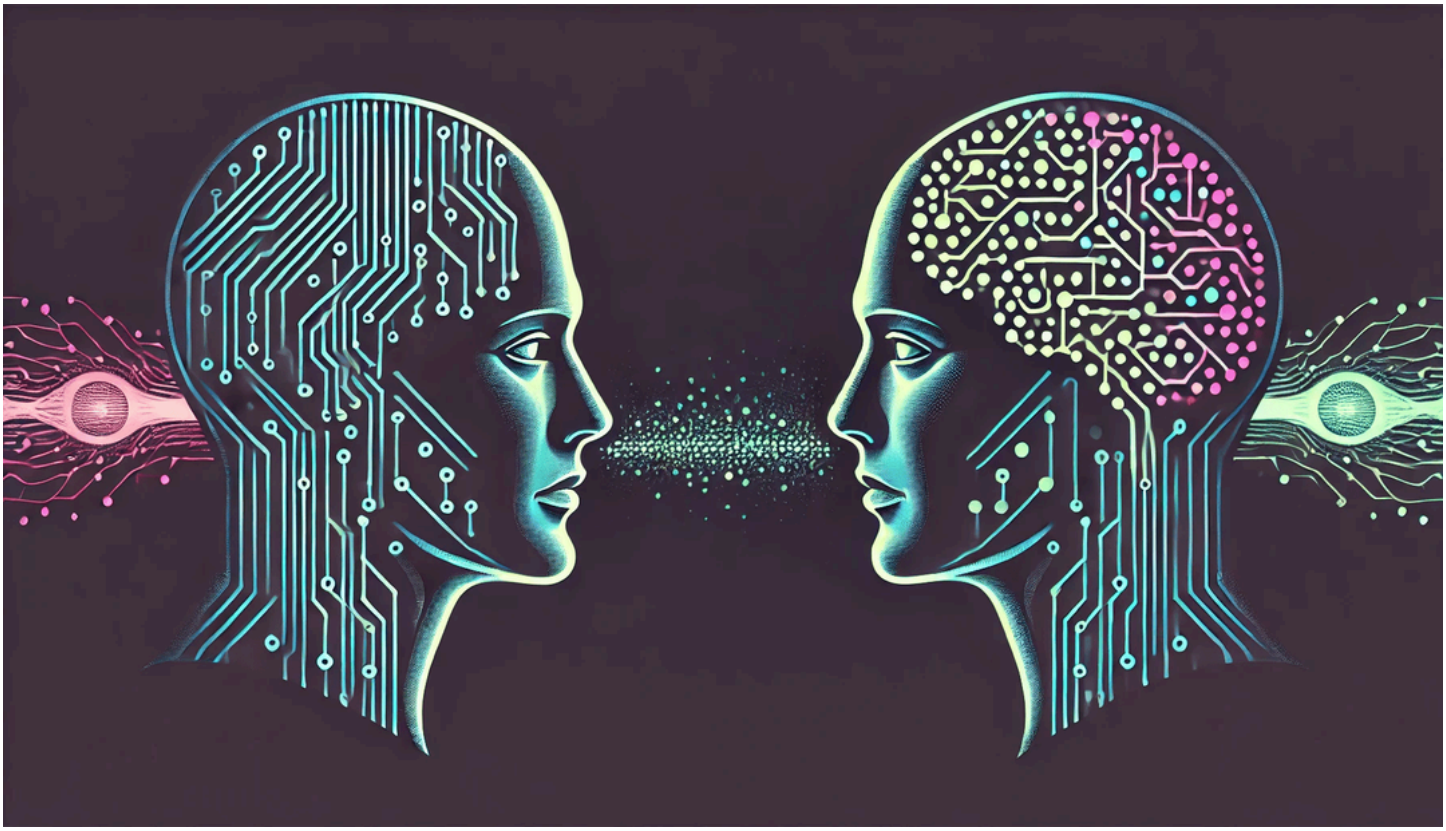
STA 561D
Press Release

GIBBERJAB

INTRODUCING GIBBERJAB: A COMPACT PROTOCOL FOR AI AUDIO EXCHANGE

Revolutionary Encoding Protocol Lets Conversational AI Agents and IoT Devices Communicate Directly through Highly Compressed, Machine-Readable Audio "Beeps," Reducing Latency, Improving Security, and Enabling Offline ZeroTrust Environments



DURHAM, NC

April 20, 2025—Today, Engineers at Duke University unveiled **GibberJab**, a cutting-edge audio-based embedding protocol designed specifically for Conversational AI (CAI) and IoT devices. **GibberJab** reimagines digital dialogue, enabling AI models to bypass traditional, verbose natural language in favor of ultra-compact, losslessly compressed exchanges sent as sequences of audio tones. Built upon deep neural compression trained on extensive literary corpora, this technology dramatically reduces latency, enhances data security, and creates new opportunities for zero-trust IoT communication.

Unlike common network-based protocols, GibberJab uses physical audio—emitted via speakers, received via microphones—as its primary transport layer, creating a standardized machine language for local data exchange that never touches traditional wired or wireless network stacks. Applications range from voice assistant protocol handshakes to zero-trust device onboarding, all leveraging audio as a secure, infrastructure-independent channel.

Today, GibberJab operates within the audible frequency spectrum (0–20 kHz), enabling communication with standard hardware available on nearly all consumer smart devices. While a simple modification to our code allows for ultrasonic (inaudible) operation, current deployments are set to audio frequencies perceptible to the human ear. Planned enhancements will extend support to fully ultrasonic data transmission—enabling truly imperceptible AI-to-AI exchanges in privacy-critical or human-occupied environments.

How GibberJab Works

Functionally, GibberJab is structured as follows:

1. **Semantic Deep Compression**
   The protocol utilizes a custom-trained TextCompressor neural network, leveraging a large corpus of classic English literature from Project Gutenberg. The system analyzes textual frequency patterns, identifying optimal words and character sequences to achieve lossless compression with ratios typically between 1.0x and 1.3x. This optimized dictionary allows natural language text to be encoded into ultra-compact forms.
2. **Neural Quantized Embedding**
   GibberJab employs a Transformer-based neural architecture utilizing positional encoding and self-attention mechanisms. This model encodes sentences into fixed-size vectors (64-dimensional embeddings), where each dimension is quantized into one of 16 discrete symbols, significantly reducing representation size and enhancing data transfer efficiency.
3. **Audio Transmission via GGWave**
   Encoded embeddings are translated into short waveform patterns using the GGWave acoustic modulation library. These waveforms are transmitted at high frequencies, enabling secure, imperceptible, and interference-resistant acoustic data communication without traditional network connections.

Importantly, the protocol is fully agent-aware: when conversational AI agents identify one another as interlocutors—and jointly confirm willingness to negotiate a compressed protocol—they switch to the GibberJab "embedding" mode, terminating verbose exchanges in favor of direct, compressed, beep-based transmission. The message handler automatically splits lengthy content into numbered, byte-size-aware chunks for reliability and reassembly. All data

payloads can be encoded and decoded without loss using only the shared compression dictionary and embedding model parameters.

“In designing GibberJab, we saw an opportunity to fundamentally improve the speed, efficiency, and privacy of AI and IoT communication,” said Thomas Jablonski, Eagle Scout & Lead Communications Engineer at GibberJab. “Our team built an open-source, robust implementation that lets devices speak a language optimized for machines, not for people—without sacrificing data integrity or opening additional attack surfaces on the network. By staying strictly within local audio hardware and making the system modular, we’re paving the way for ubiquitous, secure digital dialogue between smart devices.”

Getting started is as simple as downloading the Python SDK and integration snippets, which cover everything from loading the compressor, segmenting messages, to orchestrating audio playback and listening using PyAudio and GGWave. The repository includes the full Python source code for the encoder model, compressor utilities, and transmission scripts, alongside setup instructions, pre-trained compressor artifacts, sample text corpora, Jupyter notebooks for exploration, and example clients for sending and receiving acoustic messages. Supporting files for configuration and dependency management are also provided, enabling fast setup, full local control over model behavior, and easy customization for different hardware environments.

To learn more or contribute, visit our project repository (https://github.com/braydenhand/GibberJab_NN)

FAQ

Customer Needs & Value Proposition

Who are your customers?

Our primary customers include:

- IoT manufacturers and enterprise IT departments aiming to enhance zero-trust environments.
- Developers of conversational AI agents looking to optimize efficiency and privacy.
- Organizations prioritizing secure, low-latency internal AI communication.

How does GibberJab improve customer experiences?

GibberJab drastically reduces latency by replacing lengthy natural language exchanges with highly compressed high frequency signals. It enhances security by eliminating reliance on network connections, reducing cyber-attack surfaces, and providing privacy through undetectable acoustic communication.

How do you measure success?

Success is measured by:

- Adoption rate among IoT and AI developers.
- Latency and energy savings metrics.
- Customer satisfaction and reduced security incident rates.

Core Technology & Usage

How does GibberJab integrate with existing AI and IoT products?

Integration is straightforward via our provided SDK, supporting Python environments using PyAudio, GGWave for audio transmission, and TensorFlow/PyTorch-compatible embedding models. The SDK includes pre-trained compressors and clear integration documentation.

What hardware requirements exist?

Minimal additional hardware is needed. Standard microphones and speakers supporting at least 48 kHz sampling rates are sufficient. Most existing loT and smart devices already meet these specifications.

Does GibberJab require internet connectivity?

No. GibberJab uniquely supports fully offline, zero-trust communication. Once initial embedding dictionaries and neural models are loaded, no network connection is required.

Can human ears detect GibberJab communication?

In its default mode, yes-GibberJab transmits data as short, rapid tones within the human hearing range, optimized for clarity and speed in standard acoustic environments. However, for privacy-sensitive scenarios, Synthesis can be seamlessly adapted to ultrasonic frequencies (above 20 kHz), rendering the communication inaudible to human ears while maintaining machine interpretability.

At what rate does GibberJab emit tones?

GibberJab is currently configured to ggwave's default protocol ID–6, which corresponds to a "Base64 encoded message" profile that is higher quality (it is slightly slower but more reliable for longer messages). This profile utilizes a 48,000 Hz sampling rate, and a tone/symbol duration of 40ms.

What level of latency and bit-rate can customers expect in real-world deployments?

Measured latency and bit-rate depend heavily on environment and message size. In typical conditions, GibberJab can transmit short compressed messages in under a second per chunk, but sub-100ms end-to-end exchange has not yet been rigorously benchmarked. Estimated semantically meaningful bit-rates range from several hundred bytes per second up to 1–2kBps in ideal acoustic settings.

How robust is the communication over audio?

Balancing embedding compactness with decoding accuracy and efficiency sparked The communication is highly robust due to:

- High frequency usage, minimizing interference.
- Byte-size-aware chunking ensuring all messages fit within transmission constraints.
- Error-resistant embedding and decoding protocols.

Security, Privacy, & Reliability

What are the implications if malicious actors attempt to jam or spoof audio channels?

At present, GibberJab compresses and transmits data without cryptographic signing or session authentication. Malicious interference, jamming, or spoofing could disrupt transmission integrity. Planned future updates aim to introduce cryptographic protections, anomaly detection, and adaptive measures like frequency-hopping, but these features are not yet implemented in the current system.

Are there contingencies for accidental information leakage or eavesdropping by nearby unauthorized parties?

GibberJab uses compression that obfuscates the original message content but does not employ cryptographic encryption. Without access to the trained compression dictionary, reconstructing intercepted transmissions would be difficult but not impossible. Future updates are planned to add encryption and symbol randomization for stronger protection against unauthorized decoding.

What is the risk if the protocol's compression model is reverse-engineered or "leaks"?

If the compression dictionary were compromised, intercepted transmissions could potentially be partially decoded. GibberJab does not yet include per-session encryption, so periodic manual retraining of the dictionary and updating of devices is the primary method of risk mitigation. Future plans include automatic dictionary rotation and enhanced cryptographic protection.

How does GibberJab ensure accessibility and inclusivity for users who may rely on assistive listening devices or have heightened sound sensitivity?

GibberJab can operate in higher-frequency bands but is not set to transmit exclusively ultrasonic signals under current settings. For environments requiring strict auditory safety (e.g., near hearing aids or cochlear implants), minor configuration would be needed to restrict output to verified ultrasonic-only modes.

Limitations, Pitfalls & Contingency Plans

What happens if devices are in extremely noisy environments or areas with heavy acoustic interference?

While GibberJab is robust to standard room noise and typical acoustic environments (e.g., HVAC, speech, music), extremely high-noise conditions—such as heavy machinery or highly reflective spaces—may temporarily degrade transmission accuracy. GibberJab supports retransmission of missing chunks but does not currently implement full error correction codes (FEC) or real-time acoustic error detection beyond chunk-level recovery.

How does GibberJab handle failed message delivery or dropped audio 'chunks'?

Chunked transmission is managed with in-band sequence numbering in the message headers (e.g., [2/4]). Devices listen for all expected chunks and can reassemble multi-part messages upon receipt. However, automatic retransmission requests for missing chunks are not currently implemented; missing or incomplete messages must be manually resent if recovery fails.

Can GibberJab scale to environments with dozens or hundreds of devices transmitting simultaneously?

Currently, GibberJab is designed for single-stream communication between devices and does not implement device registration, explicit channel assignment, or time-division multiplexing. Scalability features for coordinating many devices, including dynamic collision avoidance and spectrum management, are planned for future versions but not yet available.

What happens if legacy devices without GibberJab support are present within hearing range?

Legacy devices will typically ignore GibberJab transmissions, as the data is acoustically modulated and not recognized as meaningful input. GibberJab traffic remains effectively invisible to non-participating devices unless those devices record and attempt manual analysis of the audio.

How is protocol updating, such as compressor dictionary or neural embedding model refresh, managed across devices?

GibberJab currently uses a static compression dictionary (TextCompressor) that must be manually retrained and redistributed. There is no automatic model synchronization or authenticated over-audio update mechanism yet. Future improvements are intended to add secure, incremental updates and verification procedures.

Developer Access

Is GibberJab open to third-party developers?

Yes. Developers are actively encouraged to integrate GibberJab via our comprehensive SDK available on GitHub, in hopes of promoting community-driven innovation and rapid adoption of our project.

For more details and SDK access, see our project repository (https://github.com/braydenhand/GibberJab_NN)