

USENIX

THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Motion in the Clear: Reconstructing VR User Behavior from Network Traffic

JiHo Lee, *Virginia Tech;* JinYi Yoon, *Virginia Tech and Inha University;* Taejoong Chung, Brendan David-John, and Bo Ji, *Virginia Tech*

https://www.usenix.org/conference/usenixsecurity26/presentation/lee-jiho

This paper is included in the Proceedings of the 35th USENIX Security Symposium.

August 12–14, 2026 • Baltimore, MD, USA

ISBN 978-1-939133-58-8

Open access to the Proceedings of the 35th USENIX Security Symposium is sponsored by



جامعة الملك عبدالله للعلوم والتقنية

King Abdullah University of Science and Technology

ARTIFACT EVALUATED
USENIX
AVAILABLE

MOTION IN THE CLEAR: Reconstructing VR User Behavior from Network Traffic

JiHo Lee[1], JinYi Yoon[1, 2], Taejoong Chung[1], Brendan David-John[1], Bo Ji[1]
[1]*Virginia Tech,* [2]*Inha University*

Abstract

Virtual Reality (VR) applications stream high-rate head and hand motion to support real-time multi-user interaction. This motion telemetry is privacy sensitive, enabling behavioral inference such as user identification and virtual typing inference. We show that VR motion can be reconstructed from passively observed network traffic alone, even when the adversary has no application access, device compromise, or message-format knowledge. From an ecosystem measurement of 75 popular VR applications, we find that multiplayer uplink traffic is dominated by sustained, high-frequency UDP and that payload encryption is uncommon. Despite application-specific serialization and unknown encodings, motion updates retain a smooth temporal structure that remains visible in plaintext UDP payload bytes.

Leveraging this property, we present PACMO, a passive, encoding-agnostic attack that uses controlled input patterns to automatically localize motion-related packet fields and reconstruct head and hand trajectories in a black-box manner. The reconstructed motion enables high-impact downstream attacks, achieving up to 96.2% user identification accuracy and 69.4% virtual typing inference accuracy. Our results expose a systemic network-layer privacy risk in immersive systems and motivate treating motion synchronization traffic as sensitive by default.

1 Introduction

Virtual reality (VR) is rapidly moving from a niche platform to a mass-market medium for entertainment, social interaction, education, and remote collaboration [7, 35]. The global VR market is projected to grow from around USD 43.6 billion in 2024 to nearly USD 383 billion by 2033 [15]. This growth rests on a simple premise: achieving convincing immersion requires continuous, high-resolution tracking of the user's body. Therefore, modern VR headsets stream fine-grained motion telemetry, including head pose and hand trajectories, as a core input to rendering and interaction.

However, this telemetry is not fully benign. Prior work shows that head and hand motion trajectories function as behavioral biometrics, enabling user identification and inference of sensitive user attributes [2, 18, 24, 31, 33]. Other work highlights that even avatar-level movement signatures can be linked back to individuals, motivating the need to anonymize or defend these signals [12, 26]. In short, VR user motion is sensitive telemetry, not just "game state".

Multi-user VR creates a further tension between responsiveness and motion data protection. To render avatars and support interaction, applications must synchronize motion updates with remote peers or servers at high rates under strict latency constraints. To keep latency and jitter low, developers commonly use UDP-based unencrypted synchronization channels and build on widely adopted application-layer networking frameworks (e.g., Photon, Unity Netcode, and Mirror [28, 42, 50]). While these frameworks provide transport and serialization logic, they expose flexible serialization interfaces rather than a standardized format for motion data. Instead, each application defines its own application-specific message encoding, typically implemented as a custom binary serialization of motion data into compact payloads. In practice, developers often assume that their message encoding provides a sufficient barrier as payloads look opaque on the wire (see survey findings in Section 7.1). This assumption is inherently appealing: it preserves performance and avoids the engineering friction of deploying payload encryption in latency-sensitive loops.

However, a key question is *whether this "opacity" actually holds against a realistic network adversary.* Most existing VR motion privacy attacks assume privileged access: raw sensor streams, application APIs, instrumented runtimes, or malware on the victim device [18, 31, 33]. While a smaller set of studies considers network-side adversaries, they either use coarse traffic statistics or rely on stronger assumptions, such as co-presence in the same virtual session or manual understanding of message encoding [4, 47, 54]. Closely related wireless side-channel attacks, including VR-Spy [3] and MuKI-Fi [55], show that motion induced by virtual typing can leak through

Approach	No Raw Sensor Required	No App/Device Access Required	No Virtual Co-Presence Required	No Message Encoding Knowledge Required	Real-World App	Available Information	Attack Coverage
Wang et al. [54]	✓	✓	×	✓	✓	Avatar Video (Gaze)	Keystroke Inference
Zhang et al. [60]	✓	×	✓	✓	×	Performance Counters	App & Input Inference
Jarin et al. [18]	×	×	✓	✓	✓	VR Sensors	User Identification
Baldoni et al. [4]	✓	✓	✓	✓	✓	Traffic Volume & Timing	User Identification
Nair et al. [31]	×	×	✓	✓	✓	Head/Hand Motion Telemetry	User Identification
Su et al. [47]	✓	✓	×	×	✓	Motion from Network Traffic	Keystroke Inference
Ours	✓	✓	✓	✓	✓	Motion from Network Traffic	User Identification & Keystroke Inference

Table 1: Comparison of VR privacy attack approaches by observation layer, attacker capability, and attack coverage.

physical-layer wireless observation, such as channel state information (CSI) or beamforming feedback information (BFI). These works motivate our problem by demonstrating that subtle interaction motion can be exploitable, but they assume a different observation layer: wireless sensing of physical motion rather than application-layer VR synchronization traffic. Table 1 summarizes this capability gap. It remains unclear whether a fully passive, on-path observer can recover meaningful motion signals from network traffic alone, without knowledge of message encoding, device compromise, or virtual co-presence.

In this paper, we show that the answer is *"yes"* for a wide range of VR applications. Our key insight is that human motion induces a strong temporal structure that often survives application-specific processing and appears in plaintext UDP payloads as smooth, highly correlated byte-level time series. Leveraging this insight, we design PACMO, a packets-to-motion reconstruction attack that operates on passively observed network traffic and is agnostic to message encoding. Rather than reverse engineering payload formats, the attacker performs one-time black-box profiling of the input-payload relationship by injecting synthetic (sinusoidal) motion waveforms and automatically identifying payload bytes where temporal dynamics match the injected signals. The resulting input-to-payload mapping is then applied to passively captured victim traffic to reconstruct head and hand motion.

Our findings reveal a systematic vulnerability in the VR ecosystem, rather than an isolated implementation flaw. Through an ecosystem-wide survey of real-world VR applications, we find that the uplink channel in multi-user sessions is dominated by sustained, high-frequency UDP traffic, and that payload encryption on this channel is uncommon. We further show that reconstructed motion enables high-impact downstream inference attacks under a passive threat model, including user identification and virtual typing inference, with accuracies up to 96.2% and 69.4%, respectively. These results imply that VR motion updates should be treated as sensitive telemetry transport, and that secure-by-default protections are necessary as VR adoption accelerates.

Our work makes the following contributions:

- We measure the networking ecosystem of 75 real-world VR applications and show that multi-user uplink traffic is overwhelmingly UDP, typically without payload protection on user motion updates.
- We design PACMO, an *encoding-agnostic* packets-to-motion reconstruction attack that uses synthetic (sinusoidal) motion waveform injection and frequency-domain matching to *automatically* map motion channels to payload bytes and reconstruct head and hand motion from passively observed network traffic.
- We demonstrate that reconstructed motion preserves sufficient temporal structure to enable downstream inference attacks, including large-scale user identification and virtual typing inference, under a passive on-path adversary model.
- We complement these results with a developer survey that exposes practical barriers to payload protection, as well as discussions on mitigation strategies suitable for latency-sensitive VR applications.

2 Background

VR applications continuously sense user motion, and in multi-user settings, synchronize it with remote servers or peers to render avatars and enable interaction. In this section, we summarize the lifecycle of VR motion data and explain why the interplay between high-frequency motion updates and latency-sensitive networking creates a privacy attack surface.

2.1 VR Motion Data

Sensing and Pose Representation. VR headsets estimate user motion by tracking the head-mounted display (HMD) and hand controllers using onboard sensors (e.g., IMUs and cameras) and external trackers (in some setups). The main motion data includes 6-degree-of-freedom (6-DoF) poses: 3D position and 3D orientation. These sensors are necessary for all VR applications to render the VR viewport and enable selections and interaction with the user interface. Common VR development engines (e.g., Unity [52] and Unreal [11]) represent pose using vectors for position and quaternions for encoding rotations.

Sampling and Temporal Structure. To maintain responsive interaction, VR systems must update poses at high rates (typically 60–120 Hz) [20, 37]. Since human motion is physically continuous, the resulting data streams are *strongly temporally correlated* time series rather than independent sam-

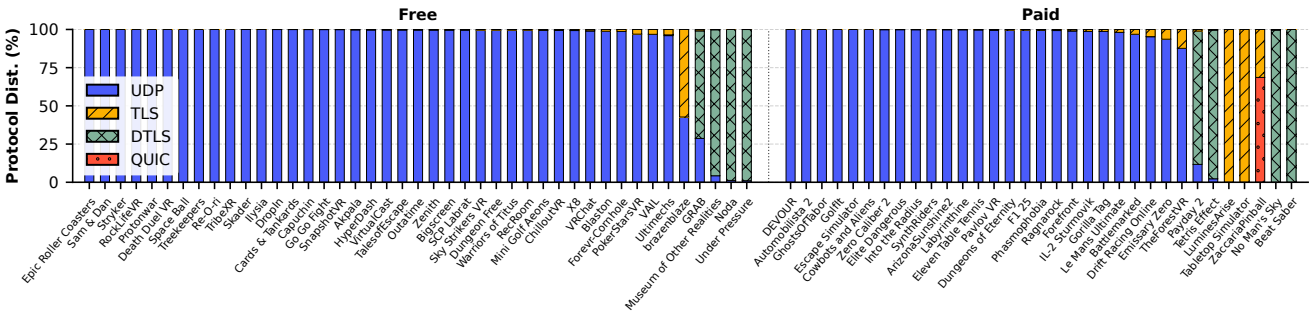


Figure 1: Uplink protocol distribution in multi-user sessions across 75 VR applications, including 43 free and 32 paid applications.

ples. Crucially, even after application-layer processing (e.g., serialization, quantization, and delta encoding that transmits frame changes only), the output signal typically retains these temporal dynamics. This means that unless obfuscated by transport-layer encryption, the motion-bearing network traffic directly carries a predictable temporal signature inherent to human movement, rather than the high-entropy, noise-like characteristics of cryptographic ciphertext.

Transport and Security Choices. The need to synchronize high-frequency motion data in VR applications drives developers to often prioritize low latency and jitter tolerance over standard transport-layer encryption. The decision matrix typically involves the following trade-offs:

- *Latency vs. Reliability (UDP vs. TCP):* Real-time avatar synchronization favors "fire-and-forget" semantics. VR applications overwhelmingly use UDP to send frequent updates. TCP is generally avoided because its reliability mechanisms (e.g., retransmissions and ordered delivery) introduce issues such as head-of-line blocking, causing latency spikes that degrade the immersive experience.
- *Encryption Overhead (QUIC/TLS):* While encrypted transports (QUIC or TLS over TCP) provide confidentiality, they introduce overheads that can be prohibitive for high-rate pose updates. The per-packet computational cost, combined with handshake latency and stateful connection management, can interfere with the tight update loops required for VR rendering. Consequently, they are not the default path in many networking frameworks [40, 41, 51].
- *Implementation Friction (DTLS):* Datagram TLS (DTLS) offers confidentiality with datagram semantics, but it is rarely a "drop-in" solution. It requires explicit configuration and careful engineering of handshake management, certificate handling, and MTU constraints across heterogeneous devices [28, 41, 46], raising the barrier to adoption.

As a result, a significant number of applications transmit motion updates over plaintext UDP, relying entirely on the application-specific message encoding for data representation.

Serialization and Numeric Encoding. With the transport layer commonly left unencrypted, the final layer of obfuscation is data serialization. Before transmission, engines serialize motion states into compact binary payloads. This step is driven by *efficiency*, not security; that is, the goal is to pack complex structures (e.g., vectors and quaternions) into minimal byte streams to reduce bandwidth and CPU overhead. This binary serialization exhibits two critical characteristics relevant to our study. First, *serialization is not encryption*. Although the serialized payload is not human-readable text, it is a structured mapping of memory values to network bytes without cryptographic protection. Second, *the serialization process preserves the correlation between input motion and output bytes,* despite potential scaling or quantization (e.g., float32 vs. integers). Since numeric encoding is deterministic, variations in specific payload offsets are directly driven by the underlying user motion data, effectively exposing users' physical movements in the unencrypted wire format.

3 Empirical Analysis & Motivation

Our attack leverages two empirical observations about real-world networked VR applications: (i) multi-user VR applications commonly use UDP to send motion data without protection; (ii) Even after application-specific message encoding, motion updates often leave structured, temporally coherent traces in packet payloads. In this section, we present these observations and explain why they together enable generalized motion reconstruction attacks from passive traffic.

3.1 Transport Protocols & Payload Protection

Methodology: Ecosystem Survey. We analyzed traffic traces from 75 popular commercial VR applications (spanning free and paid Steam titles) during active multi-user sessions. The sample consists of 43 free applications selected from the Steam top-100 free VR titles by rank order, retaining applications that were executable and supported stable multiplayer sessions. We also selected 32 paid applications drawn from the Steam top-100 paid titles. We focused on

uplink traffic, which carries the high-frequency client updates required for pose synchronization. For each application, we classified packets by transport and security layer, distinguishing plaintext UDP from encrypted protocols (TLS, QUIC, and DTLS).

Prevalence of Plaintext UDP. Figure 1 shows that multi-user uplink traffic is overwhelmingly dominated by UDP. Crucially, only 12 of the 75 applications utilize encrypted transports (DTLS or QUIC) for this channel. The remaining 63 rely on plaintext UDP. This distinction is fundamental: unlike encrypted protocols that render payloads opaque, plaintext UDP leaves serialized motion data exposed to passive on-path observation.

Traffic Characteristics. To examine whether this UDP traffic *likely* carries motion updates, we analyzed the statistical profile of the uplink traffic. Figure 2 plots each application by its average uplink packet rate and packet size for UDP vs. TCP/TLS during the active multi-user phase. The log-scale plot reveals a distinct clustering pattern:

- *Control-like Traffic (yellow markers):* Low-frequency flows (< 20 packets/s), often TCP or TLS, correspond to chat, metadata, or keep-alives [9, 13].
- *Motion-like Traffic (blue markers):* High-frequency flows (> 30 packets/s) are exclusively dominated by UDP. These high-frequency flows also exhibit relatively small packet sizes, a pattern *consistent with* frequent pose updates and state synchronization rather than bulk data transfers.

This statistical distinction provides a strong *behavioral indicator* for identifying the candidate motion-bearing channel: the uplink traffic with sustained high rate and small packets is the most plausible carrier of user motion data.

Observation 1 *Multi-user VR sessions frequently generate high-rate UDP traffic with small payloads in the uplink. Our 75-app survey reveals that payload protection is uncommon: 63/75 applications used plaintext UDP for the relevant uplink channel, while 12/75 used DTLS or QUIC.*

3.2 Feasibility of Motion Recovery

The prevalence of plaintext UDP creates exposure, but exposure alone does not guarantee exploitability. Application-layer message encoding could potentially disrupt the structure required for reconstruction. To examine whether motion signals remain recoverable, we conducted a deep-dive analysis using *Tales of Escape* [39], a representative plaintext-UDP application.

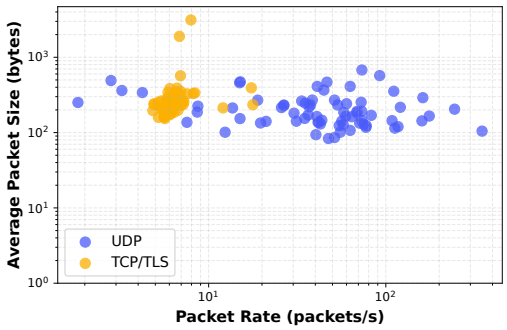


Figure 2: Average packet size vs. packet rate for UDP and TCP/TLS across 75 applications. Active multi-user sessions cluster in a high-rate, (relatively) small-payload region, consistent with continuous pose update and state synchronization.

Methodology: Controlled Waveform Injection. We utilized NVIDIA VCR [38] to conduct controlled replay experiments. NVIDIA VCR enables consistent replay of specific head and hand trajectories across the axes of motion sensors. To isolate motion signals from human noise and other game data, we held all degrees of freedom fixed except for a single target channel (e.g., head position along the x-axis). We then injected a clean *sinusoidal waveform* into that channel and captured the resulting network traffic.

(a) *Localization:* Figure 3 visualizes a byte-wise change map of the UDP payloads under these controlled inputs. Under the static condition, only a negligible set of bytes changes (likely metadata). However, when we vary a single motion axis, specific byte ranges exhibit systematic variation. Crucially, these changes are *axis-specific*: perturbing one axis affects a disjoint set of byte offsets compared to the other axes. This indicates that motion channels are serialized into fixed regions of the payload rather than being dispersed or randomized.

(b) *Reconstruction:* To validate that these changing bytes carry actual motion data, we interpreted the identified offset as a numeric stream. Decoding the field identified in Figure 3 (as float32 little-endian) yields a time series that faithfully tracks the injected sinusoidal waveform (see Figure 4). While the absolute scale may differ due to application-specific scaling or coordinate transforms, the *temporal structure* and periodicity remain intact. This confirms that the serialization process preserves the underlying motion dynamics.

Observation 2 *Even after application-layer message encoding, motion updates preserve distinctive temporal structures at fixed payload offsets. This allows a passive observer to identify and reconstruct motion signals from plaintext UDP packets without the knowledge of packet semantics.*

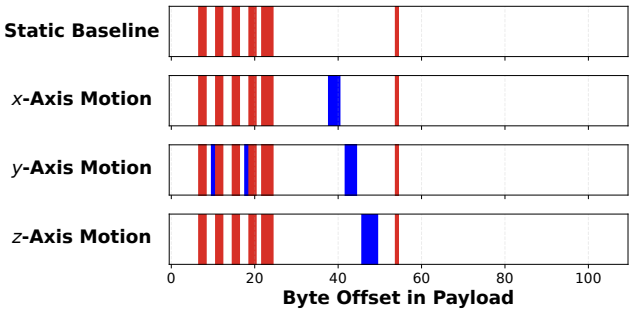


Figure 3: Visualization of motion signal isolation in *Tales of Escape* UDP payloads. Each strip represents the byte-level variance of the payload under a specific condition. *Red bytes* represent global variables (e.g., timestamps, sequence numbers) that fluctuate in all conditions regardless of user motion. *Blue bytes* highlight *axis-specific motion signals*. Note that distinct byte offsets become active (blue) only when the corresponding axis (x, y, or z) is modulated.

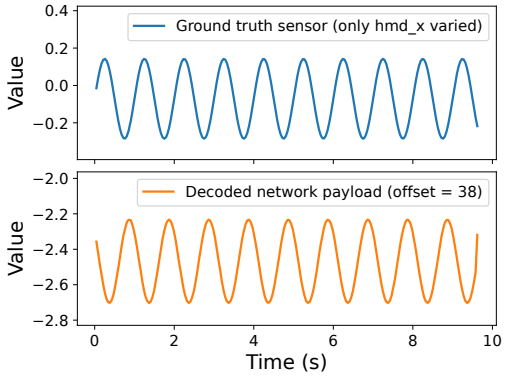


Figure 4: Validation of motion leakage. Decoding an identified payload field (orange) tracks the injected sinusoidal head-motion input (blue), confirming that plaintext UDP payloads carry motion-like trajectories.

Implication for Attack Design. These findings reveal two generic anchors for an encoding-agnostic attack: (i) motion-bearing fields vary smoothly and continuously, and (ii) they are located at fixed offsets. In Section 5, we leverage these properties to design an automated pipeline that profiles these temporal correlations to map and then reconstruct motion fields across various networked VR applications, which are unknown to the attacker.

Challenges and Roadmap. While this case study confirms feasibility, elevating it to a generalizable threat requires overcoming two key challenges. First is the *semantic gap*: a real-world passive attacker sees only raw bytes and does not know which numeric encoding scheme (e.g., float32, int16, or custom compression) each application uses. Second is *scale*: determining whether this vulnerability is systematic across the diverse ecosystem of VR applications requires more than manual analysis of a single application. A critical question remains: *Can an adversary automate the discovery and decoding of these motion channels across various unknown message encodings?* To address this, we design an attack pipeline that leverages the observed temporal structure to identify motion fields and reconstruct motion signals without semantic knowledge of the underlying message encoding (see Section 5).

4 Threat Model

Adversary and Vantage Point. We consider a passive, on-path adversary who monitors network traffic generated by a victim's VR session. The adversary does *not* require compromising the victim's device, does not install malware, and does not participate in the victim's virtual session (e.g., as another avatar). Instead, they observe traffic from realistic network vantage points, such as a compromised Wi-Fi access point, a local enterprise gateway, or an ISP monitoring node.

Capabilities and Assumptions. We assume the adversary possesses the following capabilities:

- *Traffic Capture:* The adversary can capture packet metadata (including endpoints, timing, and size), and crucially, the full payload bytes of plaintext UDP packets. From this traffic, the adversary can also infer the running VR application based on server endpoints and protocol-level identifiers (e.g., DNS queries or TLS SNI field revealing a domain such as 'backend-hyperdash.be' for Hyper Dash)
- *Flow Isolation:* Drawing on the characteristics identified in Section 3, the adversary can distinguish motion-bearing flows (sustained high-rate, small-payload UDP) from background traffic.
- *Offline Profiling:* We assume the adversary can download the same target VR application. They can run this application locally to perform one-time offline profiling (e.g., using input injection tools like NVIDIA VCR) to learn the application's payload structure.

However, the adversary cannot:

- Decrypt flows protected by DTLS, QUIC, or TLS.
- Access proprietary server-side code, message encoding specifications, or perform dynamic instrumentation on the *victim's* active device.
- Modify, inject, or drop packets (active attacks are out of scope).

Adversary Goals. The primary goal is to reconstruct high-resolution motion trajectories (head and hand 6-DoF poses)

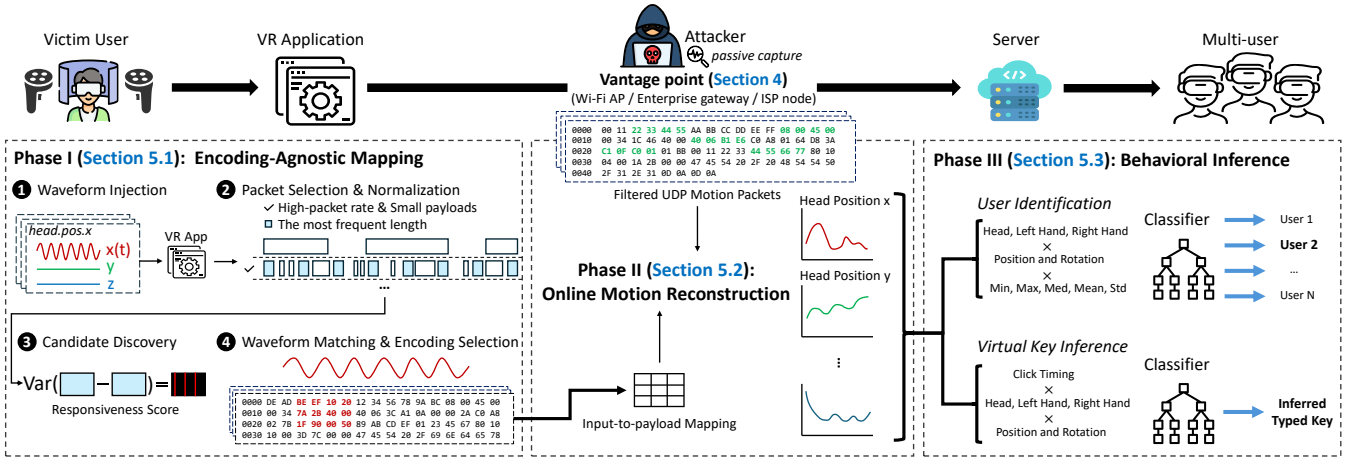


Figure 5: Attack overview. A passive on-path adversary captures VR traffic and isolates motion-bearing UDP flows. Using offline profiling with controlled injection, the adversary derives an input-to-payload mapping. This mapping is applied to victim payloads to reconstruct motion trajectories, enabling downstream user identification and typing inference.

from the captured plaintext UDP payloads without prior knowledge of the message encoding. Using these reconstructed trajectories, the adversary aims to perform downstream inference attacks, specifically: (i) *User Identification* via behavioral biometrics, and (ii) *Sensitive Input Inference*, such as reconstructing text or PINs typed on an in-application virtual keyboard.

Scope and Limitations. Our threat model relies on the visibility of payload patterns. If an application employs transport-layer encryption (e.g., DTLS or QUIC) or end-to-end payload encryption, then our attack technique is not applicable. In such encrypted scenarios, an adversary is limited to coarse-grained traffic analysis (e.g., based on packet size and timing), which is outside the scope of this work. Sensitive input inference in this work is limited to application-visible, in-application virtual keyboards. System-level inputs (e.g., OS virtual keyboard, SteamVR keyboard) are out of scope, as they may not be reflected in application traffic.

5 PACMO: System Design

PACMO is an encoding-agnostic attack that reconstructs VR motion trajectories from passive multi-user network traces. It builds on two empirical findings from Section 3: (1) many multi-user VR sessions transmit sustained high-rate UDP, often without payload protection; and (2) motion updates preserve smooth temporal structure and frequently appear at fixed payload offsets. Given plaintext UDP payload bytes, our goal is to reconstruct head and hand pose trajectories without encoding specifications, packet format knowledge, or application binary analysis.

Key Insight. Rather than reverse engineering packet formats, PACMO treats the application as a black box and performs one-time offline system identification. By injecting controlled motion waveforms during profiling, we can automatically locate payload regions whose decoded time series matches the injected signal in the frequency domain. This yields an input-to-payload mapping that generalizes to passively captured victim traffic for the same application.

Overview. Figure 5 summarizes our pipeline, which has three phases: (i) automated input-to-payload mapping (one-time per application), (ii) motion reconstruction from victim packets using the learned mapping, and (iii) downstream behavioral inference on reconstructed motion.

5.1 Phase I: Encoding-Agnostic Mapping

Goal. Given an application with unknown packet formats, Phase I outputs a mapping table that links motion channels (head/hand position and rotation components) to (a) payload byte offsets, (b) field widths, and (c) likely encodings (and endianness). This step is application-specific but not user-specific, and it needs to be performed only once per application.

5.1.1 Step 1: Controlled Waveform Injection

Profiling Setup. The attacker runs the target application in a controlled environment and uses an input-simulation tool. We use NVIDIA VCR, a VR capture-and-replay tool by NVIDIA that enables deterministic recording and playback of VR sessions by replaying VR tracker data to the application through the SteamVR. This allows us to inject motion into

exactly one channel at a time (e.g., head position along the x-axis), while holding all other channels constant. The injected signal is a periodic waveform $x(t)$ (e.g., sinusoids), which produces a distinctive spectral signature in the resulting traffic. This periodicity preserves dominant frequency components even under transformations like scaling, quantization, or delta-encoding, enabling robust matching without knowledge of the specific message encoding.

5.1.2 Step 2: Packet Selection and Normalization

From the profiling trace, the attacker first isolates the UDP flow consistent with continuous state synchronization. Within that flow, the attacker groups packets by payload length and focuses on the most frequent length(s), which typically correspond to a recurring state-update message. This reduces interference from control messages and heterogeneous packet types.

5.1.3 Step 3: Candidate Discovery via Responsiveness

Consider the selected packet group. Let $b_k[i]$ denote the byte at payload offset i in the k-th packet, and let $b_k[i:i+w-1]$ denote a window of w consecutive bytes starting from offset i in the k-th packet.

Responsiveness Score. Bytes unrelated to the injected channel tend to remain constant or change weakly over time. We therefore identify candidate payload offsets by detecting packet-to-packet changes. Specifically, for each byte offset i, we compute successive differences:

$$R(i) \;=\; \mathrm{Var}_k\big[\, b_k[i] - b_{k-1}[i] \,\big], \tag{1}$$

where $\mathrm{Var}_k[\cdot]$ denotes the variance computed over the packet index k. Contiguous responsive bytes with elevated $R(i)$ are then grouped into candidate fields and evaluated under different decoding hypotheses in the subsequent step.

Decoding Hypotheses. To avoid message encoding assumptions, we try a small set of common numeric interpretations for $\phi(\cdot)$, such as little-endian `float32`, `int16`/`uint16`, and `int32`/`uint32`. We also consider a delta view, where $\phi(\cdot)$ is applied to successive differences to handle formats that transmit increments.

5.1.4 Step 4: Waveform Matching and Encoding Selection

Frequency-domain Matching. For each retained window (i, w) and decoding hypothesis ϕ, we obtain a candidate time series

$$y_{i,w,\phi}(k) \;=\; \phi\big(b_k[i:i+w-1]\big). \tag{2}$$

We compare $y_{i,w,\phi}$ to the injected waveform $x(t)$ using a frequency-domain cosine similarity that is robust to scaling and phase:

$$S(i,w,\phi) \;=\; \frac{|\mathcal{F}\{x\}| \cdot |\mathcal{F}\{y_{i,w,\phi}\}|}{\big\|\mathcal{F}\{x\}\big\|\,\big\|\mathcal{F}\{y_{i,w,\phi}\}\big\|}, \tag{3}$$

where $\mathcal{F}\{\cdot\}$ is the discrete Fourier transform, $|\cdot|$ denotes element-wise magnitude of the Fourier coefficients, and $\|\cdot\|$ denotes the L_2 norm.

Selecting Fields. For each injected channel, we select the window and decoding hypothesis that maximize $S(i,w,\phi)$, subject to basic sanity checks (fixed offset within the selected packet group, non-degenerate variance and consistent behavior across multiple profiling runs). This yields the payload offset range and a likely encoding for that channel. We optionally fit an affine map $x \approx a y_{i,w,\phi} + c$ to the profiling trace to estimate scale and offset for reconstruction, without claiming to reconstruct exact semantic units (e.g., the application's original coordinate system).

5.2 Phase II: Online Motion Reconstruction

Goal. Phase II reconstructs head and hand motion trajectories as time-series signals. The attacker executes the following steps on the captured UDP stream:

(a) *Packet filtering:* The attacker selects the victim UDP flow and packet group that match the profiled stream (endpoints and ports when visible, plus the high-rate small-payload profile and payload length clustering). Packets outside the selected group are ignored.

(b) *Field extraction and decoding:* For each retained packet, the attacker extracts the mapped byte windows and decodes them using the selected encoding hypotheses (and any fitted affine normalization from profiling). This yields per-channel sequences over packet index.

(c) *Temporal alignment and cleanup:* The attacker aligns decoded samples using packet arrival time and resamples to a uniform rate. Packet loss and jitter are handled by applying linear interpolation only when a 1-second window contains no samples, to ensure at least one value per window.

5.3 Phase III: Behavioral Inference

Goal. Phase III uses reconstructed motion as an input signal for privacy attacks, which are typically assumed to require privileged sensor access in existing works.

Downstream Tasks. We study two representative inferences: (1) user identification from motion-derived behavioral

signatures, and (2) virtual typing inference (e.g., PINs, passcodes, and sentences) from motion patterns during virtual keyboard interactions. At a high level, the attacker converts reconstructed trajectories into features capturing motion dynamics and timing, and trains classifiers such as LightGBM on profiled data. Importantly, reconstructed motion dynamics can serve as shared inputs to multiple downstream privacy attacks (e.g., identification and keystroke inference).

Implication. PACMO shows that plaintext UDP motion updates can expose a motion reconstruction channel to passive observers. This moves the threat beyond traffic analysis based on metadata only; it restores motion-based time series that directly support inference-based privacy attacks.

6 Evaluation

We evaluate PACMO following the pipeline in Section 5. Our primary concern is selection bias, as demonstrating attacks on a limited set of applications could be dismissed as cherry-picking. To address this, we adopt a two-track evaluation strategy.

First, we evaluate *breadth* across the broader ecosystem of vulnerable applications identified in Section 3. Second, we evaluate *depth*: recognizing that sine-wave inputs are simplistic, we conduct an IRB-approved user study on five representative applications to validate that reconstruction fidelity is sufficient for downstream inference on natural motion.

6.1 Experimental Setup

Applications. For the breadth track, we use the larger set of real-world applications from Section 3. These applications are treated as black boxes: we do not modify binaries, network libraries, or runtime configuration.

For the depth track, we select five real-world VR applications spanning social, sport, shooting, rhythm, and puzzle genres: Tales of Escape [39], Hyper Dash [48], Big Screen Beta [1], Eleven Table Tennis [14], and Ragnarock [53]. All support multi-user interaction and transmit frequent motion-related updates over UDP in our environment. We chose these five applications as diverse case studies to cover distinct interaction categories rather than as a statistically representative sample. Experiments were conducted using a Meta Quest Pro headset. To maintain multi-user sessions, a second headset (Meta Quest 3) was used by the experimenter to join the sessions, solely for experimental control.

User Study and Tasks. We recruited 23 university-affiliated students via a mailing list under an IRB-approved protocol, with $10 compensation per participant. The participant pool included 12 male and 11 female participants, ages 19–37, with 3 participants not disclosing age. Self-reported VR experience

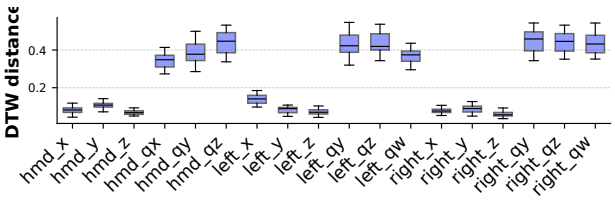


Figure 6: Distribution of Derivative DTW (DDTW) distances across 75 tested applications.

was limited for most participants: 17 had never or rarely used VR, 4 used VR occasionally, and 2 used VR frequently. Participants interacted with each application for approximately 10 minutes under natural gameplay or social interaction. For virtual typing inference, participants additionally performed three tasks using *Big Screen*'s in-application virtual keyboard: (i) random 4-digit PIN entry, (ii) passcodes sampled from Ignis-10K, and (iii) sentence typing derived from frequent English words [30]. These tasks capture sensitive inputs where leakage has clear security impact.

Ground-Truth Motion (Evaluation Only). During user study sessions, we recorded sensor-level head and hand pose streams (OpenVR and, when applicable, NVIDIA VCR recordings). This ground truth is used only to quantify reconstruction fidelity; it is not assumed available to the adversary.

Traffic Capture and Phase I Profiling We capture bidirectional traffic using a dedicated gateway PC placed between the VR device and the router. Each live session used two headsets on the same lab Wi-Fi network. We record packet timings, sizes, and (when present) plaintext UDP payload bytes. For flow isolation, we select the application-associated UDP flow using endpoint/port information and then apply the profiled high-rate, small-payload, payload-length-cluster signature described in Section 5.2. In additional checks with concurrent background web browsing and video streaming during capture, unrelated non-VR flows did not satisfy both the application endpoint/port filter and the profiled motion-packet signature. These checks are limited to the background activities we tested and do not constitute a fully representative model of all real-world network environments. Separately, for each evaluated application, we ran Phase I profiling once on an adversary-controlled device to derive the input-to-payload mappings, as described in Section 5.1.

6.2 Phase II: Online Motion Reconstruction

We evaluate the reconstruction performance of PACMO using a two-track strategy: assessing ecosystem-wide feasibility via controlled injection (*Breadth*) and measuring reconstruction fidelity on natural human motion (*Depth*).

6.2.1 Breadth: Ecosystem-Level Feasibility under Controlled Injection

To validate ecosystem-wide risks, we apply Phase I and II across the vulnerable applications identified in Section 3.

Encoding-Agnostic Validation. For each application, we inject periodic waveforms into single DoFs to learn mappings (Phase I) and apply them to held-out traces (Phase II). This lightweight experiment tests whether motion-bearing fields are reliably decoded across diverse, unknown formats without assuming Encoding semantics.

Result Summary. Figure 6 summarizes reconstruction stability across the tested applications. To quantify similarity between the injected waveform and the reconstructed signal, we use *Derivative Dynamic Time Warping (DDTW)* distance [23]. Unlike standard Euclidean distance, Derivative DTW is robust to temporal shifts and affine transformations (e.g., constant offsets) introduced by application-level coordinate systems, and is a common distance metric for motion capture data.

The results reveal two key patterns. First, position channels (e.g., *hmd_x*, *y*, *z*) consistently exhibit low DDTW distances, confirming that our pipeline reliably recovers translational motion across the ecosystem. Second, rotation channels(e.g., *hmd_q*) show comparatively higher DDTW distances. This is an artifact of our *waveform injection methodology* rather than a failure of reconstruction; injecting a full-range continuous sine wave forces rotation values to traverse the entire angular space (0° to 360°), frequently triggering mathematical discontinuities such as Euler angle warping. These artificial *discontinuities* inflate the DDTW metric.

In contrast, as will be demonstrated in the *Depth* evaluation (Section 6.2.2), natural human motion is biomechanically constrained to a local manifold and rarely triggers such discontinuities, resulting in significantly higher reconstruction fidelity when computed on real motion data.

6.2.2 Depth: Reconstruction Fidelity for Natural Human Motion

Waveform injection establishes that a mapping can be learned and applied broadly, but injected motion is simpler than real interaction; to test *practical leakage*, we evaluate Phase II on natural human motion using the five representative applications and our user study captures.

Metric. We quantify reconstruction fidelity using the same DDTW metric described above. We calculate the distance between the reconstructed (from network payloads) and the sensor-level ground truth.

Results. Table 2 reports DDTW distances across head and controller channels for the five applications. Note that some applications transmit only three quaternion components and reconstruct the fourth (e.g., q_x) client-side; when applicable, we similarly re-compute the missing component. Head translation consistently reconstructs with low DDTW, aligning with the ecosystem-level stability in Figure 6. Similarly, controller positions and rotation components generally reconstruct with high fidelity.

Signal	Tales of Escape	Hyper Dash	Big Screen	Eleven Table Tennis	Ragnarock
HMD Position					
x	0.309	0.179	0.218	0.156	0.284
y	0.215	0.263	0.241	0.197	0.309
z	0.219	0.289	0.264	0.183	0.296
HMD Rotation					
q_x	0.200	0.289	0.233	0.164	0.278
q_y	0.178	0.286	0.219	0.148	0.267
q_z	0.238	0.317	0.291	0.205	0.318
Left Hand Position					
x	0.246	0.294	0.257	0.176	0.302
y	0.249	0.241	0.228	0.154	0.287
z	0.194	0.228	0.214	0.139	0.271
Left Hand Rotation					
q_y	0.234	0.625	0.305	0.192	0.331
q_z	0.273	0.201	0.236	0.168	0.289
q_w	0.217	0.240	0.223	0.157	0.281
Right Hand Position					
x	0.264	0.294	0.269	0.181	0.314
y	0.263	0.246	0.241	0.169	0.298
z	0.280	0.239	0.258	0.193	0.321
Right Hand Rotation					
q_y	0.210	0.354	0.232	0.162	0.287
q_z	0.244	0.253	0.219	0.149	0.274
q_w	0.217	0.223	0.227	0.158	0.283

Table 2: DDTW distances between reconstructed and ground-truth motion across 5 applications (lower is better) for a representative participant. We highlight values > 0.5 with a pink background, indicating localized reconstruction outliers.

We observe a localized outlier in the left hand q_y channel of Hyper Dash (0.625). We believe this stems from application-specific artifacts, such as *aggressive quantization* or *imperfect field isolation* during Phase I, which reduce the temporal smoothness of the reconstructed signal. Crucially, however, Phase III does not require *perfect recovery* of every channel. As demonstrated in Section 6.3.3, privacy leakage persists even under partial observability. Consequently, isolated reconstruction imperfections do not meaningfully limit the adversary; they merely reduce efficiency, not feasibility.

6.3 Phase III: Downstream Privacy Attacks

We now evaluate Phase III, using the motion trajectories reconstructed in Phase II to perform concrete privacy attacks.

Application	User Study (Intra-Session)				Beat Saber Dataset (Inter-Session)			
	Full Pkt.	Pkt Feat.	PACMO	Raw Sensor	Full Pkt.	Pkt Feat.	PACMO	Raw Sensor
Tales of Escape	52.2 (36.2)	65.2 (44.0)	**69.6 (62.4)**	73.9 (68.3)	15.8 (12.6)	27.4 (22.1)	**96.2 (63.0)**	
Hyper Dash	21.7 (30.4)	26.1 (31.2)	**69.6 (62.6)**	73.9 (77.4)	18.8 (15.2)	31.6 (26.4)	**92.6 (66.2)**	
Big Screen	52.2 (44.4)	65.2 (56.2)	**73.9 (62.4)**	78.3 (72.0)	14.6 (11.8)	27.0 (21.2)	**91.4 (61.7)**	96.6 (70.4)
Eleven Table Tennis	39.1 (46.1)	43.5 (47.3)	**69.6 (59.1)**	82.6 (67.0)	17.2 (13.9)	29.8 (24.3)	**94.8 (64.9)**	
Ragnarock	52.2 (54.0)	69.6 (60.0)	**91.3 (90.4)**	100.0 (99.7)	16.0 (12.7)	28.4 (23.1)	**95.6 (63.0)**	

Table 3: User identification accuracy (%) across five applications. Results are reported as *Per-user (Per-sample)*. The table is split into two blocks: Scenario 1: live User Study results (left) and Scenario 2: large-scale Beat Saber replay results (right).

6.3.1 Downstream Attack I: User Identification

The first attack aims to re-identify users based on their unique motion signatures.

Baselines and Model. We compare against two network-only baselines and one upper bound: (i) *full-packet bytes* (raw UDP payload); (ii) *packet statistics* (traffic features like sizes and rates [4]); and (iii) *raw sensor* (pre-serialization sensor streams [31]). Following prior work on a large-scale VR dataset [31], we segment data into 1-second windows and extract summary statistics from each window, which includes mean, standard deviation, min/max, and median of each motion channel. By leveraging these summary statistics, we obtain a compact representation that captures user-specific behavioral motion signatures within each window. We then adopt the same feature representation and LightGBM-based classification pipeline as Nair *et al.* [31], using their standard preprocessing (z-score normalization) and configuration. Per-sample accuracy is computed over individual 1-second windows. Per-user accuracy is computed by aggregating all per-window predictions for a user within the evaluated session and assigning the user label by majority vote.

Scenario 1: Live User Sessions (Intra-Session). We first evaluate identification on live captures from our user study (23 participants). Table 3 (left) reports accuracy using a temporal split within the same session (70% train / 20% validation / 10% test). We use a single-session setting because each participant contributes only one live session per application in our user study. While this does not model long-term re-identification across sessions, it provides an initial validation of feasibility on natural user motion traces; multi-session robustness is evaluated separately in Scenario 2. We intentionally did not control within-session activity: in social applications, participants interacted naturally, so some users walked or interacted more while others were more passive. This follows prior VR identification work that includes social VR applications without predefined tasks to capture unscripted movement [18]. Thus, activity-level variation is part of the realistic behavioral distribution we aim to capture, and successful identification under this variation strengthens ecological validity.

PACMO achieves strong per-user accuracy ranging from 69.6% to 91.3%, consistently outperforming baselines using packet-level features. The gap is particularly evident in dynamic games like Hyper Dash (69.6% vs. 21.7% and 26.1%). Notably, Ragnarock exhibits the highest leakage (91.3%), closely approaching the raw-sensor upper bound (100%), suggesting that the repetitive, high-velocity motions in rhythm games amplify distinctive biometric patterns.

We note that this intra-session setting is intentionally favorable to packet-based baselines: because training and testing data come from the *same continuous session*, raw-byte classifiers can exploit transient session-specific artifacts. We therefore treat this scenario as an easier case for network-only approaches. Even under this favorable condition, however, our reconstructed motion consistently outperforms packet-level baselines, indicating that recovery of motion dynamics provides substantially richer behavioral signals. A more realistic and challenging multi-session setting, where such transient artifacts do not generalize, is evaluated separately in Scenario 2.

Scenario 2: Large-Scale Multi-Session (Inter-Session). To stress-test scalability and temporal robustness, we extend our evaluation using the public *Beat Saber [5] motion dataset* from BOXRR-23 [32]. Unlike our single-session user study, this dataset provides *longitudinal 6-DoF pose streams (3D position x, y, z and rotation quaternions)* for the headset and controllers across multiple days. This allows us to simulate a realistic surveillance scenario where an adversary observes the same user across time-separated sessions. We replay the motion traces through each evaluated application to generate app-specific network traffic, then run PACMO end-to-end, including Phase I mapping, Phase II reconstruction, and Phase III identification. We follow the re-identification protocol of the dataset authors by randomly sampling 500 users from the corpus of over 50,000 and replay their traces for each application to generate traffic.[1]

[1] While Beat Saber's kinematics differ from app-specific behaviors, this cross-app replay is necessary to validate scalability given the lack of app-specific public datasets. Crucially, it demonstrates that PACMO extracts

Table 3 (right) summarizes the results. PACMO maintains high accuracy (91.4–96.2%), comparable to the Raw Data upper bound. In sharp contrast, network-only baselines degrade significantly (Full Packet: ~16%, Packet Features: ~28%) because session-specific artifacts do not generalize across time. This confirms that our reconstructed trajectories capture persistent behavioral biometrics, whereas metadata-based baselines rely on ephemeral signatures. Additionally, per-user accuracy (>91%) is consistently higher than per-sample accuracy (~63%), indicating that accumulating evidence over a session allows the adversary to identify users with high confidence.

6.3.2 Downstream Attack II: Virtual Typing Inference

The second, stronger attack involves inferring specific inputs on a virtual keyboard. Unlike identification, which relies on global behavioral patterns, typing inference requires preserving short-lived micro-motions and precise event timing around each key selection. Our typing attack targets in-application keyboards, not system-level keyboards or OS UI. In our environment, when the SteamVR system keyboard or headset settings UI was opened, application-visible motion/avatar updates stopped. This behavior is consistent with the mitigation efforts in which SteamVR restricts applications from accessing hand motion data while the Steam keyboard is open, preventing typing-related hand motion from being synchronized to remote users [47]. We therefore treat in-application virtual keyboards as the primary attack surface for typing inference, and system-level keyboards and OS UI as outside our scope.

Methodology and Tasks. To perform this attack, we first segment the continuous motion stream into keystroke events. We detect key selections by decoding the controller trigger (click) signal, which is transmitted as a separate channel in the same UDP motion streams, using the Phase I mapping and applying a simple thresholding rule.

For each detected click, we extract a fixed-length motion window centered at the timestamp, and label the key user typed. We follow the VR typing inference setup used in prior works [45, 47], and adopt comparable task definitions and evaluation protocols for PIN, passcode, and sentence-level input inference. (Note: An attacker can train this model by profiling their own device by typing known strings; victim labels are not required for training).

We evaluated three scenarios reflecting common VR use cases: (i) *4-digit PIN entry* (discrete inputs); (ii) *Passcodes* sampled from Ignis-10K (alphanumeric sequences); and (iii) *Sentence typing* (complex, variable-length input consisting of lower-case alphabet characters or spaces). We compare against the same network-only baselines (Full Packet and Packet Features) and the Raw Sensor upper bound.

behavioral signatures from *any* complex motion stream and encoding.

Task (*N*)	Full Pkt	Pkt Features	**PACMO**	Raw Sensor
4-PIN (170)	12.4%	14.7%	**69.4%**	88.2%
Passcode (85)	7.1%	9.4%	**63.5%**	83.5%
Sentence (85)	4.7%	7.1%	**56.5%**	78.8%

Table 4: Key inference accuracy (%) across three typing tasks. *N* denotes the total number of keystroke samples evaluated. PACMO consistently outperforms network-only baselines.

Results. Table 4 summarizes the inference accuracy for character-level keyboard inputs. PACMO achieves practical recovery rates across all tasks: 69.4% for PINs, 63.5% for passcodes, and 56.5% for sentences. In sharp contrast, network-only baselines remain near chance level (4.7–12.7% and 7.1–14.7%).[2] This huge performance gap confirms that packet timing, traffic statistics, or raw UDP payloads alone are insufficient for keystroke-level inference; the distinguishing signal comes entirely from the motion trajectories reconstructed from payload bytes.

We observe a clear increase in task difficulty across scenarios. PIN entry yields the highest accuracy due to discrete selections and limited ambiguity. Sentence typing is the most challenging, introducing complex temporal dynamics (e.g., pauses) and richer character distributions. Despite this complexity, the 56.5% accuracy on sentences demonstrates that network-derived reconstructions preserve fine-grained interaction cues even under variable typing rhythms.

Compared to the raw-sensor upper bound, the observed performance gap (e.g., 69.4% vs. 88.2% on 4-PIN) reflects unavoidable information loss due to application-level serialization and quantization, as well as reduced sampling rates.

6.3.3 Robustness and Ablations

Finally, we characterize the boundary conditions of our attack by evaluating robustness under data degradation. *We rely on identification accuracy as the primary metric* because it provides a continuous measure of signal quality across all users. While specific tasks like typing inference (Section 6.3.2) inherently require specific channels (e.g., hand tracking), identification allows us to quantify how much behavioral information survives when some features are missing or observation time is limited. These constraints reflect fundamental limitations of the reconstructed motion signal that would impact any downstream adversary. Moreover, since different applications expose different subsets of motion data in practice, we begin by examining the impact of partial motion observability.

[2] While Su *et al.* [47] report higher accuracy (e.g., 96.8% for numbers) using a semantic extraction approach, it relies on manual reverse-engineering of application binaries (e.g., decompiling to map serialization fields) and geometric calibration of virtual keyboards. We exclude this as a direct baseline because PACMO operates under a stricter black-box threat model without manual binary analysis or geometric pre-measurement.

Feature Subset	PACMO	Raw Data
Full Motion (Head + Hands)	96.2%	96.6%
Head Only (Position + Rotation)	82.4%	87.6%
Head Only (Position Only)	28.4%	43.8%
Head Only (Rotation Only)	19.0%	36.4%
Hands Only (Position + Rotation)	40.8%	68.6%

Table 5: Impact of partial motion observability on identification accuracy in *Tales of Escape* using replayed motion data.

Impact of Partial Motion Observability. Table 5 quantifies identification accuracy when the attacker recovers only a subset of motion features, reflecting practical scenarios where applications expose different tracking modalities (e.g., head-only experiences or games that use a single controller). We observe that *head motion alone* (position + rotation) retains the vast majority of the biometric signal, achieving *82.4%* accuracy compared to 96.2% for full motion. Removing either component leads to substantial degradation. Using head position only reduces accuracy to 28.4%, while rotation alone further reduces it to 19.0%, indicating that both components contribute, with head position providing stronger cues due to absolute transitions (e.g., y-axis of head position correlated with user height), while rotation complements this by capturing individual head-orientation dynamics. Hand features alone (40.8%) provide weaker but non-trivial leakage.

The weaker hands-only result does not imply that encoding-agnostic reconstruction fundamentally fails on hand channels. As Table 2 shows, per-channel reconstruction fidelity is generally comparable across head and hand channels, with only localized outliers. The larger PACMO-vs-raw gap in the hands-only setting emerges downstream at the classifier. To test this, we additionally evaluated a deep-learning-based Gated Recurrent Unit (GRU) sequence model on the hands-only setting. Because sequence models are better suited to exploit temporal hand dynamics [22], the GRU model improved the underlying performance on hand features and successfully reduced the PACMO-vs-raw gap to 5.8 percentage points (75.1% for PACMO vs. 80.9% for raw motion). This suggests that the performance gap stems from the downstream classifier's limitations with sequential data, rather than a reconstruction flaw. Further classifier optimization is outside our scope. Overall, our findings imply that the attack is *resilient*: user privacy is compromised even if the attacker fails to recover specific channels (e.g., on hand tracking information).

7 Ecosystem Analysis and Mitigation

Given our findings that many VR applications transmit motion over plaintext UDP (Section 3 and Section 6), a key question remains: *Why is payload protection uncommon?* To answer this, we surveyed developers to identify practical barriers and inform defenses compatible with latency-sensitive VR.

7.1 Why Plaintext Persists: Survey Findings

We received responses from 7 development teams (out of 50 contacted), revealing three consistent themes explaining the prevalence of plaintext UDP.

- *Engineering friction and library gaps:* Six teams confirmed their reliance on plaintext UDP, citing engineering complexity as the dominant barrier. Integration of DTLS is rarely a "drop-in" solution; developers cited specific hurdles including handshake management, MTU fragmentation constraints, and cross-platform interoperability. Crucially, teams noted that their networking middleware lacks encryption support "out of the box"[3], and retrofitting a bespoke cryptographic layer into a custom binary protocol is viewed as a stability risk. This validates our threat model: custom serialization is used for efficiency, not security.
- *Low perceived risks:* Three teams indicated that motion data is viewed as *low-risk*. Developers emphasized that sensitive account data is already handled over HTTPS/TLS, while UDP channels are reserved for pose, voice, or gameplay state. As a result, motion traffic is not treated as sensitive telemetry, and passive on-path threats are not considered a dominant design driver by some developers.
- *Performance and operational simplicity:* Five teams highlighted latency, jitter, and development velocity as primary constraints. Even if encryption is not the computational bottleneck, developers remain cautious about adding protocol layers that complicate packetization, encapsulation, timing control, and debugging for high-frequency motion updates.

Put together, the survey indicates that plaintext UDP persists due to a combination of (i) lacking library support, (ii) non-trivial integration effort, (iii) performance and debugging concerns, and (iv) low perceived risk of motion telemetry. This motivates defenses that either make effective payload protection the default or reduce the attack surface when full DTLS deployment is infeasible or inconvenient.

7.2 Mitigation Strategies

The most direct defense is to encrypt motion-bearing UDP payloads end-to-end (e.g., using DTLS), which blocks Phases I/II of PACMO by removing access to payload bytes. Our survey suggests that the main barrier is not cryptography itself but integration cost and tooling. This motivates ecosystem-level action: widely used application-layer networking frameworks should be shipped with secure modes that are easy to enable and preferably default-on for motion updates, with performance-tuned presets and debugging support.

[3] One developer noted: "If the protocols offered this working out of the box without significant engineering effort, we'd be able to do this quickly, as that would lower the engineering barrier."

Application	PACMO	*Restricted Dynamic*	*Full Dynamic*
Tales of Escape	69.6	34.8 (-34.8)	8.7 (-60.9)
Hyper Dash	69.6	39.1 (-30.5)	13.0 (-56.6)
Big Screen	73.9	47.8 (-26.1)	21.7 (-52.2)
Eleven Table Tennis	69.6	43.5 (-26.1)	17.4 (-52.2)
Ragnarock	91.3	52.2 (-39.1)	21.7 (-69.6)

Table 6: User identification accuracy (%) under different serialization settings across five XR applications. We compare PACMO against restricted and full dynamic serialization.

Lightweight Fallback: Dynamic Serialization. When full DTLS deployment is infeasible or inconvenient, we consider a fallback mitigation that directly targets our key enabling condition: *fixed payload offsets* that preserve temporal structure across packets. In our attack, fixed byte offsets allow for consistently tracking the same motion channels over time, enabling reconstruction of continuous trajectories.

Dynamic serialization breaks this assumption by shuffling motion fields at the packet level, so that the same coordinate no longer appears at a fixed location across packets. Concretely, client and server establish a session secret during a secure control-plane handshake (e.g., TLS used for matchmaking), derive a packet-specific permutation π_t from the secret and packet index t, and serialize motion values according to π_t. The receiver then inverts π_t to recover semantics.

Unlike cryptographic encryption, this permutation relies on lightweight pseudo-random number generator (PRNG) operations, incurring $O(1)$ computational complexity per packet with negligible latency overhead, making it suitable for resource-constrained VR headsets. This disrupts the fixed byte-offset mapping and the cross-packet temporal alignment PACMO relies on, while preserving application behavior.

Limitations and Effectiveness. Dynamic serialization serves as best-effort obfuscation rather than a cryptographic guarantee. In practice, developers may opt for a *Restricted* permutation (shuffling only within logical groups like head or hand structs) to preserve code modularity, whereas a *Full* permutation requires flattening all motion fields into a single contiguous representation. To quantify the impact, we evaluated both strategies (Table 6):

- *Restricted:* Identification accuracy decreases to 47.8%. This suggests that while exact mappings are broken, the localized grouping still leaks residual statistical information.
- *Full:* Permuting globally across all motion fields reduces accuracy further to 21.7%. This confirms that maximizing the permutation space effectively disrupts the temporal structure required for reconstruction, offering a viable stopgap when full DTLS is infeasible.

8 Related Work

Our work connects (i) behavioral inference from VR motion, (ii) VR input and keystroke inference, and (iii) network-side analysis of VR traffic. Our key distinction is the observation layer: we show that a passive on-path adversary can reconstruct motion-like trajectories from plaintext UDP payloads without message-format knowledge or privileged device/application access.

VR Motion as Behavioral Biometrics. Head and hand motion traces are established behavioral biometrics. Nair *et al*. demonstrate identification at scale from head and hand motion [31]. Follow-up studies examine robustness across applications and sensor subsets and show that motion can reveal sensitive attributes beyond identity [18,24,33,56]. Other work studies avatar de-anonymization and movement-signature correlation, along with countermeasures [12,26]. These works typically assume access to raw sensor streams or in-app/OS telemetry; in contrast, we reconstruct motion from passive network observation and still achieve strong identification.

VR Input and Keystroke Inference Attacks. Prior work shows that VR micro-motions can leak sensitive inputs, including keylogging from head motion and from motion/pose telemetry, as well as broader AR/VR side channels [45,57,60]. Other attacks assume stronger vantage points in multi-user settings, such as avatar views or shared environments [47,54], and recent work proposes defenses for VR typing [58]. Additional modalities include air-tapping, infrared channels, motion-sensor side channels leaking non-textual secrets, and wireless physical-layer sensing [3,8,27,34,55]. VR-Spy and MuKI-Fi are especially relevant because they infer virtual keystrokes from motion-induced wireless channel changes: VR-Spy targets VR headset interactions using CSI-style observations, while MuKI-Fi extends keystroke inference to multi-person BFI-enabled Wi-Fi sensing [3,55]. These works share our high-level concern that subtle body and controller motion can leak sensitive input, but they observe a different side channel. They rely on wireless physical-layer measurements, whereas PACMO assumes a passive on-path network observer and extracts motion from application-layer plaintext UDP payloads used for VR synchronization. We show that relevant micro-motion structure can survive application processing and network transmission: even without co-presence, a passive eavesdropper can reconstruct motion from plaintext UDP payloads and infer typing. Prior network-side keylogging is not a direct baseline because it assumes co-presence and manual message encoding understanding [47] or relies on avatar observation rather than packet payloads [54].

VR Network Traffic Analysis. Several systems efforts infer VR/metaverse behavior from traffic statistics such as sizes,

timing, and flow-level features. OVRSeen audits traffic and privacy-policy compliance in Oculus VR [49]. MetaVRadar and related studies characterize traffic and infer coarse session or activity states [10, 25], and recent work shows that even encrypted traffic can leak application usage and coarse activity classes via ML [29]. Baldoni *et al*. present the closest network-side identification work, showing that traffic features can identify users in commercial VR [4]. Our goal and granularity differ: rather than relying on coarse traffic statistics, we decode motion-bearing fields from plaintext UDP payloads under unknown encodings to recover motion-like time series that enable stronger identification and keystroke inference.

Defenses and Privacy Controls for VR Sensing Streams. Recent work develops methodologies and tools for assessing VR security and privacy risks, including ecosystem studies [6, 16, 19, 21, 36, 43] and automated auditing frameworks [17, 44, 59]. We add a network-layer threat that applies even when applications are treated as black boxes and the adversary only passively observes traffic. By combining ecosystem measurement, automated reconstruction, and downstream attacks, we show that plaintext motion updates create a practical, high-impact leakage channel, motivating secure-by-default networking library support for motion-bearing traffic.

9 Concluding Remarks

We show that passive observers can reconstruct VR motion from plaintext UDP. Our evaluation demonstrates high-fidelity recovery, achieving up to 96.2% user identification and 69.4% typing inference accuracy, far exceeding baselines using coarse traffic statistics. The results confirm that motion updates are sensitive telemetry requiring protection, as they expose users to scalable identity and input leakage. Finally, we discuss the boundaries of our threat model and provide actionable recommendations for securing the VR ecosystem.

9.1 Threat Model Realism and Boundaries

Realism and Scalability. Our threat model is strictly passive: observing packet metadata and plaintext payloads without malware installation or virtual co-presence. The only active requirement is offline profiling on the attacker's own device to learn the input-to-payload mapping. Crucially, this mapping targets the *application's* encoding structure, not the specific user; thus, it is a one-time cost reusable across all victims using the same version of the application. In practice, this requirement is even weaker. As long as the encoding structure remains unchanged, the attack continues to work even across application updates. For example, we confirm that despite an application update in *Eleven Table Tennis*, the encoding structure remains unchanged, which allows our attack to remain effective without re-profiling.

Boundaries and Coverage. The boundaries of our threat are clear. If motion traffic is protected end-to-end (e.g., using DTLS), Phases I/II of PACMO no longer apply, and the adversary is limited to metadata-only traffic analysis. Regarding coverage, our profiling phase is most efficient when applications support simulation tooling (e.g., NVIDIA VCR). While anti-cheat checks in some applications may hinder automated injection, this represents a barrier to attackers' *convenience* (automation), not a fundamental mitigation of the root vulnerability (i.e., deterministic serialization of motion data). Even without simulation tools, an adversary can manually isolate individual inputs (e.g., moving only one controller at a time) and observe payload changes accordingly. In our experiments, such naive probing was sufficient to identify motion-related offsets, indicating that tooling mainly impacts efficiency rather than feasibility.

Our evaluation also does not exhaustively model all real-world background traffic conditions. We tested flow isolation with background web browsing and video streaming and observed that unrelated flows did not match both the application endpoint/port filter and the profiled high-rate, small-payload UDP motion signature. Nevertheless, more complex environments could contain other high-rate UDP applications, local relay behavior, VPNs, or NAT aggregation that complicate flow attribution. In such settings, PACMO still targets the application's motion-uplink flow, but the attacker may require additional flow-labeling evidence such as DNS/SNI context, server endpoints, or longer profiling traces.

9.2 Path to Secure the VR Ecosystem

Section 7 reveals that plaintext persists due to integration friction, performance concerns, and low perceived risks. Thus, the academic recommendation to "just use DTLS" is correct but incomplete. The VR ecosystem requires secure-by-default application-layer networking frameworks with performance-tuned presets to lower engineering costs. When full DTLS is infeasible or inconvenient, lightweight stopgaps like dynamic serialization can reduce exposure, although they cannot replace cryptographic confidentiality.

As multi-user VR becomes ubiquitous, the gap between performance and privacy must close. Future work should characterize residual leakage under encrypted transports (e.g., DTLS). Treating motion traffic as sensitive by default is the only credible path to prevent scalable, passive reconstruction attacks threatening user privacy.

Acknowledgements

The authors acknowledge funding support from the Virginia Tech Center for Human-Computer Interaction User Study Fund and the National Science Foundation (Grant CNS-2350116).

Ethical Considerations

This work investigates privacy risks arising from plaintext motion telemetry in multi-user networked VR applications by demonstrating that head and hand motions can be reconstructed from network traffic and subsequently used for downstream inference.

Stakeholders. We identify the following stakeholders:

- *End Users of VR Applications:* They may be exposed to privacy risks, including behavioral identification and sensitive input leakage, if motion data is transmitted without adequate protection.
- *VR Application Developers and Platform Providers:* Our findings may impose engineering burden or reputational risk, but also provide actionable evidence to improve security practices.
- *Research Participants:* Participants in our user study may face privacy risks if their motion or interaction data were mishandled.
- *Researchers:* The research team may face professional or legal risks associated with responsible disclosure and the handling of sensitive vulnerability information.
- *The Broader VR Ecosystem and Society:* The results inform systematic risks in emerging immersive platforms, which are increasingly used for social interaction, gaming, work, education, etc.
- *Potential Adversaries:* While not a beneficiary stakeholder, adversaries could exploit the techniques presented.
- *Network Operators:* Operators of local networks, enterprise gateways, or access points may be in a position to observe plaintext VR traffic and therefore have a role in deploying or encouraging secure transport practices.

Human Subjects. This study involves human-subject data in the form of VR motion and network traces. The study was approved by the Virginia Tech Institutional Review Board (IRB). All participants provided informed consent prior to participation. Participation was voluntary, and participants could withdraw at any time. Participants received modest compensation of $10 for their time, which was not contingent on task performance. IRB approval does not replace broader ethical analysis; we therefore also considered ecosystem-level risks and downstream impacts.

Data Collection and Protection. All experiments were conducted using commercial applications with consenting participants in controlled laboratory settings. We did not collect data from unsuspecting users or individuals without their knowledge and consent. Network measurements were limited to study sessions; incidental background traffic was discarded. Typing and interaction tasks used predefined, non-sensitive prompts. We anonymized the collected data and removed direct identifiers prior to analysis. Raw packet traces are stored on access-controlled systems and are not publicly released.

Responsible Disclosure. Because our work identifies privacy risks in deployed VR applications, we initiated responsible disclosure before camera-ready submission. We contacted the developers of 48 out of the 75 applications studied for which we were able to identify an appropriate contact channel (e.g., email or developer portal), as well as Steam, a relevant application distribution platform, during May 16-18, 2026. On May 30, 2026, we sent follow-up messages to the contacted parties communicating that the paper is expected to be publicly released on August 12, 2026. In the follow-up, we explicitly asked whether any party needed additional time to assess the issue, implement mitigations, or coordinate publication details before public release. Our disclosure messages described the high-level issue: plaintext motion synchronization traffic in multi-user VR sessions may allow a passive on-path observer to reconstruct sensitive head and hand motion telemetry and use it for downstream behavioral inference. We also summarized the threat model, potential privacy impact, and high-level mitigation directions.

As of June 11, 2026, we have received a total of seven responses. These responses included acknowledgments of receipt, requests for additional technical details, internal review forwarding, and two providers indicating that mitigations are under development. Separately, two teams, Hyper Dash and Synth Riders, indicated that the developer is investigating mitigations, e.g., enabling transport encryption planned before public release. No provider has requested an embargo beyond August 12, 2026.

Dual-Use Considerations and Mitigations. Our work has dual-use potential. Techniques used to analyze plaintext VR motion traffic and to reconstruct motion telemetry could be exploited by adversaries for behavioral monitoring, user profiling, user identification, or virtual typing inference. To reduce these risks, we avoid releasing raw packet payload traces or identifiable motion traces from participants, limit public artifacts to materials needed to evaluate the research claims, and frame our technical discussion around defensive implications and protocol-level protections.

While our techniques have dual-use potential, our goal is to expose a previously underexplored attack surface and motivate secure-by-design networking practices for VR systems. We reduce potential harms through informed consent, limited data collection, responsible disclosure, restricted artifact release, and no data collection from unsuspecting users or individuals without their knowledge and consent. We decided to publish this work because we believe that early disclosure of these risks, together with concrete mitigations, will help strengthen the privacy and security of future VR ecosystems.

Open Science

To facilitate reproducibility and future research, we will make the following artifacts publicly available via our project repository:

- PACMO *Source Code*: We provide the complete Python implementation of our attack pipeline, including the "Encoding-Agnostic Mapping" (Phase I), "Online Motion Reconstruction" (Phase II), and "Behavioral Inference" (Phase III) modules.
- *Profiling Scripts*: We provide the automation scripts used to control NVIDIA VCR for the waveform injection process, allowing for the profiling of new or updated VR applications.

The artifact is available at: https://doi.org/10.5281/zenodo.20343460

References

[1] Bigscreen, Inc. Bigscreen Beta on Steam. https://store.steampowered.com/app/457550/Bigscreen_Beta/, 2016. Accessed: 2026-01-21.

[2] Devon Adams, Alseny Bah, Catherine Barwulor, Nureli Musaby, Kadeem Pitkin, and Elissa M Redmiles. Ethics emerging: the story of privacy and security perceptions in virtual reality. In *Fourteenth symposium on usable privacy and security (SOUPS 2018)*, pages 427–442, 2018.

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

[4] Sara Baldoni, Salim Benhamadi, Federico Chiariotti, Michele Zorzi, and Federica Battisti. Movement-and Traffic-based User Identification in Commercial Virtual Reality Applications: Threats and Opportunities. In *2025 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 72–81. IEEE, 2025.

[5] Beat Games. Beat Saber on Steam. https://store.steampowered.com/app/620980/Beat_Saber/, 2019. Accessed: 2026-01-21.

[6] Jiaxun Cao, Anupam Das, Pardis Emami-Naeini, et al. Understanding parents' perceptions and practices toward children's security and privacy in virtual reality. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 1554–1572. IEEE, 2024.

[7] Leonor Adriana Cárdenas-Robledo, Óscar Hernández-Uribe, Carolina Reta, and Jose Antonio Cantoral-Ceballos. Extended reality applications in industry 4.0.–a systematic literature review. *Telematics and Informatics*, 73:101863, 2022.

[8] Derin Cayir, Reham Mohamed Aburas, Riccardo Lazzeretti, Marco Angelini, Abbas Acar, Mauro Conti, Z Berkay Celik, A Selcuk Uluagac, et al. Speak up, i'm listening: Extracting speech from zero-permission vr sensors. In *NDSS*, 2025.

[9] Kuan-Ta Chen, Chun-Ying Huang, Polly Huang, and Chin-Laung Lei. An empirical evaluation of TCP performance in online games. In *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, pages 5–es, 2006.

[10] Ruizhi Cheng, Nan Wu, Matteo Varvello, Songqing Chen, and Bo Han. Are we ready for metaverse?: a measurement study of social virtual reality platforms. In *Proceedings of the 22nd ACM Internet Measurement Conference (IMC)*, pages 504–518, 2022.

[11] Epic Games. The most powerful real-time 3d creation tool - unreal engine. https://www.unrealengine.com. Accessed: 2026-02-06.

[12] Brandon Falk, Yan Meng, Yuxia Zhan, and Haojin Zhu. Poster: Reavatar: Virtual reality de-anonymization attack through correlating movement signatures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2405–2407, 2021.

[13] Wu-chang Feng, Francis Chang, Wu-chi Feng, and Jonathan Walpole. A traffic characterization of popular on-line games. *IEEE/ACM Transactions On Networking*, 13(3):488–500, 2005.

[14] For Fun Labs. Eleven Table Tennis on Steam. https://store.steampowered.com/app/488310/Eleven_Table_Tennis/, 2016. Accessed: 2026-01-21.

[15] GlobeNewswire. Virtual Reality Market Competitive Analysis Report 2025: Key Players, Recent Developments, Strategy, Sustainability, Product Launch, Key Persons, and Revenue Forecast to 2033. https://www.globenewswire.com/news-release/2025/11/10/3184695/28124/en/Virtual-Reality-Market-Competitive-Analysis-Report-2025-Key-Players-Recent-Developments-Strategy-Sustainability-Product-Launch-Key-Persons-and-Revenue-Forecast-to-2033.html, November 2025. Press release.

[16] Jan Gugenheimer, Wen-Jie Tseng, Abraham Hani Mhaidli, Jan Ole Rixen, Mark McGill, Michael Nebeling, Mohamed Khamis, Florian Schaub, and Sanchari Das. Novel challenges of safety, security and privacy in extended reality. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–5, 2022.

[17] Hanyang Guo, Hong-Ning Dai, Xiapu Luo, Zibin Zheng, Gengyang Xu, and Fengliang He. An Empirical Study on Oculus Virtual Reality Applications: Security and Privacy Perspectives. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE)*, 2024.

[18] Ismat Jarin, Yu Duan, Rahmadi Trimananda, Hao Cui, Salma Elmalaki, and Athina Markopoulou. Behavr: User identification based on vr sensor data. *arXiv preprint arXiv:2308.07304*, 2023.

[19] John M Jones, Reyhan Duezguen, Peter Mayer, Melanie Volkamer, and Sanchari Das. A literature review on virtual reality authentication. In *International Symposium on Human Aspects of Information Security and Assurance*, pages 189–198. Springer, 2021.

[20] Sophie Jörg, Yuting Ye, Franziska Mueller, Michael Neff, and Victor Zordan. Virtual hands in vr: Motion capture, synthesis, and perception. In *SIGGRAPH Asia 2020 Courses*, pages 1–32. 2020.

[21] Emiram Kablo, Melina Kleber, and Patricia Arias Cabarcos. {PrivaCI} in {VR}: Exploring perceptions and acceptability of data sharing in virtual reality through contextual integrity. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 1531–1548, 2025.

[22] Alex Kantchelian, J Doug Tygar, and Anthony Joseph. Evasion and hardening of tree ensemble classifiers. In *International conference on machine learning*, pages 2387–2396. PMLR, 2016.

[23] Eamonn J. Keogh and Michael J. Pazzani. Derivative Dynamic Time Warping. In *SDM*, 2001.

[24] Jonathan Liebers, Mark Abdelaziz, Lukas Mecke, Alia Saad, Jonas Auda, Uwe Gruenefeld, Florian Alt, and Stefan Schneegass. Understanding user identification in virtual reality through behavioral biometrics and the effect of body normalization. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–11, 2021.

[25] Minzhao Lyu, Rahul Dev Tripathi, and Vijay Sivaraman. Metavradar: Measuring metaverse virtual reality network activity. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 7(3):1–29, 2023.

[26] Yan Meng, Yuxia Zhan, Jiachun Li, Suguo Du, Haojin Zhu, and Xuemin Shen. De-anonymizing avatars in virtual reality: Attacks and countermeasures. *IEEE Transactions on Mobile Computing*, 2024.

[27] Ülkü Meteriz-Yıldıran, Necip Fazıl Yıldıran, Amro Awad, and David Mohaisen. A keylogging inference attack on air-tapping keyboards in virtual environments. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 765–774. IEEE, 2022.

[28] Mirror Networking. Mirror Networking – Open Source Networking for Unity. https://mirror-networking.com/. Accessed: 2026-02-06.

[29] Sheikh Samit Muhaimin and Spyridon Mastorakis. I Know What You Did Last Summer: Identifying VR User Activity Through VR Network Traffic. *arXiv preprint arXiv:2501.15313*, 2025.

[30] Nacho Caballero. Word-by-frequency. https://github.com/nachocab/words-by-frequency, 2024. Accessed: 2026-01-21.

[31] Vivek Nair, Wenbo Guo, Justus Mattern, Rui Wang, James F O'Brien, Louis Rosenberg, and Dawn Song. Unique identification of 50,000+ virtual reality users from head & hand motion data. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 895–910, 2023.

[32] Vivek Nair, Wenbo Guo, Rui Wang, James F. O'Brien, Louis Rosenberg, and Dawn Song. Berkeley Open Extended Reality Recordings 2023 (BOXRR-23): 4.7 Million Motion Capture Recordings from 105,000 XR Users. *IEEE Transactions on Visualization and Computer Graphics*, 30(5):2239–2246, May 2024.

[33] Vivek Nair, Christian Rack, Wenbo Guo, Rui Wang, Shuixian Li, Brandon Huang, Atticus Cull, James F O'Brien, Marc Latoschik, Louis Rosenberg, et al. Inferring private personal attributes of virtual reality users from head and hand motion data. *arXiv preprint arXiv:2305.19198*, 2023.

[34] Tao Ni, Yuefeng Du, Qingchuan Zhao, and Cong Wang. Non-intrusive and unconstrained keystroke inference in vr platforms via infrared side channel. *arXiv preprint arXiv:2412.14815*, 2024.

[35] Naheem Noah and Sanchari Das. Exploring evolution of augmented and virtual reality education space in 2020 through systematic literature review. *Computer Animation and Virtual Worlds*, 32(3-4):e2020, 2021.

[36] Naheem Noah, Sommer Shearer, and Sanchari Das. Security and privacy evaluation of popular augmented and virtual reality technologies. In *Proceedings of the 2022*

IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence, and Neural Engineering (IEEE MetroXRAINE 2022), 2022.

[37] Aline Normoyle, Gina Guerrero, and Sophie Jörg. Player perception of delays and jitter in character responsiveness. In *Proceedings of the ACM symposium on applied perception*, pages 117–124, 2014.

[38] NVIDIA. NVIDIA VCR - VR Capture and Replay. https://github.com/NVIDIA/vr-capture-replay, 2024. Accessed: 2026-01-21.

[39] OnSkull Games. Tales of Escape on Steam. https://store.steampowered.com/app/587860/Tales_of_Escape/, 2017. Accessed: 2026-02-04.

[40] OptiTrackR. Data Streaming | EXTERNAL OptiTrack Documentation. https://docs.optitrack.com/motive/data-streaming. Accessed: 2026-01-21.

[41] Photon. Pun 2 - Binary Protocol | Photon Engine. https://doc.photonengine.com/pun/current/reference/binary-protocol. Accessed: 2026-01-21.

[42] Photon Engine. Multiplayer Game Development Made Easy | Photon Engine. https://www.photonengine.com. Accessed: 2026-02-06.

[43] Abhinaya SB, Abhishri Agrawal, Yaxing Yao, Yixin Zou, and Anupam Das. " What are they gonna do with my data?": Privacy Expectations, Concerns, and Behaviors in Virtual Reality. *Proceedings on Privacy Enhancing Technologies*, 2025(1):58–77, 2025.

[44] Muhammad Shoaib, Alex Suh, and Wajih Ul Hassan. Principled and Automated Approach for Investigating AR/VR Attacks. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 4325–4344, 2025.

[45] Carter Slocum, Yicheng Zhang, Nael Abu-Ghazaleh, and Jiasi Chen. Going through the motions: AR/VR keylogging from user head motions. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 159–174, Anaheim, CA, August 2023. USENIX Association.

[46] Steamworks.NET. Steamworks.Net. https://steamworks.github.io/. Accessed: 2026-02-06.

[47] Zihao Su, Kunlin Cai, Reuben Beeler, Lukas Dresel, Allan Garcia, Ilya Grishchenko, Yuan Tian, Christopher Kruegel, and Giovanni Vigna. Remote Keylogging Attacks in Multi-user VR Applications. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2743–2760, 2024.

[48] Triangle Factory. Hyper Dash on Steam. https://store.steampowered.com/app/1386890/Hyper_Dash/, 2021. Accessed: 2026-02-04.

[49] Rahmadi Trimananda, Hieu Le, Hao Cui, Janice Tran Ho, Anastasia Shuba, and Athina Markopoulou. OVRseen: Auditing network traffic and privacy policies in oculus VR. In *31st USENIX security symposium (USENIX security 22)*, pages 3789–3806, 2022.

[50] Unity. Networking & Netcode Software Solution. https://unity.com/features/netcode. Accessed: 2026-02-06.

[51] Unity. State Synchronization Details (Legacy). https://docs.unity3d.com/510/Documentation/Manual/net-StateSynchronization.html. Accessed: 2026-01-21.

[52] Unity Technologies. Unity real-time development platform | 3d, 2d, vr & ar engine. https://unity.com/. Accessed: 2026-02-06.

[53] WanadevStudio. Ragnarock on Steam. https://store.steampowered.com/app/1345820/Ragnarock/, 2021. Accessed: 2026-01-21.

[54] Hanqiu Wang, Zihao Zhan, Haoqi Shan, Siqi Dai, Maximilian Panoff, and Shuo Wang. Gazeploit: Remote keystroke inference attack by gaze estimation from avatar views in vr/mr devices. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1731–1745, 2024.

[55] Hongbo Wang, Jingyang Hu, Tianyue Zheng, Jingzhi Hu, Zhe Chen, Hongbo Jiang, Yuanjin Zheng, and Jun Luo. Muki-fi: Multi-person keystroke inference with bfi-enabled wi-fi sensing. *IEEE Transactions on Mobile Computing*, 23(10):9835–9850, 2024.

[56] Qidi J Wang, Alec G Moore, Nayan N Chawla, and Ryan P McMahan. Cross-Domain Gender Identification Using VR Tracking Data. In *2024 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 180–189. IEEE, 2024.

[57] Yi Wu, Cong Shi, Tianfang Zhang, Payton Walker, Jian Liu, Nitesh Saxena, and Yingying 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)*, pages 3382–3398. IEEE, 2023.

[58] Zhuolin Yang, Zain Sarwar, Iris Hwang, Ronik Bhaskar, Ben Y Zhao, and Haitao Zheng. Can virtual reality protect users from keystroke inference attacks? In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 2725–2742, 2024.

[59] Yuxia Zhan, Yan Meng, Lu Zhou, Yichang Xiong, Xiaokuan Zhang, Lichuan Ma, Guoxing Chen, Qingqi Pei,

and Haojin Zhu. VPVet: Vetting Privacy Policies of Virtual Reality Apps. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2024.

[60] Yicheng Zhang, Carter Slocum, Jiasi Chen, and Nael Abu-Ghazaleh. It's all in your head (set): Side-channel attacks on AR/VR systems. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3979–3996, 2023.