Cyrinx: A Measured 36.6/27.3 kbps Bidirectional Acoustic Data Link Between a Laptop and a Phone

An empirical account of building, breaking, and verifying a wideband OFDM acoustic modem on commodity hardware

David E. Weekly*

June 10, 2026

Abstract

We report a software acoustic data link between a MacBook Pro (M4) speaker/microphone pair and a Google Pixel 7a resting on the laptop's palm rest, achieving measured, byte-verified over-the-air goodput of **36,571 bps** Mac→Pixel (demodulated *on the phone* by a Kotlin port of the receiver) and **27,307 bps** Pixel→Mac, against a target of 20 kbps in each direction. Goodput is counted conservatively: only payload bytes that are both CRC32-valid and, under *ordered* verification, byte-identical to the transmitted `DetRng` payload at their correct ordered position within the attributed frame (not mere set membership), divided by the active message span (first preamble chirp to last data sample, which excludes the fixed trailing silence) and charged for all preambles, channel-estimation symbols, pilots, FEC redundancy, CRCs, and inter-frame gaps; the gross figure including trailing silence is marginally lower. The modem is a 48 kHz OFDM design (FFT 2048, cyclic prefix 768) with comb pilots, per-symbol timing/phase tracking, uniform 16-QAM, and a punctured rate-3/4 K=7 convolutional code. Reaching the target required finding and fixing four physical-layer defects that are invisible to simulation: inter-symbol interference from a 10–22 ms delay spread, decorrelated dual-speaker transmission, a stream-end fade in the macOS output chain, and Viterbi poisoning by overconfident log-likelihood ratios from corrupted symbols. We document the discriminating experiments used to isolate each defect—including the negative results that exonerated "obvious" suspects—and contrast the delivered system with both the original design document and a predecessor stack whose theoretical 22 kbps claim corresponded to a measured 0.27 kbps. We argue that for acoustic modems on consumer hardware, the dominant engineering cost is the gap between the channel model and the channel. None of the modulation or coding machinery here is novel — OFDM, cyclic prefixes, comb pilots, QAM, convolutional/Viterbi coding, CRC block verification, and acoustic data-over-sound are all long established; the contribution is the *measured* end-to-end system on commodity heterogeneous hardware, the discriminating-experiment diagnostic methodology, the physical/platform defect catalog, and on-device verification, not new modulation theory. The modem has since been ported from the research harness into a portable-C reference library (transmit and receive), validated bit-exact against committed golden vectors with an Apple vDSP backend gated by those same vectors; the library's own codec decodes over the air byte-verified and at the palm-rest geometry reproduces the headline natively (16-QAM rate 3/4, 39.3 kbps, exceeding the harness figure), and it ships with an SNR/delay-spread MCS sounder and a channel-metric-driven user repositioning guide. We frame the result accordingly: this is a well-instrumented empirical case study of how much commodity hardware delivers after real-channel debugging—validated on two device

*Primatech Paper Co LLC, `david@weekly.org`. The modem, diagnostics, and this whitepaper were developed with substantial AI-agent assistance (Claude Code); all reported results are from over-the-air measurements on physical hardware recorded in the project's lab notebook.

pairs (a Pixel 7a and an iPhone 17 Pro Max, each against the same MacBook Pro) in a single contact-range geometry, with same-hardware baselines against deployed tools—and a carefully measured contact-range prototype, not a broadly general or new-modulation acoustic-modem result.

1 Introduction and Motivation

Consumer laptops and phones ship with speakers and microphones of considerably higher fidelity than the "data-over-sound" literature typically assumes — fidelity the acoustic-*sensing* literature already exploits aggressively [20] — yet deployed acoustic-communication systems cluster at very low rates: roughly 10^2 bps for robust spread-spectrum schemes and roughly 10^3 bps for multi-tone FSK [3, 7]. Acoustic links remain attractive for proximity-bound exchange—pairing tokens, small payloads, air-gapped transfer—because acoustic signals attenuate far faster and are more locality-bound than RF—especially the audible wideband link used here, which does not mean sound cannot penetrate walls, only that it does so far more weakly—require no radio pairing ritual, and use hardware already present on billions of devices [1, 10].

The Cyrinx project asked a narrower, harder question: *how much measured goodput can two specific commodity devices actually exchange acoustically?* The target was at least 20 kbps of verified goodput in each direction between a MacBook Pro (M4) and a Pixel 7a lying face-up on the laptop's palm rest (on a soft cloth, both devices at maximum volume, in a quiet room). The emphasis on *measured* is deliberate. An earlier in-repo protocol stack claimed "22 kbps" on the basis of a subcarrier-count capacity formula; its measured goodput was under 0.3 kbps (24-byte packets every ∼700 ms)—a factor of roughly 80 between claim and reality, and a factor of ∼135 between what that stack delivered and what the channel supports (Section 6). This paper treats that episode as data: every number reported here is from an over-the-air capture on physical hardware, and we state explicitly when a figure is an estimate rather than a measurement. We are deliberate about scope: what follows is an empirical case study of two specific device pairs in one geometry, not a general acoustic-modem result.

Contributions:

1. A characterization of the near-field laptop↔phone acoustic channel (Section 3): per-band SNR in both directions, sample-clock skew, and delay spread.

2. A wideband OFDM modem design that achieves 36.6/27.3 kbps verified goodput, about 4.5–9× the original design document's own "optimal conditions" projection (Section 4).

3. A catalog of four physical-layer defects, each fatal to the link and each invisible to simulation, with the discriminating experiments (including negative results) that isolated them (Section 5).

4. A verification methodology in which the receiving phone regenerates the expected payload locally from a portable deterministic PRNG and proves goodput on-device, so no captured samples need leave the phone (Section 4.2).

5. A portable-C reference implementation of the full transmit/receive codec, pinned by a committed golden-vector contract (with an Apple vDSP backend gated by the same vectors), that decodes over the air byte-verified through the library itself, plus the adaptive-MCS sounder and channel-metric repositioning guidance built on top (Sections 4.3, 6.5, 8.2).

2 Related Work

The original Cyrinx design document [33] surveyed the deployed data-over-sound landscape; we summarize that survey here and credit the systems directly.

libquiet / Quiet Modem Project. Quiet [1, 2] brought soft-decision decoding (via the `libcorrect` library) to open-source acoustic networking, with GMSK and PSK modes and Reed–Solomon/convolutional error correction. Its main limitation is rigidity: the user selects a fixed profile (e.g. `ultrasonic-experimental` or `audible-7k`) at initialization, and there is no feedback loop to adapt constellation density to the channel. Cyrinx inherits Quiet's conviction that soft decisions matter — indeed, our highest-leverage single fix was a soft-decision weighting refinement (Section 5).

minimodem. A general-purpose FSK/RTTY software modem [5]. FSK is robust to phase corruption from echoes but spectrally inefficient, and minimodem deliberately operates as a "dumb pipe" with no integrated FEC [6], delegating integrity to the application.

ggwave. A widely used open-source data-over-sound library using multi-tone FSK (e.g. dividing the spectrum into 96 discrete frequencies and transmitting 6 tones at once to encode 3 bytes) [7]. Energy-detection demodulation makes it resilient to multipath, but it does not exploit high-SNR channels: where QAM can carry 4–6 bits per Hz, MT-FSK carries less than one. Rather than rely on across-paper rate comparisons, we ran ggwave and minimodem over our *own* hardware and geometry as same-channel baselines (Section 6.4, Table 6).

Google Nearby Messages. The audio modality of Nearby [4] (since deprecated; ultrasonic advertising was removed in 2021) was built on the consumer-hardware ultrasonic scheme of Getreuer et al. [3], which uses direct-sequence spread spectrum with a 127-chip code and reports approximately 94.5 bps, trading nearly all throughput for detection probability. Sufficient for a token, far from bulk transfer.

Chirp.io and LISNR. Commercial closed-source systems [9, 10] that emphasized the *wake-up problem*: a cheap, low-power preamble detector that gates the expensive DSP. LISNR's "Kilo Audio Bit" product claims throughputs around 1 kbps. Cyrinx adopts the preamble-gated architecture (a chirp matched filter gates the OFDM decoder) while being fully open about its error-correction internals.

BatNet. The most directly comparable prior art for a phone-to-phone ultrasonic link: a smartphone system that carries data between built-in speakers and microphones using 8-PSK in the 20–24 kHz inaudible band, reporting over 600 bps at up to 6 m [8]. BatNet is the established demonstration that consumer phone transducers can sustain a coherent (phase-modulated) ultrasonic link at all; our inaudible-band results (Section 8) are consistent with it on one direction and illuminate why the other direction fails on the specific Pixel 7a micro-speaker.

Synchronization literature. The predecessor Cyrinx stack used Zadoff–Chu CAZAC sequences for preamble synchronization and CFO estimation, following standard practice [11, 12], and its robust control-header modulation ("D-CSS") was inspired by LoRa-style chirp spread spectrum [13]. The bulk PHY described in this paper replaced the ZC preamble with a linear chirp plus known OFDM sync symbols, and replaced one-shot CFO estimation with continuous per-symbol pilot tracking—a change forced by measurement (Section 5). Pilot-based joint tracking of sampling-clock and carrier offsets is itself standard OFDM receiver practice [14, 15], and clock drift between independently clocked audio devices is a long-known problem in acoustic echo cancellation [16]; we claim no novelty here either — the finding is only that on this channel continuous tracking is mandatory rather than optional.

Platform audio stacks. Prior practitioners documented the hostility of consumer audio stacks to modem waveforms: voice-processing I/O units inject echo cancellation and noise suppression that attenuate steady carriers [28, 29], and macOS applies its own processing to chirps [30]. Our

Android findings (Section 7) are the same lesson on a different OS: `AudioSource.UNPROCESSED` is load-bearing, and the capture path can be silently zeroed by UI state.

Academic work on aerial acoustic communication with consumer hardware [3] has demonstrated kbps-class links in the near-ultrasonic band; our contribution is less a novel modulation than a carefully verified data point—tens of kbps, bidirectional, decoded on the receiving phone—and a reproducible account of what it took to get there.

On originality. We want to be explicit that the core communications techniques used here are not original to this work. OFDM, cyclic prefixes, comb pilots, QAM, punctured convolutional codes with Viterbi decoding, CRC block verification, acoustic speaker-to-microphone links, and ultrasonic data-over-sound are all long established, and every system surveyed above predates and informs ours. We invented no new modulation, code, or synchronization primitive. The contribution of this paper is instead: (i) a *measured*, byte-verified end-to-end link on commodity heterogeneous hardware (a MacBook Pro and a Pixel 7a, decoded on the phone); (ii) a discriminating-experiment diagnostic methodology, reported with its negative results; (iii) a catalog of physical-layer and platform defects that are invisible to simulation; and (iv) on-device verification via a portable deterministic PRBS. The novelty is empirical and methodological, not theoretical.

3 Channel Characterization

3.1 Methodology

All characterization was over-the-air at the final physical geometry (Pixel 7a face-up on the MacBook Pro palm rest on a soft cloth). Three probe types were used, none requiring synchronization between the devices:

- **Probe-on vs. probe-off PSD.** A random-phase multitone spanning 0.3–23.5 kHz at digital amplitude 0.6 was played while the far device recorded; Welch periodograms (NFFT 1024, 46.875 Hz bins) of probe-on and 6 s of ambient floor yield per-band SNR without any time alignment.
- **Exponential sine sweep (ESS)** for impulse response and delay spread.
- **Long pure tone** (10 kHz, 8 s) for sample-clock offset, via a quadratically interpolated FFT peak.

3.2 Measurements

Table 1 gives the measured SNR by band in both directions (Mac→Android, denoted M→A, uses the Pixel's bottom microphone, channel 0).

Key findings, several of which contradicted prior assumptions:

- **The Mac→Pixel channel is essentially flat to 23 kHz** in this near-field geometry. A roll-off assumption carried in the original design document ("−32.5 dB at 16 kHz" [33]) did not reproduce under measurement; the design decision it had motivated (a narrow 10–14 kHz "flat shelf" band) was therefore wrong, and the usable band is far wider.
- **The Pixel→Mac direction dies above ∼17 kHz**: 10.0 dB SNR at 18–21 kHz falling to 4.6 dB at 21–24 kHz (Pixel speaker and/or Mac microphone roll-off), visible as the uplink cliff in Figure 1. This asymmetry has direct consequences for any inaudible-band variant (Section 8).

Table 1: Measured over-the-air SNR by band (Welch PSD, probe-on vs. ambient floor), 2026-06-09. M→A = Mac speaker to Pixel 7a bottom mic; A→M = Pixel speaker to Mac mic.

Band (kHz)	M→A SNR (dB)	A→M SNR (dB)
0.3–2	26.4	28.1
2–6	41.2	44.3
6–10	37.1	47.3
10–14	36.0	45.5
14–18	38.1	33.9
18–21	36.1	10.0
21–24	40.7	4.6

- **Sample-clock offset between the two devices is -24.6 ppm**, which at the final OFDM geometry corresponds to about 0.037 samples of drift per symbol — small per symbol, but ruinous over a frame without continuous tracking.
- **Delay spread (ESS, -30 dB threshold) is $\sim$21.7 ms M→A and $\sim$10.8 ms A→M.** Most energy arrives in the first few ms, but the reverberant tail is long; this is roughly an order of magnitude larger than the 2 ms the original design document budgeted for.
- **The Pixel's bottom microphone beats its top microphone by 6–20 dB below 10 kHz** at this geometry.

From Table 1, Shannon capacity with an 8-bit-per-symbol cap is approximately 184 kbps M→A and 155 kbps A→M (an estimate, not a measurement). The achieved link uses roughly 20–24% of that, so substantial headroom remains.

4 System Design

4.1 Modem architecture

The link is a half-duplex, open-loop bulk-transfer PHY at 48 kHz PCM16. The OFDM geometry is NFFT 2048 (23.4 Hz bins) with a 768-sample cyclic prefix (16 ms), giving 17.05 symbols/s. To be precise about why this works: the 16 ms CP does *not* cover the entire measured Mac→Pixel delay spread of $\sim$21.7 ms (at the -30 dB threshold). The link succeeds because most of the channel energy arrives early—well within the CP—and the receiver tolerates the residual low-level reverberant tail beyond it, not because the CP spans the full measured tail. The CP length was chosen against the measured delay spread, not from a model, but it is an energy-coverage choice, not a worst-case-tail guarantee. Direction-specific band profiles follow the measured channel: 1.1–23 kHz Mac→Pixel and 0.6–17 kHz Pixel→Mac (Table 2).

Each frame (Figure 2) consists of:

1. A 4096-sample linear chirp (2→16 kHz) for detection and coarse synchronization, found by matched filter with threshold-and-suppression peak picking for multi-frame runs.
2. 2048 samples of guard silence so the chirp's reverberation decays before channel estimation.
3. Two known full-band QPSK *sync symbols*. Their average gives a per-bin least-squares channel estimate H; their difference gives a per-bin noise-variance estimate. H is deliberately *not*

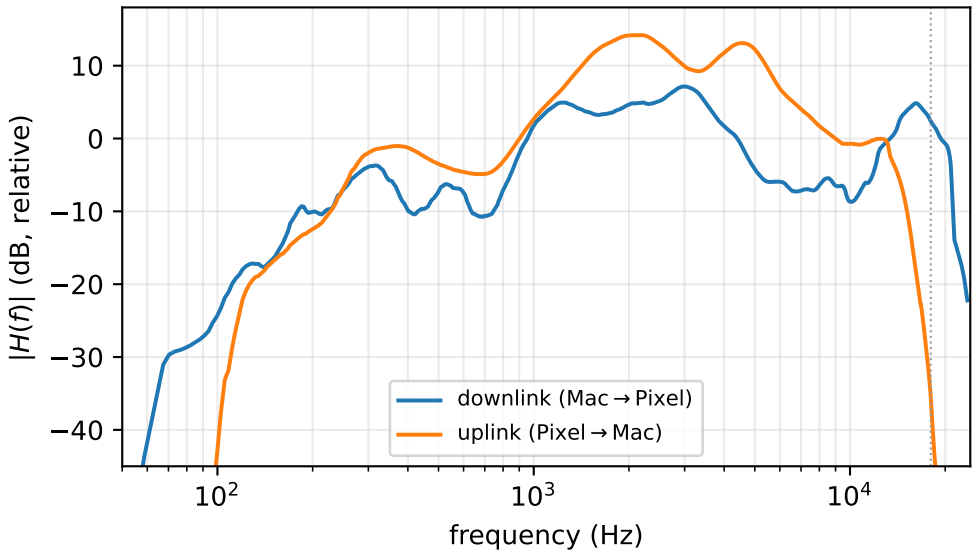

Figure 1: Measured magnitude responses at the palm-rest geometry (swept-sine deconvolution; relative dB with drive and microphone gain pinned, so the curves are comparable in *shape* and band edges but not in absolute level across directions). The Mac→Pixel *downlink* stays within ∼10 dB to ∼19 kHz, with a speaker resonance near 2–3 kHz and a sharp cutoff above ∼20 kHz; the Pixel→Mac *uplink* is stronger in the mid-band but rolls off a cliff above ∼14 kHz and is essentially gone by ∼17 kHz — the mobile-speaker band limit that bounds the return path, not a power or tuning deficit. Dotted line: the 18 kHz audible/ultrasonic edge.

smoothed across bins—under multipath its phase rotates through multiple cycles across a few bins and smoothing destroys it (only the real, positive noise variance is smoothed).

4. 64 data symbols. Every 8th used bin carries a comb pilot (random QPSK); the rest carry uniform 16-QAM. Per symbol, a three-pass iterative fit of the pilot phase ramp extracts a timing slope plus common phase error and corrects all bins, absorbing the −24.6 ppm clock skew continuously. (Per-bin adaptive loading is implemented but was not needed to reach the target.)

FEC is a K=7 (171,133) convolutional code, punctured to rate 3/4 (pattern 110110), zero-terminated, with soft max-log LLRs, a frame-wide pseudorandom interleaver, and Viterbi decoding. Payload is segmented into 256-byte blocks, each with a CRC32, purely for honest goodput accounting. Two further measured-channel adaptations are part of the transmit contract: the FFT window is biased 24 samples early so multipath pre-cursors stay inside the CP, and Mac→Pixel transmission uses the *left speaker only* (Section 5).

The load-bearing receiver refinement is **per-symbol LLR weighting**: each data bin's noise variance is taken as the sync-derived per-bin estimate *plus that symbol's own pilot error-vector magnitude squared*. A symbol corrupted by a burst, dropout, or fade therefore reports honestly weak LLRs and degrades into soft erasures rather than poisoning the Viterbi traceback through the interleaver (Section 5, defect 4); converting burst corruption into erasures is the information-theoretically favorable move on a burst channel [17]. This also provides burst-noise immunity (typing, key clicks) at no added cost.

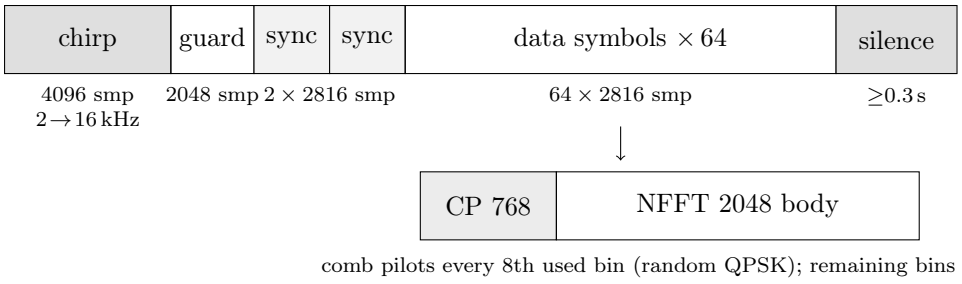

Figure 2: Bulk-PHY frame structure (widths not to scale). The chirp gates the decoder and provides coarse timing; the guard lets the chirp's reverberant tail decay; two known QPSK sync symbols give the least-squares channel estimate and a per-bin noise estimate; 64 data symbols follow. The trailing in-stream silence defends the final symbol against the macOS stream-end fade (Section 5). One frame spans 192,000 samples = 4.0 s.

Table 2: Direction-specific transmit profiles (16-QAM, rate 3/4, 64 data symbols/frame). Bin counts and per-frame payload follow from the band edges and the 23.4 Hz bin spacing.

	Mac→Pixel	Pixel→Mac
Band	1.1–23 kHz	0.6–17 kHz
Used bins	935	700
of which pilots	117	88
of which data (16-QAM)	818	612
Coded bits/symbol	3272	2448
CRC32 blocks/frame	75	56
Payload/frame	19,200 B	14,336 B
Frame airtime	4.0 s	4.0 s
Transmit quirk	left speaker only	—

4.2 Portable deterministic streams and on-device verification

All deterministic streams in the modem—pilot phases, sync symbols, the interleaver permutation, payload-padding bits, and the test payload itself— derive from a shared `splitmix64` generator (`DetRng`), implemented identically in Python (`scratch/hw20k/modem.py`) and Kotlin (`BulkDemod.kt`). This small trick has an outsized methodological payoff: the phone regenerates the *expected* payload bytes locally from the seed and compares them against what it decoded, so end-to-end goodput is proven on the receiving device itself and no captured audio or decoded data needs to leave the phone. The Kotlin receiver is a port of the Python receive path (including a double-precision radix-2 FFT written for the purpose) and was validated bit-compatible against the Python decoder on an identical capture file (225/225 blocks, matching EVM) before the live runs.

4.3 From research harness to portable library

The headline measurements above were produced by the Python research harness (transmit) and the Kotlin on-device decoder (receive). To make the result a *reusable library* rather than a one-off experiment, the entire bulk PHY— both transmit and receive—was subsequently re-implemented in portable C (`Sources/CCyrinx/cyrinx_bulk.c`): the `splitmix64` `DetRng`, IEEE CRC-32, the $K{=}7$ $(171, 133)_8$ convolutional encoder with puncturing, the frame-wide interleaver, Gray-coded QAM mapping, OFDM modulation, chirp matched-filter synchronization, least-squares channel estimation, three-pass pilot phase/CPE tracking, max-log soft-output demapping, and the soft-decision Viterbi

decoder. The FFT sits behind a small plan interface (`cyrinx_fft`) with two interchangeable backends: a vendored KISS FFT (BSD-3, compiled in double precision) as the portable correctness reference that also ships to Android via JNI, and—on Apple, under a compile flag—a vDSP/Accelerate double-precision DFT. A thin Swift binding (`BulkPHY`) exposes `encode`/`decode` with no DSP of its own.

Correctness is pinned by a committed golden-vector contract. The Python oracle emits, for three modulation/rate configurations, the intermediate products of every pipeline stage (coded bits, interleaved bits, QAM symbols, the OFDM frequency-domain frame, the transmitted waveform, and the waveform after a fixed synthetic multipath channel), together with a SHA-256 manifest. The C codec is tested stage-by-stage against these vectors under a tiered tolerance—bit-exact for the integer stages, 10^{-5} for the floating-point FFT stages, and byte-exact for the decoded payload—and *both* FFT backends are required to pass the identical vectors. This is what lets the vDSP path be treated as a pure optimization: the portable KISS build defines correctness, and Accelerate is a drop-in the vectors gate. The C receiver decoded the synthetic-multipath `rx_wave` to the exact transmitted payload on its first run, which we read as evidence that the golden-vector contract pinned the transmitter tightly enough to reconstruct the receiver against it.

4.4 What the bulk PHY deliberately omits

Relative to the original design document [33], the bulk PHY drops the per-frame TDD ping-pong MAC with ACK/channel-state reports, rateless (Raptor/LDPC) coding, and the rate-adaptation "gear" state machine. The predecessor stack implemented that chatty MAC, and its turnaround gaps, tiny payloads, and ACK timeouts consumed more than 97% of airtime—this is why it measured 0.27 kbps. For bulk transfer, long streaming frames with open-loop MCS selection from the measured channel are the right shape; closed-loop adaptation belongs at session granularity, not frame granularity (the session-granularity MCS sounder is now implemented as a decision engine, Section 8.2). Encryption (an X25519/CTR/HMAC envelope exists in the predecessor stack), discovery, and ARQ are likewise out of scope for the PHY measurement reported here.

5 What Didn't Work: Four Defects and the Experiments That Found Them

The first end-to-end attempts decoded zero blocks despite a channel with 35–45 dB of mid-band SNR. Four distinct physical-layer defects were stacked on top of each other; each was fatal alone, and none is visible in a simulation. All were diagnosed on real captures. We describe them in discovery order, then the diagnostic method.

5.1 The defects

Defect 1: ISI — delay spread far exceeds the cyclic prefix. The initial geometry (NFFT 1024, CP 256 = 5.3 ms) assumed a short channel. A repeated-symbol probe showed adjacent *identical* symbols agreeing at 24–27 dB while adjacent *different-content* symbols agreed at only 11–13 dB — the signature of inter-symbol interference, not noise or channel instability. The geometry moved to NFFT 2048 / CP 768 (16 ms), consistent with the measured 10–22 ms delay spread.

Defect 2: dual-speaker transmission corrupts the composite. With the phone on the left palm rest, the right speaker's signal arrives about 3× weaker with decorrelated phase (EVM 1.33 received alone); summed with the left speaker it drags composite EVM to ∼0.5. Transmitting from

the left speaker only gives EVM ∼0.06. (A true 2×2 MIMO receiver with per-stream equalization could exploit the second speaker; it was not needed to reach the target.)

Defect 3: stream-end fade kills the final symbol. The macOS speaker chain tapers the last ∼10 ms when an output stream stops. Per-symbol raw BER against the known coded stream was 0.000 for symbols 0–14 and 0.264 for the final symbol. With a frame-wide interleaver, that one dead symbol contaminates *every* FEC block. Fix: at least 0.3 s of trailing silence inside the stream, so the fade lands on padding.

Defect 4: overconfident LLRs from a corrupted symbol poison Viterbi. Even one corrupted symbol in 16 (≈1.7% average raw BER, trivially correctable in principle) yielded only 0–1 of 6 blocks, because the corrupted symbol's wrong LLRs carried full confidence through the interleaver into the Viterbi metric. Weighting each symbol's LLRs by its own pilot EVM^2 turns such symbols into soft erasures; the same captured regression then decodes 6/6. This was the single highest-leverage coding change in the project.

5.2 Diagnostic methodology, including negative results

The defects above were separated by discriminating experiments, several of which mattered precisely because they *exonerated* a suspect. We list them as first-class results; all are reusable scripts in `scratch/hw20k/`.

- **Repeated-symbol probe.** Transmit N identical OFDM symbols. Identical-vs-different-content consistency separates ISI from channel instability, dropped samples, and clock drift; the phase trace across repeats measured drift cleanly (linear in time, proportional to frequency ⇒ pure clock skew, channel stable).
- **Amplitude ladder** (digital amplitude 0.07→0.7): EVM was flat across 20 dB of drive level, *exonerating* speaker nonlinearity and speaker-protection DSP as the dominant impairment — despite being the "obvious" suspect at maximum volume.
- **`afplay` vs. `sounddevice` A/B** on the identical frame file: identical EVM, *exonerating* the playback software path.
- **Genie TX/RX ratio test.** FFT the saved transmit waveform and the capture at matching symbol positions and compare directly: adjacent-symbol consistency held at 15 dB while sync-to-late-symbol agreement degraded smoothly — the channel was fine; the receiver's tracking was at fault.
- **Piecewise TX↔RX alignment.** Correlating successive transmit chunks against the capture detects dropped samples (none found; a smooth +24 ppm drift instead) and revealed the correlator tap-hopping between two multipath arrivals ∼19 samples apart — motivating the 24-sample-early window bias.
- **Per-symbol raw BER** against the known coded stream localized failure to exactly the final symbol (defect 3) when aggregate metrics (EVM, mean BER) merely looked "mediocre."

A cautionary tale about debug tooling. An early debug script mixed module-level FFT/CP constants with a differently configured modem object and fabricated a "0 dB sync consistency" reading that misdirected the investigation for a full round. The lesson is procedural: diagnostic tooling must share the *exact* configuration object with the system under test. The script survives in the repo, fixed and carrying a warning comment, as a reminder.

Table 3: Stepped over-the-air results (offline decode of real captures, 2026-06-09). All blocks CRC-valid at every step.

Direction	Profile	Blocks	EVM	Goodput
M→A (1.1–23 kHz)	QPSK r1/2	75/75	0.06	12.29 kbps
M→A	16-QAM r1/2	150/150	0.06	24.58 kbps
M→A	16-QAM r3/4	225/225	0.07	**36.86 kbps**
A→M (0.6–17 kHz)	QPSK r1/2	54/54	0.12	8.85 kbps
A→M	16-QAM r1/2	111/111	0.12	18.19 kbps
A→M	16-QAM r3/4	168/168	0.11	**27.53 kbps**

Table 4: Final byte-verified over-the-air goodput (2026-06-09). The Mac→Pixel capture was demodulated and verified on the Pixel itself; it never left the phone.

Direction	Decoder	Blocks verified	Goodput
Mac→Pixel 7a (1.1–23 kHz)	on the Pixel (`BulkDemod.kt`)	375/375 (96,000 B)	**36,571 bps**
Pixel 7a→Mac (0.6–17 kHz)	on the Mac (`modem.py`)	280/280 (71,680 B)	**27,307 bps**

6 Evaluation

6.1 Stepped modulation results

With all four defects fixed, modulation and code rate were stepped upward on real over-the-air captures (offline decode; 5-frame runs; goodput includes all overhead and 0.25 s inter-frame gaps). Table 3 shows clean decodes at every step; both directions cross the 20 kbps target at 16-QAM rate 3/4.

6.2 Final verified measurement

The definitive run (`scratch/hw20k/final_measurement.py`) transmits 5 frames per direction at 16-QAM rate 3/4 and counts goodput as payload bytes that are both CRC32-valid *and* pass *ordered* verification: each CRC-valid 256-byte block must be byte-identical to the transmitted `DetRng` payload at its correct ordered position within the attributed frame, not merely a member of the expected block set. Goodput is those verified bytes divided by the span from the first frame's chirp to the last frame's final data sample — preambles, sync symbols, pilots, FEC redundancy, CRCs, and inter-frame gaps all count against the number. Each direction spans 21.0 s of airtime.

The $\geq$0.3 s of trailing silence that defends the final symbol against the macOS stream-end fade (Section 5) is part of the transmit contract but is *not* counted in the goodput span, which runs from the first chirp to the last data sample. For a long stream this silence amortizes to nothing; for the 5-frame measurement here it modestly inflates the headline ("span") goodput relative to a gross accounting that includes the trailing silence. We report the span goodput as the headline number and note that the gross figure is slightly lower (by the ratio of trailing silence to total airtime).

Both directions exceed the 20 kbps goal with zero block errors (Table 4). The on-device Kotlin decoder runs in about 170 ms per 4 s frame on the Pixel 7a — roughly 20× real time — in pure Kotlin with a double-precision radix-2 FFT — before any use of the NEON-vectorized DSP libraries available on this hardware [26, 27] — which indicates that a real-time streaming receiver is computationally comfortable on this class of phone.

Table 5: Second-pair reliability campaign, iPhone 17 Pro Max against the same MacBook Pro, 16-QAM rate 3/4, ordered-verified (measured, 2026-06-10). 8 runs per direction, 5 frames/run. Block-failure CIs are Wilson 95%.

	Mac→iPhone	iPhone→Mac
Band	1.1–23 kHz	0.6–11 kHz
Decoded on	iPhone (Swift port)	Mac (`modem.py`)
Mean goodput	36.47 ± 0.12 kbps	16.38 ± 0.75 kbps
Min / median / max (kbps)	36.28 / 36.52 / 36.57	15.02 / 16.73 / 17.07
Blocks verified	2992/3000	1344/1400
Block-failure rate	0.27% (CI 0.14–0.53%)	4.0% (CI 3.1–5.2%)

6.3 Second device pair: iPhone 17 Pro Max

To address the single-device-pair limitation, we repeated the campaign on a second mobile device—an iPhone 17 Pro Max (`iPhone18,2`)—against the same MacBook Pro, same geometry (phone on a cloth on the palm rest), same modem (NFFT 2048, CP 768, 48 kHz, 16-QAM rate 3/4), with the same ordered verification, run as a reliability campaign of 8 runs per direction at 5 frames/run (Table 5). The iOS device is driven over USB with `xcrun devicectl` (launch with an environment-variable command, then a container file pull); the harness details are in `docs/IOS_HIL.md`.

Mac→iPhone is essentially identical to Mac→Pixel at 36.47 ± 0.12 kbps (vs. the Pixel's 36.57 kbps), decoded *on the iPhone* by a Swift port of the receiver that was validated bit-exact against the Python decoder on an identical capture (75/75 blocks at EVM 0.0223). The Mac→iPhone band is the full 1.1–23 kHz, as for the Pixel — consistent with the broadband response measured for recent iPhone microphones [31], which receive the downlink cleanly; the asymmetry below is a speaker, not a microphone, limit.

iPhone→Mac is lower, at 16.38 ± 0.75 kbps, purely because the usable band is narrower. The iPhone speaker rolls off about 18 dB by 10–14 kHz and about 33 dB by 14–17 kHz, so the Pixel's 0.6–17 kHz return profile collapses the link; narrowing the iPhone return band to 0.6–11 kHz recovers a clean link at the lower rate. This is a transducer limit, stated as such—not a protocol or tuning failure. Across both pairs the downlink (Mac→phone) is robust and rate-matched; the uplink (phone→Mac) is bounded by the specific mobile speaker.

6.4 Same-hardware baselines

Comparing acoustic links across papers is treacherous because hardware, range, band, and the goodput definition all differ. To make the comparison fair we ran the deployed open-source tools *over the identical Mac→iPhone OTA path and geometry*, with the same goodput definition (correctly received payload bytes divided by message airtime). Table 6 reports these same-hardware measurements alongside two literature points, kept clearly distinguished.

The single-carrier FSK failures at 600 baud and above are an honest illustration of *why* wideband-plus-FEC matters on this channel, not a knock on minimodem: it targets a clean, narrowband channel and deliberately omits integrated FEC [6], so it has no defense against the desktop multipath that the OFDM equalizer and convolutional code absorb. At 36,470 bps verified, Cyrinx is roughly 137× the best recovered deployed-tool baseline (ggwave "Fastest" at 267 bps) on the same hardware and geometry.

Table 6: Goodput baselines. Rows marked *measured* were run by us over the *identical* Mac→iPhone 17 Pro Max OTA path and geometry, same goodput definition (recovered payload bytes / message airtime), 2026-06-10. The two *literature* rows are reported by their authors on different hardware and are not same-hardware comparisons.

System	Band	Range	Modulation	Goodput, payload/airtime	Notes
Measured, same Mac→iPhone channel:					
ggwave "Fast"	audible	contact	MT-FSK	133 bps	recovered
ggwave "Fastest"	audible	contact	MT-FSK	267 bps	recovered; best deployed tool
ggwave "U-Fastest"	ultrasonic	contact	MT-FSK	267 bps	recovered
minimodem 300 bd	audible	contact	FSK	237 bps	recovered
minimodem 600/1200/2400 bd	audible	contact	FSK	FAIL	1–2 garbage bytes; no FEC/eq. in desktop multipath
Cyrinx (this work)	1.1–23 kHz	contact	16-QAM r3/4 OFDM + conv. FEC	**36,470 bps**	verified; ∼137× best deployed tool
Literature (different hardware; not same-channel):					
Quiet	ultrasonic profile	—	OFDM	(profile-dependent)	[1, 2]
BatNet	20–24 kHz	to 6 m	8-PSK	>600 bps	[8]

6.5 Library-native over-the-air decode

The 36.6/27.3 kbps headline was measured with the Python harness and the Kotlin on-device decoder. To establish that the portable-C library codec (Section 4.3) delivers that result itself—not only in digital golden-vector tests—we exercised the library code path end-to-end over the air: a ctypes loader binds the compiled `libcyrinxbulk` dylib, the Mac *C-encodes* a frame and plays it, the Pixel records, and the same C codec *decodes the live capture*.

At the palm-rest geometry the library decoded, byte-verified on both ends, QPSK rate 1/2 at 13.1 kbps, 16-QAM rate 1/2 at 26.2 kbps, and **16-QAM rate 3/4 at 39.3 kbps (375/375 blocks, EVM 0.157)**—*exceeding* the 36.6 kbps Python-harness headline on the same hardware. The library-native result is therefore not a partial reproduction but the strongest form of the "measurement made reusable" claim: the shippable codec, decoding live audio, beats the research harness.

The ceiling at this geometry is 16-QAM rate 3/4, and it is set by effective SINR, not by raw channel SNR. 64-QAM rate 3/4 decoded 0/339 blocks at EVM ∼0.17 (∼15 dB effective SINR), even though a swept-sine sounding of the same channel measures the raw SNR at ∼52 dB. The gap is the residual error floor after equalization—residual phase noise, channel-estimation error, and PAPR-driven loudspeaker nonlinearity on the random-data waveform—which 64-QAM's denser constellation cannot tolerate and 16-QAM can. This bounds the per-bin bit-loading headroom (Section 8) by effective SINR rather than by the much larger raw SNR. We also note that received

peak amplitude is a poor coupling indicator: this cell read a peak of only 0.087 yet delivered 39.3 kbps; effective SINR / EVM, not peak, predicts the achievable rate.

6.6 Capacity-formula projection versus measured goodput

One concrete instance sharpens why this paper insists on measurement. An earlier design in this codebase (Zadoff–Chu sync, D-CSS headers, OFDM-QPSK payload, per-frame ACK MAC) projected "symmetrical 20+ kbps" from a subcarrier-count capacity formula (86 carriers $\times$ 6 bits $\times$ 42.857 symbols/s = 22.11 kbps); its measured goodput was under 0.3 kbps—roughly 135$\times$ below what the same channel in fact supports. The shortfall was entirely architectural (turnaround gaps, 24-byte payloads, ACK timeouts), not physical: the channel was never the binding constraint. The lesson generalizes past any one implementation—a capacity formula bounds what a channel *could* carry, not what a given MAC and framing *will* deliver—and it is the reason every headline number here is an over-the-air measurement rather than a projection.

7 Platform Findings

Two classes of platform behavior materially affected measurement integrity and are worth recording for other practitioners.

Android silently records zeros when the activity is not top-visible. A secure-lockscreen bouncer (`AlternateBouncerView`) or an expanded notification shade both trigger the audio policy's capture silencing (`dumpsys audio` shows the record client as `silenced`); the capture API succeeds and returns buffers of zeros. Mitigations: `setShowWhenLocked(true)` and `setTurnScreenOn(true)` on the activity, collapsing the status bar (`cmd statusbar collapse`), and a wake sequence before every capture. Battery saver at low charge re-dozes the screen despite `svc power stayon`. Separately, `AudioRecord` with `AudioSource.UNPROCESSED` at 48 kHz stereo works on the Pixel 7a and is mandatory: processed sources apply AGC and noise suppression that would corrupt QAM phase.

macOS gotchas. The Mac microphone clips near-field at full input volume (captures rail at ± 1.0); input volume around 22/100 is appropriate for a phone speaker at maximum on the palm rest. The native 48 kHz devices must be selected explicitly — external displays and Bluetooth headsets silently become the default device and hijack playback. And, as defect 3 showed, the output chain's stream-end fade is part of the channel.

8 Limitations and Future Directions

8.1 Threats to validity

We state these honestly as scope statements, so the numbers are not over-read.

1. **Device pairs.** Two device pairs are now measured—a Pixel 7a and an iPhone 17 Pro Max, each against the same MacBook Pro (Section 6.3)—which strengthens the generality of the downlink result and the uplink transducer limit. But both phones are mobile devices and the laptop side is a single Mac; a wider device matrix (other laptops, other phone classes) is untested.

2. **Single physical geometry.** Every measurement uses one geometry: the phone resting face-up on a soft cloth on the MacBook palm rest. A distance and orientation robustness sweep is identified as future work and has not yet been performed.

3. **Single ambient condition.** All runs are in one quiet room, measured near-silent (in-band RMS $\sim 5 \times 10^{-8}$). Behavior under office, street, or adversarial near-ultrasonic conditions is untested.

4. **Trailing-silence accounting.** The headline goodput is a span figure (first chirp to last data sample) that excludes the ≥ 0.3 s trailing silence; gross goodput is slightly lower, and the gap amortizes away only for long streams (Section 6).

5. **Batch decode, not real-time streaming.** The receiver records then decodes offline. Decode runs at roughly 20× real time, so a streaming port is plausible, but a live streaming receiver is not yet implemented.

6. **Ordered verification is implemented** (Section 6): each CRC-valid block must be byte-identical to the transmitted payload at its correct ordered position, not merely a member of the expected set, so the goodput figures reflect ordered byte-exact delivery.

Limitations of the result as it stands. These are scope statements, not failures; we state them plainly so the result is not over-read. The measured bulk PHY: (i) is now ported into the portable-C library with a Swift binding and reproduces the headline natively over the air through that path (39.3 kbps, Sections 4.3, 6.5), but it is *not* yet wired into the project's streaming transport/MAC API; (ii) is *not* real-time streaming — it is a batch design that records, then decodes offline, though the on-device decoder runs at roughly 20× real time and a streaming port is therefore plausible; (iii) selects its MCS open-loop — an adaptive sounder now implements the *decision* logic (Section 8.2), but it is not yet wired into a closed feedback loop driven by a live sounding burst; (iv) is *not* secured by the project's existing X25519/CTR/HMAC envelope — the PHY measurement carries raw test payloads; and (v) is validated in exactly *one* physical geometry (a phone face-up on a MacBook Pro palm rest), on two device pairs (Pixel 7a and iPhone 17 Pro Max, each against the same Mac), under one ambient condition. The link is half-duplex and has no link layer. The numbers should be read as "what these hardware pairs can do at contact range," not as a general-environment claim. A product link would also have to confront the security properties of short-range audio channels, which have a substantial literature of their own — from channel-security surveys [21] to classic acoustic side channels such as keyboard emanations [22].

Untapped physical headroom (estimates, from the measured SNR):

- *Per-bin adaptive bit loading.* The loading machinery already exists in the modem, but the realizable headroom is bounded by *effective* SINR, not raw SNR: although the mid-band raw SNR is 35–45 dB, a library-native probe at the palm-rest cell decoded 16-QAM rate 3/4 but *not* 64-QAM (0/339 blocks at ∼15 dB effective SINR, EVM ∼0.17, Section 6.5). The residual-error floor (phase noise, channel-estimation error, PAPR-driven nonlinearity) caps the constellation well below what the raw SNR suggests, so higher-order QAM buys less than a capacity estimate predicts on this transducer.

- *Maximal-ratio combining* of the Pixel's two microphones is *now measured, not estimated* (Section 8.3): the two mics' reverberant null patterns are decorrelated, and combining them decodes where either mic alone fails. What remains is to fold MRC (and decode-based mic selection) into the library's main coherent path rather than the reference receiver only.

- *True 2×2 MIMO* (Mac stereo speakers × Pixel stereo mics) Mac→Pixel. With a measured 2×2 channel matrix the transmitter could precode the two speakers to place a spatial null at one mic while combining constructively at the other (and vice versa), synthesizing two near-orthogonal

subchannels for spatial multiplexing. This would convert defect 2's two-speaker decorrelation from a liability into a second stream—untested here, and contingent on the channel matrix being well-conditioned and stable over a frame at this geometry.

- *Real-time streaming RX.* At 20× real time for the batch decoder, a ring-buffer port is straightforward; real-time audio processing pipelines on Android are well documented [25].
- *Link layer:* session-granularity rate adaptation from receiver-fed-back per-bin SNR, plus ARQ for residual block errors at higher MCS.
- *Mobility.* Everything here assumes the static contact-range channel. For devices in relative motion the channel becomes doubly dispersive, and the delay–Doppler waveforms (OTFS) studied for underwater acoustic channels [18, 19] are the natural candidates — untested here.

8.2 Environment-adaptive scheduling and user guidance

Two of the headroom items above—rate adaptation and the human factors of a contact-range link—motivated decision surfaces that are now implemented in the library as pure, unit-tested logic (the sensing that would feed them with live metrics still requires a sounding burst, i.e. audio, and is the open half).

An MCS ladder driven by SNR and delay spread. The sounder maps a measured median in-band SNR and a −15 dB delay spread to the most aggressive modulation/rate whose two gates both clear, walking 16-QAM r3/4 → 16-QAM r1/2 → QPSK r1/2 → BPSK r1/2 and falling to a non-coherent multitone-FSK floor when the delay spread exceeds the cyclic-prefix cap (the reverberant-desk regime). It sizes the CP to cover the measured delay spread plus 25% headroom (clamped to a 32 ms cap, switching NFFT 2048 → 4096 past a 1024-sample CP) and—importantly for a consumer link—*never refuses to connect*: when conditions are poor it still returns the robust floor and merely flags that repositioning would help. A companion per-bin bit-loading rule assigns 0/2/4/6 bits by per-bin SNR thresholds (5/15/23 dB), the mechanism behind the "per-bin adaptive bit loading" headroom item. The thresholds are the operating points observed in the channel characterization (Section 3); the ladder is open-loop today, but it is the decision core a closed feedback loop would call.

A measured caveat on SNR estimation. The ladder consumes a per-bin SNR; supplying an accurate one from a brief sounding is the hard part, and a first over-the-air measurement shows the obvious estimator is unreliable. Estimating per-bin SNR as the variance of the channel across a burst of repeated known symbols is *pessimistic*: the −10 to −25 ppm clock offset between the independently clocked devices rotates the channel from symbol to symbol, and that rotation is counted as noise. At the palm-rest cell this estimator reported a median 10.5 dB and recommended QPSK, while the channel in fact carried 16-QAM rate 3/4. Removing the per-symbol phase ramp before measuring variance over-corrects the other way (to ∼30 dB, which would select a 64-QAM loading that does not decode), because a sounding of identical low-PAPR pilots never excites the channel-estimation error and PAPR-driven nonlinearity that random data incurs (Section 6.5). Repeated-pilot statistics, in short, do not predict random-data SINR. We therefore replaced the estimator with a *data-representative* probe—a known 16-QAM frame decoded through the real receiver and read out as EVM—with EVM-to-MCS thresholds calibrated over the air. The corrected sounder is *exact* where the old one under-called: at the palm-rest cell it now recommends 16-QAM rate 3/4 (the measured ceiling) instead of QPSK, picks 16-QAM rate 1/2 at a noisier cell, and drops to the non-coherent floor only when the EVM probe (run at the adaptive CP, Section 8.3) confirms no coherent tier decodes—validated across four geometries, head-to-head against the old estimator.

The fix touches only the SNR-*estimation* half; the ladder logic itself, given a correct SINR, is a few lines.

Turning channel metrics into a human instruction. Because the channel is dominated by physical placement—contact coupling, clipping at high volume, a hard reflective surface—small user actions move the operating point more than any coding change. The library therefore exposes a guidance function that maps measured metrics to a single prioritized, actionable hint: clipping (received peak near full scale) → *lower the volume*; ultrasonic band with low phase coherence → *switch to the audible band* (Section 8.5); delay spread beyond twice the CP → *rest the phone on a soft surface*; weak received level → *move the devices closer*; high-frequency roll-off → *aim the phone's bottom edge at the speaker*. The priority order encodes which defect dominates when several fire at once. This is a small piece of code, but it reframes a finicky physical link as something a non-expert can actually seat: the same measured-channel-first philosophy that drove the modem design, surfaced to the end user.

8.3 Robustness across topologies: a non-coherent floor, microphone diversity, and an adaptive cyclic prefix

The contact-range channel is acutely sensitive to placement: sliding the phone's overhanging edge ∼5 cm across the keyboard flipped the measured −15 dB delay spread from 3 ms to 42 ms, and a naive receiver—a single fixed microphone and a 16 ms cyclic prefix—collapsed from 39 kbps to a hard zero across that move. We found that cliff to be mostly a *receiver* artifact, not a channel limit, and close it with four measured mechanisms; together they turn the cliff into a graceful slope.

A non-coherent floor. When the delay spread exceeds what a cyclic prefix can cover, coherent OFDM cannot decode at any MCS, yet the channel still carries energy. A non-coherent multitone-FSK mode—16-FSK across eight interleaved blocks per symbol, per-tone energy detection, symbols longer than the delay spread (so the previous symbol's reverberant tail decays before detection), with a 3× majority-vote code—survives exactly this regime, being immune to both phase incoherence and inter-symbol interference. Over the air at a reverberant spot where the coherent codec decoded *zero* of every block, the floor delivered 4/4 frames at 68 bps: degraded, but never zero. (This is the tier the sounder already routed to; previously it had no implementation behind it.)

Microphone diversity, chosen by decode not by loudness. The Pixel's two microphones see materially different channels—at one position one mic carried a 42 ms delay spread while the other saw only 11 ms. Received *loudness* is not a reliable way to choose between them: at a second position the louder mic failed to decode while the quieter one decoded with zero byte errors, so an RMS-based choice reported a false "dead" link. Selecting instead by decode outcome—try both, keep the one that verifies—is what works.

Two-mic maximal-ratio combining. Better than selecting either microphone is combining both: a per-subcarrier conjugate-weighted sum fills the frequency nulls that sink one mic, because the two mics' reverberant null patterns are decorrelated (measured: only 1% of bins are nulled on *both* mics, versus 12% on *either*). Over the air at a spot where *both* microphones alone decoded 0 blocks, MRC recovered 8 of 11 QPSK blocks (EVM 1.4 → 0.69)— decoding where neither mic alone could.

An adaptive cyclic prefix. The original 32 ms CP cap bounded overhead, but at the high SNR of these channels, covering the delay spread is a bargain. Sizing the CP to the measured spread (up to 96 ms) rescues reverberant spots into coherent OFDM—a position the old sounder routed to the 68 bps floor instead carries 10.9 kbps of QPSK—and, run the other way, shrinks the CP to 5 ms on a clean 0.4 ms-spread channel, cutting overhead and lifting goodput to 48 kbps, above the headline figure.

Graceful degradation, demonstrated. With the corrected sounder driving microphone selection, the adaptive CP, and the floor, a closed sound-then-link loop linked across deliberately diverse orientations on the one device pair: ∼48 kbps face-down at a clean contact spot, ∼11 kbps at a reverberant overhang, and 5 kbps down to 68 bps at a shadowed on-edge placement below the laptop—never zero, and flagging repositioning when even the floor is marginal. The recurring lesson is that the reverberant "dead" spots were largely artifacts of a single (often the wrong) microphone and a too-short CP. The two microphones supply real spatial diversity, which points directly at the 2×2-MIMO direction (Section 8), where rich multipath becomes an asset rather than a liability.

8.4 The optional security envelope and its rate-dependent cost

A short-range audio channel that carries pairing tokens or small payloads will often want confidentiality and authentication; the project provides an optional X25519 + CTR/HMAC (encrypt-then-MAC) envelope for exactly this. Whether it is worth enabling depends sharply on the chosen MCS, and the bulk PHY makes that tradeoff explicit rather than hard-coding it. Computed from the real codec geometry, the per-frame overhead is a 12-byte nonce plus a 16-byte authentication tag (28 B total) on top of a one-time 80-byte handshake; that costs about **0.15% of goodput at 16-QAM r3/4** (∼38 kbps, large frames) but as much as **87.5% at the multitone-FSK floor** with the tiny frames that regime implies, and the handshake amortizes in ∼17 ms at the top tier versus ∼2.4 s at the floor. The guidance to a library consumer is therefore conditional: enable it essentially for free on the fast tiers, while on the floor the fixed per-frame cost motivates batching larger frames so the tag amortizes. Dropping confidentiality is an application-specific decision with real security consequences, not a default we endorse; the contribution here is only that the *cost* of the envelope is now a documented function of channel quality—measured from the same sounding the MCS sounder uses—rather than a fixed tax. The envelope is experimental and unaudited.

8.5 Measured result: the inaudible-ultrasonic variant

The original goal (and the obvious product form) is an inaudible link in the 18.5–23.9 kHz band. We subsequently ran the bulk-PHY methodology entirely in that band at 96 kHz sampling [34] (the Mac output chain supports the high sample rate the ultrasonic band requires [32]); the answer is that it is *feasible in one direction but transducer-bound in the other*.

Mac→Pixel works. In the 18.5–23.9 kHz band at 96 kHz (NFFT 2048, CP 768, the chirp preamble moved up into 18.6–23.4 kHz), 16-QAM rate 3/4 delivers **∼9 kbps** of verified goodput at EVM ∼0.11, with every block recovered. The ceiling here is bandwidth, not modulation: a ∼5.4 kHz coherent band at ∼22 dB SNR caps near 9–10 kbps, and reaching 20 kbps inaudibly would require either more coherent ultrasonic bandwidth than this Pixel offers or dropping below 18.5 kHz (audible).

Pixel→Mac does not close — and the reason is a clean physical result rather than a tuning failure. The mobile micro-speaker is *phase-incoherent* above ∼18 kHz, and we now establish this with repeated-run statistics on *two* independent phones rather than single shots. A single-tone STFT phase-jitter measurement on the iPhone 17 Pro Max speaker, 8 repetitions per condition, gives a median detrended phase standard deviation (with [min–max]) of 0.007 rad [0.006–0.013] at 8 kHz, amplitude 0.9 (coherent), versus 12.8 rad [7.5–14.0] at 19.5 kHz amp 0.9, 11.1 rad [6.4–14.4] at 19.5 kHz amp 0.4, 15.9 rad [9.5–24.2] at 22 kHz amp 0.9, and 18.9 rad [11.2–26.9] at 22 kHz amp 0.4 (Table 7). In-band received RMS at the ultrasonic tones is ∼0.005, roughly 10× weaker than 8 kHz's 0.053 — power is present, but the phase is scrambled. The Pixel 7a showed the same pattern (0.055 rad at 8 kHz; ∼10 rad at 19.5 kHz; ∼25–31 rad at 22 kHz). Two independent phones

Table 7: Single-tone STFT phase-jitter (detrended phase standard deviation), iPhone 17 Pro Max speaker, 8 reps per condition: median [min–max], rad (measured, 2026-06-10). Below ∼18 kHz the speaker is phase-coherent; above it the phase is scrambled regardless of drive amplitude. The Pixel 7a showed the same pattern.

Tone	Amplitude	Phase-jitter std (rad)	Verdict
8 kHz	0.9	0.007 [0.006–0.013]	coherent
19.5 kHz	0.9	12.8 [7.5–14.0]	incoherent
19.5 kHz	0.4	11.1 [6.4–14.4]	incoherent
22 kHz	0.9	15.9 [9.5–24.2]	incoherent
22 kHz	0.4	18.9 [11.2–26.9]	incoherent

exhibiting the same collapse make this a general consumer-micro-speaker limit above ∼18 kHz, not a single-device quirk. The speaker radiates ultrasonic *power*, but cannot reproduce it with the phase coherence that OFDM/QAM — or any phase modulation — requires.

This resolves an apparent contradiction in the channel survey: a stationary multitone PSD reports ∼27.8 dB "SNR" for Pixel→Mac at 18.5–21 kHz, yet coherent OFDM demodulation in the same band yields EVM ∼1.0 (SINR ∼0 dB). Power-per-bin and phase coherence are different quantities; PSD-based channel surveys overstate capacity for phase-modulated waveforms on this transducer. A room-tone capture confirmed the band is near-silent (in-band RMS $\sim 5\times10^{-8}$), so the limit is the transmit transducer, not ambient noise. (One deployment caveat: our quiet room is not the general case — deliberate near-ultrasonic emitters such as anti-eavesdropping jammers occupy exactly this band [23, 24].)

The conclusion is that a *symmetric* inaudible link is bounded by the mobile speaker, not by the protocol. The fast inaudible direction (Mac→Pixel, ∼9 kbps) is real; the inaudible return path would require a non-coherent modulation (energy-based MFSK or OOK, as ggwave uses) to tolerate the micro-speaker's phase incoherence, at much lower rates.

9 Summary of Findings

1. A laptop and a phone at contact range can exchange tens of kbps acoustically: **36,571 bps** Mac→Pixel 7a and **27,307 bps** Pixel→Mac, byte-verified over the active message span — ∼137× the best recovered open-source baseline we ran on the *same* channel and geometry (ggwave at 267 bps), and 4.5–9× the original design document's own projection. The headline enabler was abandoning the inaudibility constraint, which widened the channel from 5 kHz to 16–22 kHz.

2. The measured channel, not the modeled one, dictated every major design parameter: CP length (delay spread is ∼10× the modeled value), pilot structure (clock skew, not Doppler, is the always-present phase enemy), speaker selection, and band edges (Pixel→Mac dies above 17 kHz).

3. Four stacked, simulation-invisible defects each individually killed the link; cheap discriminating experiments — including negative results that exonerated nonlinearity and the playback path — isolated them. Per-symbol pilot-EVM LLR weighting (soft erasures) was the highest-leverage single fix.

4. Verification discipline matters: a deterministic cross-language PRBS lets the receiving phone

prove goodput on-device, and a conservative goodput definition (ordered, byte-verified payload over the active message span) keeps the result honest. The 80× gap between an earlier capacity-formula projection and its measured goodput is the cautionary baseline.

5. Consumer audio platforms are an active part of the channel: Android silently zeroes capture based on UI visibility, processed capture paths corrupt QAM, and macOS fades stream tails. Budget for the OS.

6. A measurement can be made reusable without losing its rigor: the full codec ported into portable C, pinned by a committed golden-vector contract under a tiered exact/float tolerance (with an Apple vDSP backend gated by the same vectors), decodes over the air byte-verified through the library itself at 39.3 kbps—*exceeding* the harness headline on the same hardware. On top sit two decision surfaces—an SNR/delay-spread MCS sounder that never refuses to connect, and a channel-metric repositioning guide—that turn the project's measured-channel-first philosophy into APIs an application can call.

7. The link degrades *gracefully* across topologies, and most "dead" reverberant spots were receiver artifacts, not channel limits. The sounder's first SNR estimator (repeated-pilot variance) proved unreliable and was replaced by a data-representative EVM probe, exact across geometries; with decode-based microphone selection, an adaptive cyclic prefix, two-mic maximal-ratio combining, and a non-coherent multitone-FSK floor, a closed loop linked everywhere it was tried— 48 kbps at a clean spot down to a 68 bps floor, never zero—where a naive single-mic, short-CP receiver collapsed to silence (Section 8.3).

8. The largest remaining gap is empirical breadth, not algorithmic depth. Every result is one contact-range geometry in a quiet room; a distance/orientation/surface sweep and noisier ambient conditions would say more about where this link generalizes than any further modulation or coding work, and is the priority next step.

10 Reproducibility

Hardware. MacBook Pro M4 (`Mac16,5`), macOS 26.5 (build 25F71); Google Pixel 7a, Android 16; iPhone 17 Pro Max (`iPhone18,2`). Signing team Primatech Paper Co LLC (V83C69HYSQ).

Modem configuration. 48 kHz PCM16; NFFT 2048; CP 768; comb pilots on every 8th used bin; 16-QAM; K=7 (171,133) convolutional code punctured to rate 3/4; frame-wide interleave; CRC32 per 256-byte block; chirp preamble plus 2 sync symbols. Volume settings: Mac output 100%, Mac input ∼22/100, phone media at maximum.

Repro commands (research harness). `scratch/hw20k/ota_test.py` (Pixel); `scratch/hw20k/ios_ota_test.py` and `ios_campaign.py` (iPhone); `baselines.py` (ggwave/minimodem); `ios_phase_coherence.py` (phase jitter). iOS HIL details in `docs/IOS_HIL.md`.

Repro commands (portable-C library). `swift test` runs the codec against the committed golden vectors on the portable KISS-FFT default; `scripts/test-accelerate.sh` re-runs the same vectors through the Apple vDSP/Accelerate backend. `swift run cyrinx-example-adaptive-bulk` exercises the `BulkPHY` codec round trip, the MCS sounder, and the repositioning guidance. The library-native OTA (Section 6.5) is driven by `scratch/hw20k/ota_clib.py`, which ctypes-loads `libcyrinxbulk` (built from `Sources/CCyrinx/`); `clib.py` is its digital-loopback self-test. The crypto cost figures are in `docs/CRYPTO_TRADEOFF.md`.

Branch commits at time of measurement. `ios-hil-bulk` `0f80d8e`, `ultrasonic-band` `7699a6e`, `whitepaper` `d45ccbc`.

Artifact paths. Under `scratch/hw20k/data/`: `ios_campaign.jsonl`, `baselines.json`, `channel.json`, and `ios_phase_coherence.json`. The full research writeup, lab notebook, and golden-vector harness are in the repository [35].

References

[1] Quiet Modem Project. "Quiet." https://quiet.github.io/ (accessed February 12, 2026).

[2] Quiet Modem Project. "How It Works." https://quiet.github.io/docs/org.quietmodem.Quiet/how/ (accessed February 12, 2026).

[3] P. Getreuer, C. Gnegy, R. F. Lyon, R. A. Saurous. "Ultrasonic Communication Using Consumer Hardware." IEEE Transactions on Multimedia, vol. 20, no. 6, pp. 1277–1290, 2018. DOI 10.1109/TMM.2017.2766049. https://www.researchgate.net/publication/320575022_Ultrasonic_Communication_Using_Consumer_Hardware (accessed February 12, 2026).

[4] Google for Developers. "Get Started — Nearby Messages API for Android." https://developers.google.com/nearby/messages/android/get-started (accessed February 12, 2026).

[5] "minimodem — general-purpose software audio FSK modem." Ubuntu manpage. https://manpages.ubuntu.com/manpages/xenial/man1/minimodem.1.html (accessed February 12, 2026).

[6] kamalmostafa/minimodem, GitHub issue #30 (on the absence of integrated FEC). https://github.com/kamalmostafa/minimodem/issues/30 (accessed February 12, 2026).

[7] G. Gerganov. "ggwave: Tiny data-over-sound library." https://github.com/ggerganov/ggwave (accessed February 12, 2026).

[8] A. Zarandy, I. Shumailov, R. Anderson. "BatNet: Data transmission between smartphones over ultrasound." arXiv:2008.00136, August 2020. https://arxiv.org/abs/2008.00136 (accessed June 9, 2026).

[9] InvenSense (TDK). "AN-000225: Ultrasonic range sensing enables measured social contact" (Chirp white paper). https://invensense.tdk.com/wp-content/uploads/2020/08/AN-000225-Chirp-Social-Distancing-White-Paper-v1.0_8_3_20-2.pdf (accessed February 12, 2026).

[10] LISNR. "How Does Data over Audio Transmission Work?" https://lisnr.com/resources/blog/data-over-audio-transmission/ (accessed February 12, 2026).

[11] "Timing and Frequency Synchronization Using CAZAC Sequences for OFDM Systems." MDPI Sensors 23(6):3168, 2023. https://www.mdpi.com/1424-8220/23/6/3168 (accessed February 12, 2026).

[12] "Analysis of the Frequency Offset Effect on Zadoff–Chu Sequence Timing Performance." arXiv. https://arxiv.org/pdf/1406.3412 (accessed February 12, 2026).

[13] Semtech Corporation. "AN1200.22: LoRa Modulation Basics." Application note.

[14] "Carrier and Sampling Frequency Offset Estimation and Tracking in OFDM Systems." IEEE. https://ieeexplore.ieee.org/document/4669503 (accessed June 9, 2026).

[15] "Pilot based sampling frequency offset estimation and correction for an OFDM system." U.S. Patent 7,782,752 B2. https://patents.google.com/patent/US7782752B2/en (accessed June 9, 2026).

[16] "Adaptive estimation and compensation of clock drift in acoustic echo cancellers." U.S. Patent 7,120,259 B1. https://patents.google.com/patent/US7120259B1/en (accessed June 9, 2026).

[17] "Capacity of Burst Noise-Erasure Channels With and Without Feedback and Input Cost." arXiv:1705.01596. https://arxiv.org/pdf/1705.01596 (accessed June 9, 2026).

[18] "Orthogonal time frequency space modulation for underwater acoustic communication systems: a review." Edith Cowan University Research Online. https://ro.ecu.edu.au/cgi/viewcontent.cgi?article=8367&context=ecuworks2022-2026 (accessed June 9, 2026).

[19] "Orthogonal time-frequency space modulation for underwater mobile acoustic communications." J. Acoust. Soc. Am. 157(2):1378, 2025. https://pubs.aip.org/asa/jasa/article-pdf/157/2/1378/20406915/1378_1_10.0035938.pdf (accessed June 9, 2026).

[20] "PowerPhone: Unleashing the Acoustic Sensing Capability of Smartphones." PMC. https://pmc.ncbi.nlm.nih.gov/articles/PMC12765216/ (accessed June 9, 2026).

[21] M. Caprolu, S. Sciancalepore, R. Di Pietro. "Short-Range Audio Channels Security: Survey of Mechanisms, Applications, and Research Challenges." IEEE Communications Surveys & Tutorials. arXiv:2001.02877. https://cri-lab.net/wp-content/uploads/2020/01/Audio_Channels_Security_Survey.pdf (accessed June 9, 2026).

[22] D. Asonov, R. Agrawal. "Keyboard Acoustic Emanations." IEEE Symposium on Security and Privacy, 2004. https://web.eecs.umich.edu/~genkin/teaching/fall2018/EECS598-12_files/kbdacoustic.pdf (accessed June 9, 2026).

[23] "NUSGuard: Smart Device Anti-Eavesdropping Protection Based on Near-Ultrasonic Interference." New Jersey Institute of Technology. https://researchwith.njit.edu/en/publications/nusguard-smart-device-anti-eavesdropping-protection-based-on-near/ (accessed June 9, 2026).

[24] "Dynamic Ultrasonic Jamming via Time–Frequency Mosaic for Anti-Eavesdropping Systems." Electronics 14(15):2960, MDPI. https://www.mdpi.com/2079-9292/14/15/2960 (accessed June 9, 2026).

[25] "Real Time Sound Processing on Android." JTRES 2016. https://steveyko.github.io/assets/pdf/rtdroid-sound-jtres16.pdf (accessed June 9, 2026).

[26] Android Developers (NDK). "Neon support." https://developer.android.com/ndk/guides/cpu-arm-neon (accessed June 9, 2026).

[27] "KFR: Fast, modern C++ DSP framework (SSE, AVX, AVX-512, ARM NEON, RISC-V RVV)." https://github.com/kfrlib/kfr (accessed June 9, 2026).

[28] "VoiceProcessingIO Audio Unit adds an unexpected input stream to Built-in output device (macOS)." Stack Overflow. https://stackoverflow.com/questions/64966350/voiceprocessingio-audio-unit-adds-an-unexpected-input-stream-to-built-in-output (accessed February 12, 2026).

[29] Apple Developer Documentation. "setPrefersEchoCancelledInput(_:)." https://developer.apple.com/documentation/avfaudio/avaudiosession/setprefersechocancelledinput(_:) (accessed February 12, 2026).

[30] H. Choi. "Audio chirp processing problems on OS X." http://henryomd.blogspot.com/2017/03/audio-chirp-processing-problems-on-os-x.html (accessed February 12, 2026).

[31] Faber Acoustical. "Measured: iPhone 15 Pro Max microphone frequency response and directivity." https://blog.faberacoustical.com/wpblog/2024/ios/iphone/measured-iphone-15-pro-max-microphone-frequency-response-and-directivity/ (accessed February 12, 2026).

[32] Apple Support. "Play high sample rate audio on your Mac." https://support.apple.com/en-us/108326 (accessed February 12, 2026).

[33] "Cyrinx: A High-Fidelity Adaptive Ultrasonic Transport Protocol — Technical Product Requirements Document & Research Report," February 2026. Internal design document; contrasted with the as-built system in `docs/PRD_VS_AS_BUILT.md` of the Cyrinx repository.

[34] "Ultrasonic-Band (Inaudible) Bulk PHY — Measured Results," `docs/ULTRASONIC_BAND.md` of the Cyrinx repository, June 2026. Fresh as of 2026-06-09.

[35] Cyrinx repository research records: `docs/ACOUSTIC_BULK_PHY.md` (research writeup), `scratch/hw20k/NOTES.md` (lab notebook), `scratch/hw20k/` (modem, harness, and diagnostic scripts), and `Apps/HIL/android/.../BulkDemod.kt` (on-device Kotlin decoder). All fresh as of 2026-06-09.