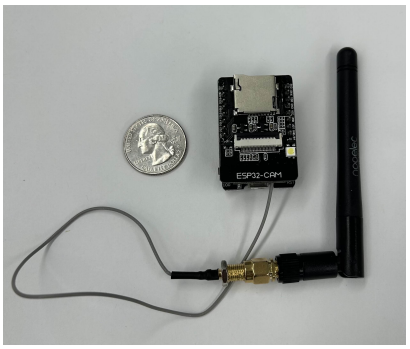

Fig. 10: TwiST uses ESP32 as attacker device which is small (40mm x 27mm x 17mm) and low-cost (10 USD). ESP32 has a microcontroller, a Wi-Fi chipset, and an internal antenna [48]. They can also be configured to use an external antenna.

We use commercial off-the-shelf devices for both attacker and victim devices to run experiments. For the victim, we use a Meta Quest 2 [1]. To enable efficient data collection, we developed a web XR application based on A-Frame with a keyboard that allows the user to freely type. For the attacker, we use two collocated ESP32 modules, which are low-cost small WiFi devices (see Figure 10). In particular, we use one ESP32 as the injector device and another one as a sniffer. The injector is designed to transmit fake WiFi packets, while the sniffer is designed to collect packets from the environment. Two ESP32s were used for ease of running the experiment; however, in practice, the attack can be performed with a single ESP32. Finally, it is worth mentioning that for all experiments we used the internal antenna of ESP32 except for long-range experiments where we used the external small dipole antenna as shown in Figure 10.

Evaluations were performed on 30, 60, and 90-second segments of typing data, each of which was processed in isolation. During these data collection rounds, the user types from a virtual teleprompter in XR, which provides new words uniformly sampled from the most common 10,000 English words [49]. Note that TwiST does not require pre-training and can generate an accurate graph dynamically in less than 90 seconds. Once the graph is generated, it can be used to infer the keystrokes.

TABLE I: Conditions for each experiment type.

Experiment Type	Conditions
Baseline	1m
Distance	1m, 2m, 4m, 10m
Angle	1m: -90°, -45°, 0°, 45°, 90°
Through Wall	1m: -90°, -45°, 0°, 45°, 90°
Indoor Extreme	8m NLOS through wall
Cross Building	30m
New User and Headset	1m: LOS and NLOS

B. Results

To comprehensively evaluate the attack, we consider the following factors to determine their effects on the attack.

- **Distances.** Typical keystroke logging attacks have minimal distance. In contrast, TwiST enables long-range attack. Thus, we evaluate the performance of TwiST for different distances, including both short and long range.
- **Angles.** The angle between the sniffer and the XR headset can change during the user's activity. Therefore, we demonstrate TwiST's high accuracy and resiliency to changes in angle.
- **Environments.** Various factors in the environment could interfere with the attack, such as walls. We demonstrate TwiST's resilience to these factors (even across-building), to highlight TwiST's improvement upon prior attacks.
- **Users and Headsets.** For TwiST to be reliable, we need the attack to work for different users and headsets. Therefore, we evaluate its performance using different headsets and different users.

For all attacks, we report Top-K accuracy. TWiST scores keys on distance to the predicted position. Top-K shows the fraction of TWiST's predictions where the correct key is among the K highest-scoring keys. Words are scored by multiplying per-letter probabilities. Additionally, we report several other statistics, such as median top k, dictionary k, and L2 error across all conditions. A summary of the experiments is shown in Table I.

1) Baseline Test: To create a baseline for TwiST, we test a line-of-sight scenario where the victim and attacker are separated by 1m. The goal of this test is to demonstrate that key press information can indeed be extracted from CSI data and network activity.

The results show that TwiST obtains good accuracy, even achieving up to 94.20% for the Top 16. The attacker can achieve 6.40% in Top 1 scenarios, 38.60% in Top 4, and 72.60% in Top 9. This showcases TwiST ability to localize key presses to a reasonably small region of the keyboard and when those key presses occurred.

2) Effect of Distance: We expand upon the *Baseline Test* by increasing the distance in a line-of-sight scenario. We test TwiST at distances of 2, 4, and 10 meters under LOS conditions.

The results show minimal to no reductions in accuracy even at increased distances. In Figure 12, we observe that the accuracy between 1m LOS, 4m LOS, and 10m LOS are quite similar. We provide the detailed Top 1, Top 4, Top 9, and Top 16 numbers in Table II. Between 1m LOS and 10m

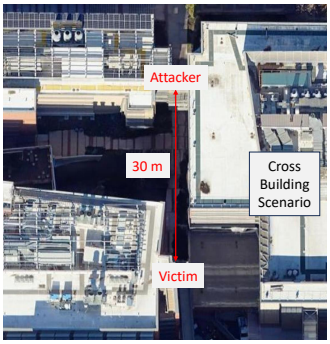

Fig. 11: Aerial view of the 30m cross-building scenario.

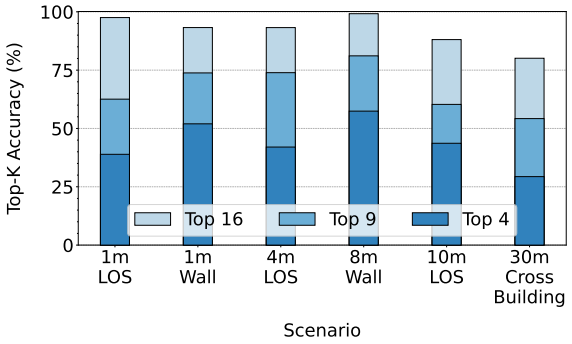

Fig. 12: Effect of distance in different environments on TwiST's accuracy.

LOS, there is minimal accuracy difference, especially for Top 9 and Top 16. It is worth mentioning that TwiST works at a much further range than prior work, mainly due to the fact that it eliminates the need for a dedicated attacker transmitter and instead forces the victim headset (which is in proximity of the user's hand) to continuously transmit.

3) Effect of Angle: We demonstrate TwiST's robustness to changes of angle. The attack is evaluated at 1m in a line-of-sight scenario. Angles are tested at {-90, -45, 0, 45, 90} degrees relative to the target.

Compared to the *Baseline Test*, TwiST is able to maintain reasonable accuracy. For example, at -90°LOS and -45°LOS, the attack has higher accuracy than at 0°for Top 1 and Top 4. Some variations may exist due to the different wireless signal paths measured via these different angles, but TwiST maintains reasonable accuracy in all settings. The ability to be robust to angle increases attack feasibility as the attacker may not be able to orient their measurement setup the exact same every time.

4) Effect of Walls: For the through the wall test, we separate the victim and the sniffer with a wall. For this setup, the point-to-point distance between the victim and the sniffer is 1m.

The attack accuracy actually improves in the 1m through-the-wall scenario compared to the *Baseline Test*. The Top 1 accuracy is nearly double for through the wall (12.50% compared to 6.40%), reaching an even higher Top 16 accuracy (98.00 compared to 94.20%). Compared to prior attacks, this substantially increases the attack feasibility in real scenarios as the adversary does not even need to be in the same room

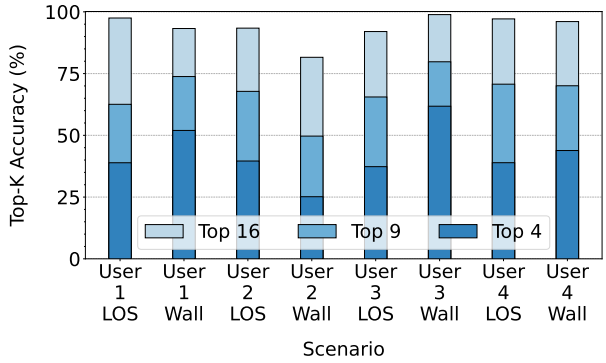

Fig. 13: Effect of different users on TwiST's accuracy, in different environments.

as the victim. This is enabled by the removal of the constraint that the victim needs to be between the adversarial transmitter and receiver.

5) Effect of User: To demonstrate robustness towards different users, we conduct a 1m LOS and 1m through-the-wall test for two users, each with different headsets. We compare the results between all scenarios to determine if TwiST is generalizable to different users. We additionally compare to a Quest 3 headset in LOS on User 1.

TwiST achieves similar performance across both users and headsets. As shown in Figure 13, the Top-K accuracies are largely similar. The attack is slightly more accurate on User 2 in an LOS setting, but is slightly more accurate on User 1 in a through-the-wall scenario. This demonstrates TwiST's applicability to a wide variety of users, even unseen users. This is enabled because TwiST generalizes on properties of motion found across users and headsets.

6) Complex Scenarios: Realistic attack environments may contain several combinations of distances, angles, and variables. Thus, we test across a wide combination of these variables, as shown in Table II. For example, we test 8m through the wall, and at various angles through the wall at 1m. The goal of these tests is to demonstrate that TwiST can be applied to more complicated scenarios with more sources of error.

Compared to their simpler counterparts, the complex scenario experiments are able to achieve similar accuracies. Comparing the angle tests through the wall and not through the wall, we achieve even up to double the accuracy in Top 1 scenarios, such as in 45°and 90°scenarios. In the 8m through-the-wall test, we obtain 81% accuracy in the Top 16. TwiST is able to maintain success even in more complicated environments.

7) Cross Building: Finally, we test the feasibility of an attack on a larger scale. In this test, the victim is in one building while the attacker is in another, as shown in Figure 11. The victim and attacker are separated by 30m. The goal of this test is to demonstrate the range and limits of TwiST.

The attack achieves remarkable accuracy, even compared to the *Baseline Test*. The attack achieves 16.30% accuracy in Top 1, 51.40% accuracy in Top 4, 74.30% accuracy in Top 9, and

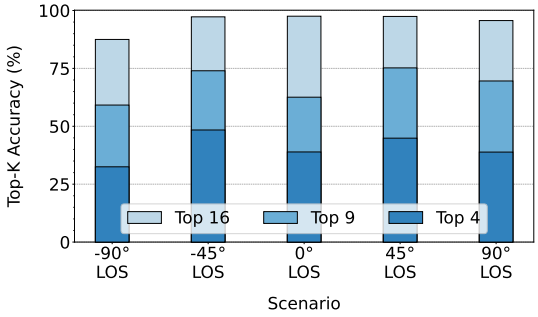

Fig. 14: Effect of angle on TwiST accuracy.

Env.	Top 4	Top 9	Top 16
1m LOS	38.91%	62.56%	97.54%
2m LOS	33.06%	61.22%	93.88%
4m LOS	42.02%	73.91%	93.24%
8m Wall	22.40%	52.70%	81.00%
10m LOS	31.40%	65.60%	93.50%
30m Bld.	51.40%	74.30%	97.10%
-90°LOS	32.50%	59.17%	87.50%
-90°Wall	34.30%	53.72%	76.86%
-45°LOS	48.40%	73.97%	97.26%
-45°Wall	57.43%	81.12%	99.20%
0°LOS	38.91%	62.56%	97.54%
0°Wall	57.43%	81.12%	99.20%
45°LOS	44.87%	75.21%	97.44%
45°Wall	35.19%	67.13%	93.98%
90°LOS	38.86%	69.57%	95.65%
90°Wall	46.00%	67.39%	95.90%
User 1 LOS	38.90%	62.56%	97.50%
User 2 LOS	39.60%	67.80%	93.40%
User 3 LOS	37.32%	65.53%	92.02%
User 4 LOS	38.92%	70.71%	97.14%
User 1 Wall	51.98%	73.80%	93.25%
User 2 Wall	25.15%	49.69%	81.59%
User 3 Wall	61.79%	79.78%	98.88%
User 4 Wall	43.85%	70.07%	96.06%
Quest 3 LOS	49.12%	75.40%	89.66%
Mean	**40.77%**	**69.24%**	**91.54%**

TABLE II: TopK Per Key Across All Conditions.

97.10% in Top 16. This highlights that TwiST can be applied even on grander scales, increasing the potential risk of this attack on end users.

8) Dictionary Attack: Using the same set of experiments, we conduct a dictionary attack, where we use constraints from language to improve accuracy. The goal is to determine the effect of language constraints on TwiST accuracy.

The average median top k for the dictionary attack is 24.175. This means the midpoint number of words that need to be guessed by the adversary is around 24. Furthermore, we evaluate the dictionary attack on Top 1 through Top 1000 scenarios, at several increments, as shown in Table VI in Appendix A. TwiST achieves 75.18% accuracy in Top 50, 92.91% accuracy in Top 250, and 97.14% in Top 500.

9) Positional L2 Error: To further highlight TwiST's positional accuracy, we report the L2 error in cm across our experiments. A low L2 error suggests that our model can localize effectively.

The average L2 error for all experiments is 4.113cm, as shown in Table V. Based on the dimensions of the virtual keyboard, 4cm accuracy means that on average, the correct key is within the surrounding nearby keys, significantly reducing the guesses needed in a brute force attack.

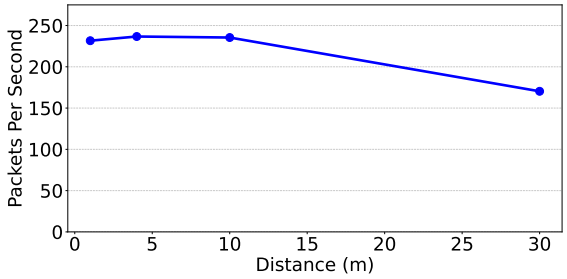

Fig. 15: Effect of Distance (victim-attacker) on Packets Per Second received from the XR Headset, forced by the attacker.

10) Distance Analysis: To further analyze the data, we compared the average number of packets received from the Meta Quest per second at various distances. In Figure 15, we graph the average packets per second received at 1m, 4m, 6m nLOS, 10m, and across a building.

We extract several insights from these graphs that demonstrate the validity of our approach. First, the number of packets received per second does not substantially degrade across 1-10m, remaining at around 220 packets per second. We notice a reduction in packets per second in our cross-building scenario; however, we still receive 170 packets per second, which is more than enough to achieve high accuracy. Although not validated, we believe TwiST can further increase the attack distance (50m). We did not further test this due to logistical constraints and left it for future work.

11) Comparison to Machine Learning: We compare TwiST's algorithm to a machine learning-based approach. We trained a traditional ResNet-based model on data collected in many environments. We use our best effort and best judgment to provide a model representative of a deep learning approach. We compare the average positional error of the machine learning model to our graph generation-based model.

Our graph generation-based approach has significantly better positional performance compared to machine learning. In Table IV, we showcase the L2 error (in cm) for TwiST's algorithm compared to ML. In new environments, TwiST performs significantly better than machine learning (4.1cm compared to 7.5cm, respectively). In fact, for ML, the new person and new environment scenario result is just slightly better than guessing the mean ground truth hand location (7.8cm). Each environment change widely affects the CSI characteristics, taking it out of the ML training distribution. Thus, while we provide a diverse training set to the machine learning model, it is still unable to generalize to unseen environments. Instead, TwiST removes the need for a large amount of training data and only uses a short data sequence.

C. Comparison to Priors

Table III situates TwiST among existing keystroke-inference attacks across XR, mobile, and traditional keyboards. We

TABLE III: Comparison of Keystroke Inference Attacks

XR Attacks	Target	Modality	Zero-Shot	New User	New Env	NLOS	Range	Cost	Key GOR	Log_{10} Word GOR
TyPose [10]	XR-Controller	Malware	N	Y	N	N/A	N/A	$0	N/A	0.89
Privacy [11]	XR-Controller	Malware	N	Y	Y	N/A	N/A	$0	34.85	12.56
Head(set) [12]	XR-Hand	Malware	N	N	N	N/A	N/A	$0	9.40	N/A
Hologlogger [15]	XR-Hand	Malware	Y	Y	Y	N/A	N/A	$0	N/A	11.18
Protect [22]	XR-Keyboard	Multi-User Apps	N	N	Y	N/A	N/A	$0	26.48	N/A
Remote [36]	XR-Controller	Multi-User Apps	Y	Y	Y	N/A	N/A	$0	13.67	N/A
Keylogging [13]	XR-Hand	Hand Tracker	N	Y	Y	N	0.8m	$200	N/A	22.07
I know [14]	XR-Head	Camera	Y	Y	Y	N	2m	$500	N/A	12.11
HR [18]	XR-Hand	Camera	Y	Y	Y	N	10m	$750	18.55	2.65
VRecKey [40]	XR-Controller	IR Array	Y	Y	Y	Y	4m	$120	26.60	21.81
Strokes [39]	Keyboard	RSS	N	N	Y	Y	20m	$100	N/A	N/A
WiKI-Eve [28]	Smartphone	BFI	N	Y	Y	Y	10m	$40	14.40	8.38
WindTalker [21]	Smartphone	CSI	N	N	N	Y	1.6m	$20	3.17	5.30
Wiki [20]	Keyboard	CSI	N	N	N	N	4m	$40	35.7	N/A
Hurdles [50]	Keyboard	CSI	Y	Y	Y	Y	3.5m	$5000+	N/A	3.16
VR-Spy [19]	XR-Controller	CSI	N	N	N	N	1.27m	∼$400	25.11	N/A
TwiST (ours)	XR-Hand	CSI	Y	Y	Y	Y	30m	$20	4.20	2.70

Note: GOR is the ratio of top-K accuracy to the likelihood of a correct classification given K random guesses over the method's input space. TwiST achieves strong GOR while functioning across users, environments, NLOS, and long ranges.

Cond.	Rand.	ML	Transfer	TwiST's Alg.
Same	11.6	4.0	4.0	4.1
New	11.6	7.5	5.5	4.1

TABLE IV: A comparison of different approaches and the average L2 error in centimeters. Same = Same Person Same Environment. New = New Person New Environment.

compare methods using Gain Over Random (GOR), which measures how many times better a system performs than a random guess over its input space. Relative to prior XR attacks, TwiST provides practical zero-shot usability, substantially longer operational range, and strong key- and word-level GOR, showing that CSI-based keystroke inference can function in real XR settings rather than controlled environments.

Compared to prior CSI-based keystroke-inference attacks, TwiST is the only method that targets XR-hand input, operates at tens of meters, and requires no victim-specific calibration or training data (with the partial exception of [50]). While its GOR is lower than CSI attacks on physical keyboards or smartphones, those methods rely on fixed, highly constrained input surfaces. TwiST instead prioritizes real-world deployability in unconstrained XR environments, a setting that prior CSI techniques were not designed to support.

D. Impact on Network Performance

To investigate the impact of this attack on WiFi network performance, we evaluate the impact of our attack on a simple network consisting of three laptops and Meta Quest 2, and a phone, streaming 4k YouTube videos. During the attack, all devices remained capable of streaming 4k without interruption. We found that on our simple network, the attack adds a small overhead of just 20% of the packets sent.

E. MAC Address Derandomization

A device fingerprinting tool was built following the specifications described in prior work [43], [44]. Using this tool, a fingerprint of a Meta Quest 3 was produced, which was successfully used to identify a second Meta Quest 3 device from nearly 1000 other devices. This highlights the feasibility of such an approach, where attackers can easily build fingerprints of XR devices and identify them at range.

F. Ablations

Several ablation studies are performed, evaluating design choices, data characteristics, and network monitoring. The results can be found in Appendix A.

VII. Discussions

There are several key limitations of our work. The variety of users was limited to three people who experienced with XR. While strong results were seen across user orientations and environments, it is possible that TwiST could fail on certain body types or less experienced users. The data collection and evaluation were done with users staying fairly still. While XR keyboards remain static, not all XR users do. The efficacy of TwiST requires consistency in the motions of a user; if their motions are random or dramatic (walking around or strongly fidgeting while typing) it will reduce accuracy. The results of our dictionary attack are limited by the quality of the dictionary used. In our data collection and evaluation, all typed words were found in the dictionary that was later searched, guaranteeing that each password could be found.

VIII. Conclusions

This work presented a novel and robust keylogging attack leveraging WiFi wireless sensing, which overcomes the limitations of prior methods by eliminating the need for line of sight, close proximity, or complex setups. By exploiting vulnerabilities in WiFi chipsets and utilizing a novel signal processing algorithm, we achieved accurate keystroke inference under diverse conditions and distances, highlighting the pressing need for stronger security measures in XR devices and wireless communication protocols.

Acknowledgment

We thank our reviewers for their guidance. This work has been supported, in part, by NSF grants CNS-2338623, CNS-2303115, and CNS-2312089. The views and findings in this

paper are those of the authors and do not necessarily reflect the views of NSF.

REFERENCES

[1] Meta, "Meta quest 2," 2020. [Online]. Available: https://www.meta.com/quest/products/quest-2/

[2] ——, "Meta quest 3," 2020. [Online]. Available: https://www.meta.com/quest/quest-3/

[3] A. S. Pillai and P. S. Mathew, "Impact of virtual reality in healthcare: a review," *Virtual and augmented reality in mental health treatment*, pp. 17–31, 2019.

[4] U.S. Chamber of Commerce. (n.d.) How virtual reality is transforming healthcare. [Online]. Available: https://www.uschamber.com/technology/how-virtual-reality-is-transforming-healthcare

[5] S. Alizadehsalehi, A. Hadavi, and J. C. Huang, "Virtual reality for design and construction education environment," *AEI 2019*, pp. 193–203, 2019.

[6] J. E. Naranjo, D. G. Sanchez, A. Robalino-Lopez, P. Robalino-Lopez, A. Alarcon-Ortiz, and M. V. Garcia, "A scoping review on virtual reality-based industrial training," *Applied Sciences*, vol. 10, no. 22, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/22/8224

[7] C. Brown, J. Hicks, C. H. Rinaudo, and R. Burch, "The use of augmented reality and virtual reality in ergonomic applications for education, aviation, and maintenance," *Ergonomics in Design*, vol. 31, no. 4, pp. 23–31, 2023.

[8] Meta. (n.d.) For those using metaverse technologies, the impact is real. [Online]. Available: https://about.fb.com/news/2023/09/impact-of-vr-and-ar/

[9] A. Inc. (2024) Use apple vision pro at work. [Online]. Available: https://support.apple.com/guide/apple-vision-pro/use-apple-vision-pro-at-work-tan2f5f77158/visionos

[10] C. Slocum, Y. Zhang, N. Abu-Ghazaleh, and J. Chen, "Going through the motions:{AR/VR} keylogging from user head motions," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 159–174.

[11] Y. Wu, C. Shi, T. Zhang, P. Walker, J. Liu, N. Saxena, and Y. Chen, "Privacy leakage via unrestricted motion-position sensors in the age of virtual reality: A study of snooping typed input on virtual keyboards," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 2023, pp. 3382–3398.

[12] Y. Zhang, C. Slocum, J. Chen, and N. Abu-Ghazaleh, "It's all in your head (set): Side-channel attacks on ar/vr systems," in *USENIX Security*, 2023.

[13] Ü. Meteriz-Yıldıran, N. F. Yıldıran, A. Awad, and D. Mohaisen, "A keylogging inference attack on air-tapping keyboards in virtual environments," in *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2022, pp. 765–774.

[14] Z. Ling, Z. Li, C. Chen, J. Luo, W. Yu, and X. Fu, "I know what you enter on gear vr," in *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2019, pp. 241–249.

[15] S. Luo, X. Hu, and Z. Yan, "Holologger: Keystroke inference on mixed reality head mounted displays," in *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2022, pp. 445–454.

[16] H. Khalili, A. Chen, T. Papaiakovou, T. Jacques, H.-J. Chien, C. Liu, A. Ding, A. Hass, S. Zonouz, and N. Sehatbakhsh, "Virtual keymysteries unveiled: Detecting keystrokes in vr with external side-channels," in *2024 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2024, pp. 260–266.

[17] S. Luo, A. Nguyen, H. Farooq, K. Sun, and Z. Yan, "Eavesdropping on controller acoustic emanation for keystroke inference attack in virtual reality," in *The Network and Distributed System Security Symposium (NDSS)*, 2024.

[18] S. R. K. Gopal, D. Shukla, J. D. Wheelock, and N. Saxena, "Hidden reality: Caution, your hand gesture inputs in the immersive virtual world are visible to all!" in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 859–876.

[19] A. Al Arafat, Z. Guo, and A. Awad, "Vr-spy: A side-channel attack on virtual key-logging in vr headsets," in *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. IEEE, 2021, pp. 564–572.

[20] K. Ali, A. X. Liu, W. Wang, and M. Shahzad, "Keystroke recognition using wifi signals," in *Proceedings of the 21st annual international conference on mobile computing and networking*, 2015, pp. 90–102.

[21] 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 the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 1068–1079.

[22] Z. Yang, Z. Sarwar, I. Hwang, R. Bhaskar, B. Y. Zhao, and H. Zheng, "Can virtual reality protect users from keystroke inference attacks?" in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 2725–2742.

[23] Z. Yang, Y. Chen, Z. Sarwar, H. Schwartz, B. Y. Zhao, and H. Zheng, "Towards a general video-based keystroke inference attack," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 141–158.

[24] Y. Chen, T. Li, R. Zhang, Y. Zhang, and T. Hedgpeth, "Eyetell: Video-assisted touchscreen keystroke inference from eye movements," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 144–160.

[25] L. Mei, R. Liu, Z. Yin, Q. Zhao, W. Jiang, S. Wang, S. Wang, K. Lu, and T. He, "mmspyvr: Exploiting mmwave radar for penetrating obstacles to uncover privacy vulnerability of virtual reality," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 8, no. 4, pp. 1–29, 2024.

[26] J. Hu, T. Zheng, Z. Chen, H. Wang, and J. Luo, "Muse-fi: Contactless muti-person sensing exploiting near-field wi-fi channel variation," in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, 2023, pp. 1–15.

[27] A. Abedi and O. Abari, "Wifi says" hi!" back to strangers!" in *Proceedings of the 19th ACM Workshop on Hot Topics in Networks*, 2020, pp. 132–138.

[28] J. Hu, H. Wang, T. Zheng, J. Hu, Z. Chen, H. Jiang, and J. Luo, "Password-stealing without hacking: Wi-fi enabled practical keystroke eavesdropping," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 239–252.

[29] Z. Zhang, N. Avazov, J. Liu, B. Khoussainov, X. Li, K. Gai, and L. Zhu, "Wipos: A pos terminal password inference system based on wireless signals," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7506–7516, 2020.

[30] A. Bhatia, M. H. Mughrabi, D. Abdlkarim, M. Di Luca, M. Gonzalez-Franco, K. Ahuja, and H. Seifi, "Text entry for xr trove (text): Collecting and analyzing techniques for text input in xr," in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, 2025, pp. 1–18.

[31] Apple, "Apple vision pro," 2024. [Online]. Available: https://www.apple.com/apple-vision-pro/

[32] D. Abdlkarim, M. Di Luca, P. Aves, M. Maaroufi, S.-H. Yeo, R. C. Miall, P. Holland, and J. M. Galea, "A methodological framework to assess the accuracy of virtual reality hand-tracking systems: A case study with the meta quest 2," *Behavior research methods*, vol. 56, no. 2, pp. 1052–1063, 2024.

[33] Y. Ma, G. Zhou, and S. Wang, "Wifi sensing with channel state information: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 3, pp. 1–36, 2019.

[34] W. Jiang, C. Miao, F. Ma, S. Yao, Y. Wang, Y. Yuan, H. Xue, C. Song, X. Ma, D. Koutsonikolas *et al.*, "Towards environment independent device free human activity recognition," in *Proceedings of the 24th annual international conference on mobile computing and networking*, 2018, pp. 289–304.

[35] M. Zhao, F. Adib, and D. Katabi, "Emotion recognition using wireless signals," in *Proceedings of the 22nd annual international conference on mobile computing and networking*, 2016, pp. 95–108.

[36] Z. Su, K. Cai, R. Beeler, L. Dresel, A. Garcia, I. Grishchenko, Y. Tian, C. Kruegel, and G. Vigna, "Remote keylogging attacks in multi-user {VR} applications," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 2743–2760.

[37] D. Asonov and R. Agrawal, "Keyboard acoustic emanations," in *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*. IEEE, 2004, pp. 3–11.

[38] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 1, pp. 1–26, 2009.

[39] G. Oligeri, S. Sciancalepore, S. Raponi, and R. Di Pietro, "Brokenstrokes: on the (in) security of wireless keyboards," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 231–241.

[40] T. Ni, Y. Du, Q. Zhao, and C. Wang, "Non-intrusive and unconstrained keystroke inference in vr platforms via infrared side channel," *arXiv preprint arXiv:2412.14815*, 2024.
[41] A. Abedi, H. Lu, A. Chen, C. Liu, and O. Abari, "Wifi physical layer stays awake and responds when it should not," *IEEE Internet of Things Journal*, vol. 11, no. 3, pp. 4483–4496, 2024.
[42] Apple, "Use private wi-fi addresses on apple devices," 2024. [Online]. Available: https://support.apple.com/en-us/102509
[43] R. Rusca, F. Sansoldo, C. Casetti, and P. Giaccone, "What wifi probe requests can tell you," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*. IEEE, 2023, pp. 1086–1091.
[44] G. Baccichet, C. Innamorati, A. E. Redondi, and M. Cesana, "Mac address de-randomization using multi-channel sniffers and two-stage clustering," in *2024 IEEE 35th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2024, pp. 1–6.
[45] Google, "How Google autocomplete predictions work," 2024. [Online]. Available: https://support.google.com/websearch/answer/7368877?hl=en
[46] ——, "Learn about Google Tasks," 2024. [Online]. Available: https://support.google.com/tasks/answer/7675772?hl=en
[47] ——, "How to use Google Docs," 2024. [Online]. Available: https://support.google.com/docs/answer/7068618?hl=en&co=GENIE.Platform%3DDesktop
[48] HiLetgo, "Hiletgo esp32-cam development board," 2025, accessed: 2025-01-22. [Online]. Available: https://www.amazon.com/HiLetgo-ESP32-CAM-Development-Bluetooth-Raspberry/dp/B07RXPHYNM
[49] https://github.com/first20hours. (2013) google-10000-english. [Online]. Available: https://github.com/first20hours/google-10000-english/blob/master/google-10000-english.txt
[50] S. Fang, I. Markwood, Y. Liu, S. Zhao, Z. Lu, and H. Zhu, "No training hurdles: Fast training-agnostic attacks to infer your typing," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1747–1760.

APPENDIX

A. Ablations

To better understand the characteristics of TwiST, we perform ablations to analyze the impact of design choices and data characteristics on its results. Each analysis was performed on the data from the Cross-Building Scenario. The first category on which TwiST is evaluated is the length of data it is provided. We evaluate its performance when data is provided in 30s, 60s, and 90s chunks. As seen in Figure 18, reducing the temporal length of data provided to TwiST from 90 seconds to 30 did not cause any significant loss of accuracy, and performance slightly improved from 3.9cm error to 3.7cm. In Figure 19 we examine ablation L, which evaluates the impact of an optimization of the dictionary attack, wherein the words evaluated in the dictionary are restricted to words whose lengths match the number of observed keypresses. Removing this optimization increases the median top-k per-word score on the dictionary attack from 4.5 to 37.

The efficacy of ensembling is tested in the S ablation, where a single graph is used rather than the ensemble of 300. The results of this ablation show that ensembling plays a large role in reducing prediction error, as using only a single graph causes an increase in error from 3.9cm to 5.1cm due to the significant randomness in generating a force-directed graph layout shown in Figure 20. In ablation P, we do not provide press times to TwiST, simulating when passwords are entered and press times must be estimated using a user's typing cadence. Our final evaluation explores the combination of these conditions, shown in Figure 21. Requiring the press

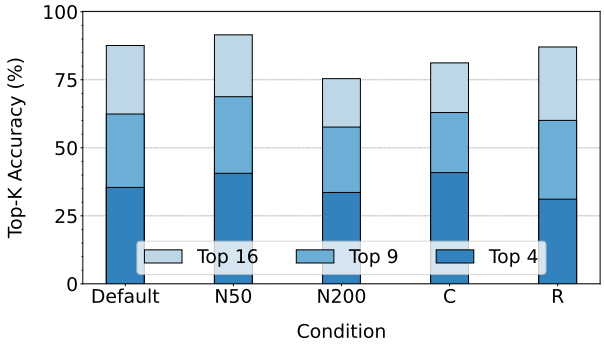

Fig. 16: Effect of graph generation parameters on top-k per letter accuracy. N50 and N200 change the number of nodes in the graph from 100 to 50 and 200, respectively. C changes the clustering algorithm from Agglomerative to Gaussian, and R projects the final predictions to a keyboard 5cm longer.

times to be estimated from cadence increases top-k per letter from 4.0 to 6.0, reducing the provided data increases the top-k to 7.0. Finally, after removing aggregation, we achieve a median top-k score of 9, which still yields an improvement over random guessing.

The influence of graph-generation choices on a single environment is evaluated in Figure 16. Reducing the number of layout nodes from 100 to 50 (N50) improves performance, while increasing the node count to 200 (N200) degrades it, suggesting that overly clustered graphs amplify noise in the force-directed layout. Changing the clustering algorithm from Agglomerative to a Gaussian mixture model (C) raises Top-4 accuracy but lowers Top-16 accuracy, indicating a less stable cluster structure at larger k. Finally, projecting predictions onto a keyboard extended by 5cm (R) results in a small but consistent decline in accuracy.

To evaluate the stability of CSI under realistic ambient conditions, we perform a sensitivity analysis where we collected measurements while typing the same word in XR across two conditions within the same environment: one stable and one with two users moving and multiple additional devices active on the network. As shown in Figure 17, the cleaned amplitude traces for subcarrier 55 remain highly consistent between the two recordings, exhibiting a correlation of 0.94 despite substantial motion and network activity. This illustrates that TwiST remains robust even when the surrounding environment is in motion.

To assess whether TwiST remains practical on real, monitored networks, we conducted a sensitivity analysis using polite WiFi on both open and password-protected public networks with active traffic monitoring. Over a continuous 16-hour period, TwiST consistently extracted usable CSI without triggering any administrative alerts or interference, demonstrating that polite WiFi remains effective even on managed public networks.

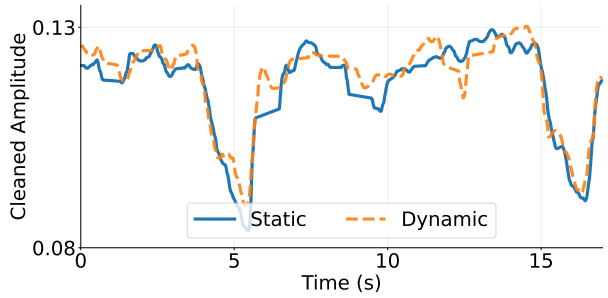

Fig. 17: CSI amplitude values for typing the same word in XR under static and dynamic scenarios. For the dynamic scenario, users walked around the environment, and the network had additional activity. Despite this, amplitudes remain strongly correlated at 0.94.

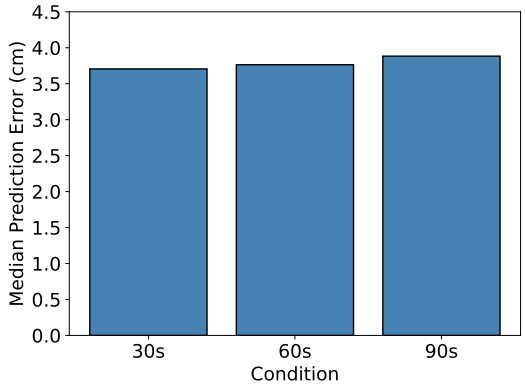

Fig. 18: L2 Errors From Graphs Generated Using 30, 60, and 90 seconds of Data. The median positional prediction error remains consistent with a slight decrease in error as the length of data provided to it decreases.

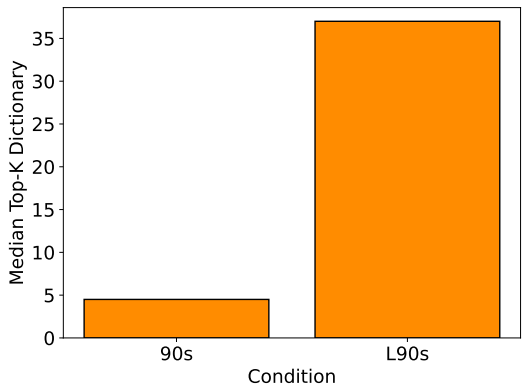

Fig. 19: Median Top-K Dictionary Scores with and without known word lengths (L). Removing the optimization on the dictionary attack, which limits the search to words that match the number of observed keypresses, increases the median top-k value in the dictionary attack from 4.5 to 37.

TABLE V: Median TopK and L2 Values across all Conditions.

Env.	Letter-K	Dict-K	L2 Error (cm)
1m LOS	6.00	9.000	3.920
2m LOS	8.00	57.000	4.130
4m LOS	5.00	4.000	3.480
8m Wall	9.00	112.500	5.440
10m LOS	7.00	28.000	4.580
30m Bld.	4.00	4.500	3.880
-90°LOS	5.00	4.000	4.150
-90°Wall	5.00	9.000	3.400
-45°LOS	6.00	59.500	5.320
-45°Wall	5.00	1.000	3.590
0°LOS	6.00	9.000	3.920
0°Wall	5.00	2.500	3.750
45°LOS	10.00	114.500	5.250
45°Wall	6.00	16.000	3.970
90°LOS	8.00	33.000	4.720
90°Wall	4.00	1.000	3.610
User 1 LOS	5.00	2.500	3.920
User 2 LOS	5.00	7.000	3.590
User 3 LOS	7.00	35.000	4.994
User 4 LOS	6.0	18.500	4.38
User 1 Wall	5.00	2.500	3.750
User 2 Wall	5.00	7.000	3.890
User 3 Wall	5.00	38.500	5.151
User 4 Wall	6.00	14.500.	3.984
Quest 3 LOS	5.00	13.000	3.790
Mean	**5.95**	**24.175**	**4.113**

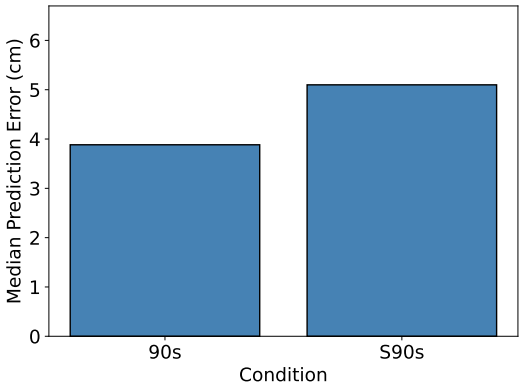

Fig. 20: Positional Error in an Ensembling vs Single Graph (S). By aggregating predictions across many graphs, TwiST improves accuracy and reduces its variability.

TABLE VI: Dictionary Attack Performance Across all Conditions.

Environment	Top 5	Top 10	Top 25	Top 50	Top 100	Top 250	Top 500	Top 1000
1m LOS	44.60%	52.30%	75.40%	87.70%	95.40%	98.50%	100.00%	100.00%
2m LOS	14.80%	14.80%	37.00%	48.10%	70.40%	88.90%	96.30%	96.30%
4m LOS	51.20%	60.50%	65.10%	76.70%	83.70%	93.00%	97.70%	100.00%
8m Wall	10.30%	15.50%	25.90%	34.50%	48.30%	67.20%	91.40%	96.60%
10m LOS	29.00%	38.70%	41.90%	64.50%	74.20%	93.50%	96.80%	100.00%
30m Cross-Building	53.60%	61.90%	78.60%	90.50%	96.40%	98.80%	100.00%	100.00%
-90°LOS	52.80%	66.00%	75.50%	77.40%	88.70%	100.00%	100.00%	100.00%
-90°Wall	38.70%	51.60%	64.50%	67.70%	77.40%	96.80%	96.80%	100.00%
-45°LOS	22.90%	31.20%	39.60%	50.00%	58.30%	70.80%	81.20%	93.80%
-45°Wall	60.70%	75.40%	85.20%	95.10%	96.70%	100.00%	100.00%	100.00%
0°LOS	44.60%	52.30%	75.40%	87.70%	95.40%	98.50%	100.00%	100.00%
0°Wall	57.70%	69.20%	88.50%	94.20%	96.20%	100.00%	100.00%	100.00%
45°LOS	4.40%	4.40%	10.30%	23.50%	45.60%	80.90%	94.10%	100.00%
45°Wall	30.10%	41.90%	59.10%	75.30%	87.10%	97.80%	100.00%	100.00%
90°LOS	23.00%	29.50%	42.60%	62.30%	72.10%	77.00%	88.50%	96.70%
90°Wall	77.80%	87.00%	98.10%	100.00%	100.00%	100.00%	100.00%	100.00%
User 1 LOS	44.60%	52.30%	75.40%	87.70%	95.40%	98.50%	100.00%	100.00%
User 2 LOS	59.30%	66.70%	81.50%	96.30%	96.30%	100.00%	100.00%	100.00%
User 3 LOS	25.00%	29.17%	41.67%	66.67%	70.83%	91.67%	91.67.%	95.83%
User 4 LOS	43.33%	50.00%	50.00%	83.33%	100.00.%	100.00%	100.00%	100.00%
User 1 Wall	57.70%	69.20%	88.50%	94.20%	96.20%	100.00%	100.00%	100.00%
User 2 Wall	45.10%	64.70%	76.50%	90.20%	92.20%	98.00%	100.00%	100.00%
User 3 Wall	37.50%	50.00%	50.00%	75.00%	100.00%	100.00%	100.00%	100.00%
User 4 Wall	32.69%	42.31%	65.38%	75.00%	82.69%	96.15%	100.00%	100.00%
Quest 3 LOS	26.47%	41.18%	61.76%	70.59%	79.41%	88.24%	100.00%	100.00%
Mean	**41.15%**	**50.26%**	**64.23%**	**75.18%**	**83.30%**	**92.91%**	**97.14%**	**99.17%**

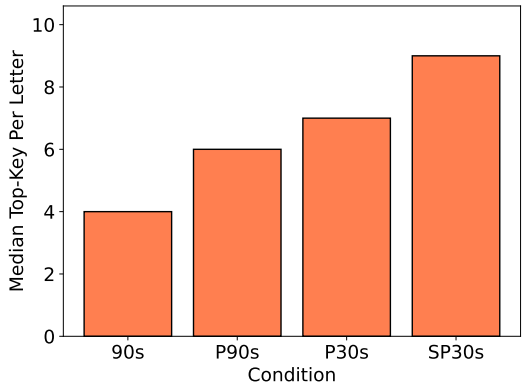

Fig. 21: Median Top-K Scores Across All Conditions. Performance is evaluated when press-times must be estimated by cadence (P), when data is further reduced to 30s, when the dictionary attack is unrestricted, and finally when only a single graph is used (S).