gr-PHYSEC: Real-time Channel-based Key Generation for Physical Layer Secure Wireless Communications

Jose A. Sanchez Viloria JOSESANCHEZ2019@FAU.EDU
George Sklivanitis GSKLIVANITIS@FAU.EDU
Dimitris Pados DPADOS@FAU.EDU
Center for Connected Autonomy and AI (CA-AI.fau.edu), Florida Atlantic University, Boca Raton, FL, USA

Abstract

Securing wireless communication against eavesdropping is critical, particularly in dynamic and decentralized environments. We present *gr-PHYSEC*, a new GNU Radio out-of-tree (OOT) module for real-time physical-layer key generation. Unlike traditional key generation that relies on pre-shared secrets or computational complexity, our approach derives symmetric keys from the wireless channel's inherent randomness. We embed a trained neural network within GNU Radio to extract channel features between trusted parties (Alice and Bob) during probe exchanges. These features are quantized into binary keys, reconciled via Reed-Solomon encoding, and further secured with SHA-256 hashing. The generated keys are then directly used to encrypt data. Real-world experiments at the FAU CAAI connected robotics testbed using ADALM Pluto software-defined radios and NVIDIA Jetson Orin validate the approach with ground robotic platforms. Results demonstrate low key disagreement rates and strong randomness, as verified by the NIST test suite for random and pseudorandom number generators for cryptographic applications. This integration showcases how GNU Radio can support real-time AI-driven security solutions, pushing the boundaries of software-defined secure communication. The source code for this project is available at:

https://github.com/C2A2-at-Florida-Atlantic-University/gr-PHYSEC.

Proceedings of the 15th GNU Radio Conference, Copyright 2025 by the author(s).

1. Introduction

Wireless links expose traffic to passive and active adversaries; yet, symmetric key generation and exchange remains a difficult challenge for distributed, resource-constrained devices in infrastructure-free ad-hoc networks. Physical-layer key generation leverages channel reciprocity, temporal variation, and spatial decorrelation to derive similar features from over-the-air probes without prior information exchange (Zhang et al., 2016b;a). Prior systems have quantized received signal strength (RSS), channel state information (CSI), or channel impulse response (CIR) and reconciled errors via interactive protocols or forward error correction (FEC) (Liu et al., 2013; Zeng et al., 2010; Mathur et al., 2008). In parallel, deep learning at the physical layer (PHY) has proven effective for robust feature extraction, automatic modulation classification and radio-frequency (RF) fingerprinting application (O'Shea & Hoydis, 2017; Wang et al., 2017; Shen et al., 2021).

Existing implementations of real-time key generation follow the same pipeline for channel probing, quantization, reconciliation, and privacy amplification but differ on channel sounding as well as processing techniques. A GNU radio LoRa SDR implementation demonstrated end-to-end probing, quantization, error-correction on long range based networks (Hu et al., 2023). Other GNU Radio implementations that aimed to facilitate physical layer security techniques on GNU Radio involved a multiple-input single-output (MISO) system that uses CSI from a target receiver to inject noise as well as a single-carrier Alamouti coding system for undoing phase shifts of a pseudo-random sequence for decoding a signal (Ryland et al., 2017).

In this paper, we demonstrate for the first time real-time physical layer key generation for on-the-fly data encryption in peer-to-peer ad-hoc wireless communications. We implement and test on GPU-enabled software-defined radios GNU Radio out-of-tree (OOT) signal processing blocks for key generation between two trusted communicating parties, Alice and Bob. 1 depicts the end-to-end AI key generation process from training to deployment and testing. We

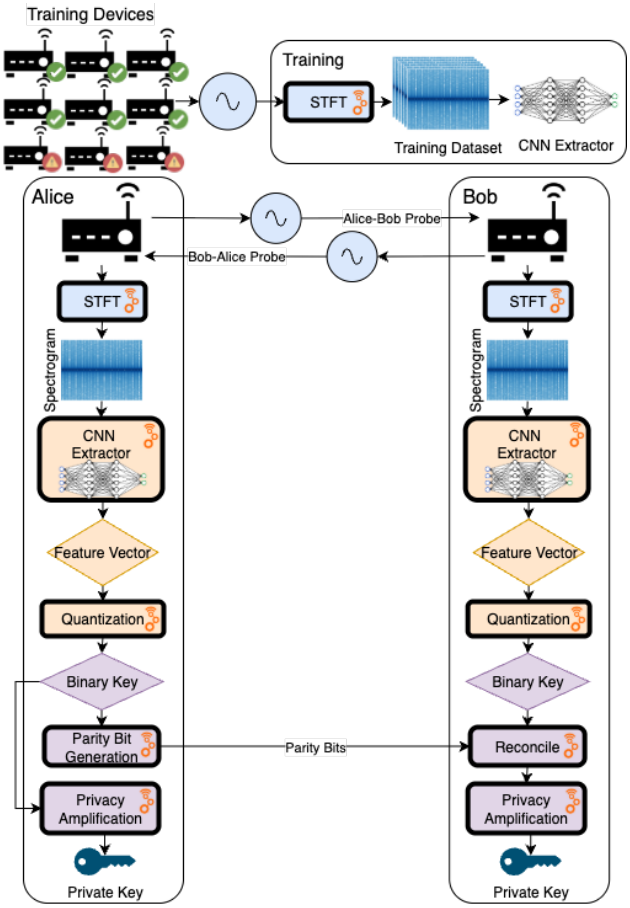


Figure 1. System overview of the AI-assisted physical layer key generation system implemented in GNU Radio.

trained a convolutional neural network (CNN) with quadruplet loss to map IQ-level spectrograms of channel probes to compact channel embeddings. Alice and Bob use their respective channel embeddings to generate a bit vector long enough to render its estimation by a brute-force attack computationally impractical no matter the computing resources of an eavesdropper. Hardware impairments, slow exchange of channel probes and mobility may impact channel reciprocity, therefore there may be differences between the generated keys. Alice and Bob follow a reconciliation algorithm based on a Reed-Solomon (RS) error correction coding technique. To minimize the amount of information leaked to Eve during the reconciliation process, the SHA-3 is used to map the reconciled bit vector to a random bit sequence of 512 bits i.e., the desired cryptokey.

We contribute to the software-defined radio community an open-source GNU Radio OOT project titled gr-PHYSEC, that implements the key generation pipeline (described in Fig. 1) in real time with COTS SDR hardware, controlled by Jetson Orin GPU platforms on ground robotic platforms. We expose simple PMT-based control messages for control flow implementation, and ready-to-run flowgraphs. gr-PHYSEC includes GNU Radio OOT blocks for orchestrating channel probing between two SDRs, short-time Fourier Transform (STFT) spectrogram generation, deep learning inference from the Open Neural Network Exchange (ONNX), channel feature quantization, RS encoding/decoding for key reconciliation, and privacy amplification via SHA-512 hashing. We also developed message-oriented state machines that coordinate probing between a pair of SDRs and parity bit exchanges for the reconciliation algorithm including retransmissions and an acknowledgment protocol. The project source code is available open-source via github including the pre-trained AI feature extractor model, and Docker containers to reproduce embedded deployment on NVIDIA Jetson platforms and ADALM-Pluto SDRs.

2. Secret Key Generation Protocol

Figure 1 depicts our proposed secret key generation protocol implemented in GNU Radio. Two legitimate/trusted parties (Alice/Bob) implement bi-directional channel probing. Each receiver generates spectrograms based on received IQ samples. Each party extracts channel embeddings based on a pre-trained CNN feature extractor that is installed at each SDR node. The neural network output is quantized to a 512-bit initial secret message. Alice uses an RS encoder, and then transmits her parity bits to Bob to reconcile. Bob combines the received parity bits with his initial secret message to reconcile bit disagreements using an RS decoder. The final step of the protocol includes privacy amplification to enhance the randomness of the desired cryptographic key key (Yıldırım et al., 2025; Pelekanakis et al., 2021; Sklivanitis et al., 2021; Zhang et al., 2016a; Rukhin et al., 2010; MacWilliams & Sloane, 1977).

2.1. Channel-based feature extraction

During bi-directional channel probing, each SDR node buffers L=8192 IQ samples. We compute an STFT using a Hamming window N=256 and 50% overlap (R=128), forming an $M \times N$ log-power spectrogram (M=31) (Allen, 1977). A 15-layer ResNet CNN model maps the 31×256 spectrogram input to a 512×1 element feature vector embedding using L2-norm and bounded activation for stable quantization (He et al., 2016). We use the quadruplet loss function to train the neural network. The objective is to minimize Alice$\leftrightarrow$Bob channel embeddings while maximizing the distance between Alice$\rightarrow$Eve and Bob$\rightarrow$Eve embeddings where Eve is a third SDR node playing the role of a passive eavesdropper. Our goal is to preserve channel reciprocity and boost spatial decorrelation.

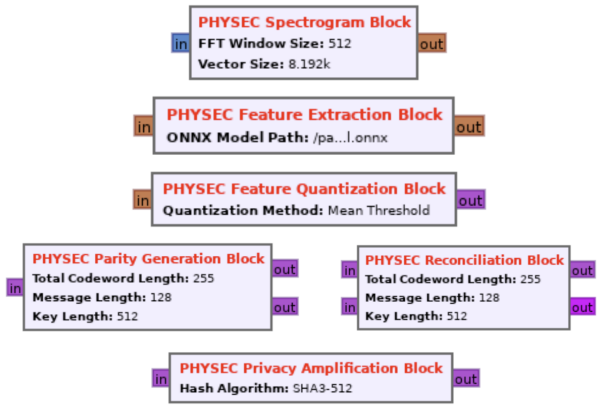


Figure 2. gr-PHYSEC blocks: spectogram generation, feature extractor, feature quantization, parity bit generation, reconciliation, and privacy amplification.

2.2. Key generation

After channel probing and feature extraction, Alice and Bob proceed with quantizing the channel embeddings to generate an initial secret message. Due to non-perfect channel reciprocity and hardware impairments, we except that there will be bit disagreements between the two messages. Let us denote by RS(C,K) the RS encoder that takes as input $K = L/P$ symbols (or L bits where P represents the number of bits per symbol) and maps it into a codeword of $C = 2^P - 1$ symbols. The first K symbols of the RS codeword are identical to the input symbols while the remaining $S = C - K$ symbols correspond to the parity symbols. We use $C = 255$ and $K = 128$ and consider that Bob receives Alice's generated parity symbols with no errors. By adding S parity symbols to his data, Bob's RS(C,K) decoder can detect any combination of up to and including S erroneous symbols, or locate and correct up to and including $T = \lfloor S/2 \rfloor$ erroneous symbols at unknown locations. Finally, Alice and Bob apply the SHA-3-512 has function to the output bit vector from the reconciliation stage to produce an output bit vector of fixed length, i.e., the desired cryptographic key of 512 bits. The properties of the SHA-3 hash function guarantee that even if Eve's reconciled vector has 1-bit difference from the Alice's message, her output will be completely different than that of Alice and Bob (Rukhin et al., 2010).

3. gr-PHYSEC

We developed six GNU Radio OOT blocks to enable secret key generation as well as additional blocks to control channel probing and information exchange between the two SDRs. Figure 2 shows the individual blocks while Figures 8 and 9 show the GNU Radio flowgraphs for Alice and Bob, respectively.

Spectrogram Generation. A Python block pre-processes the collected IQ samples using STFT (NumPy/FFT) and log-power scaling to produce spectrograms.

Feature Extraction. This block loads the ONNX-formatted ResNet CNN model for channel-based feature extraction and runs ONNX runtime inference to produce channel embeddings. On Jetson Orin, the CUDA execution provider accelerates inference.

Feature Quantization. This block uses the arithmetic mean of each channel embedding as a threshold to quantize channel features between 0 and 1 and outputs the initial secret message at Alice and Bob.

Reconciliation. Two GNU Radio blocks (Parity Generation and Reconciliation) implementing the reconciliation algorithm. We used a Python/C++ wrapper for a Reed-Solomon library. Alice generates parity bits using her quantized channel feature vector and an RS(255,128) encoder. Bob uses Alice's parity bits and his locally generated quantized channel feature vector with an RS(255,128) decoder and reconciles to Alice.

Privacy Amplification. This block implements SHA hashing of the reconciled secret messages. We allow the user to toggle between different variations of the SHA-3 algorithm and support different output key sizes such as 128, 256, and 512 bits.

Controller. A lightweight state machine coordinates: (1) Bob→Alice probe request, (2) Alice TX probe & Bob Rx, (3) Alice←Bob probe request, (4) Bob TX probe & Alice Rx, (5) Alice TX parity & Bob Rx, (6) Reconciliation success/failure acknowledgment. All transitions use PMT, PlutoSDR source/sink blocks for probe transmission and ZMQ Pub/Sub for information exchange. We developed an Rx gate GNU Radio block to control the flow of received samples through the PlutoSDR Source at Bob or Alice only when Alice or Bob is transmitting.

4. Experimental Setup

We collected IQ samples from pairs of SDR devices and a third SDR playing the role of a passive eavesdropper in an indoor lab setup at FAU CAAI. We collected probe exchanges between Alice and Bob while Eve listened to each of their exchanges. We trained the ResNet CNN model on an NVIDIA A100 GPU for channel-based feature extraction. We exported the CNN model to ONNX format and deployed and tested it on both Jetson Orin and Jetson Nano devices. Our experiments consider two trusted parties (Alice/Bob) at approximately $\sim$1 m apart and then

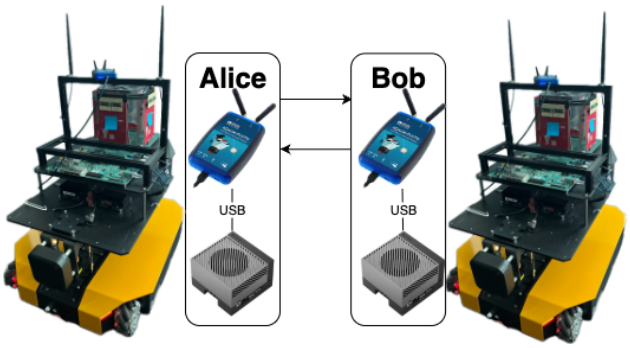


Figure 3. Alice and Bob experimental setup.

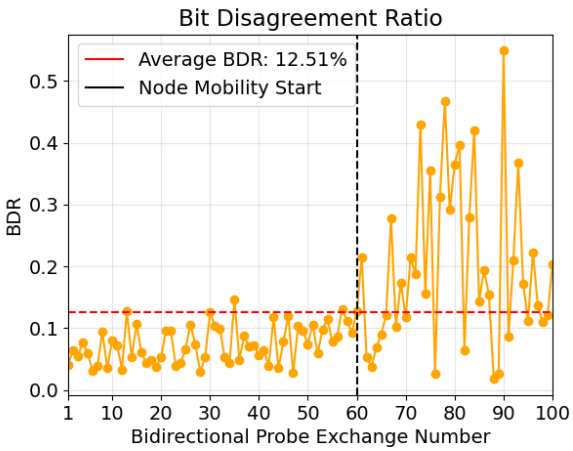


Figure 4. Key generation success rate.

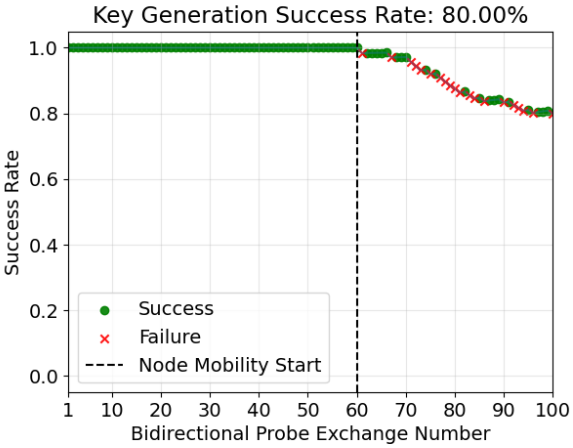


Figure 5. Bit disagreement ratios at every exchange.

moving one of the nodes at no more than a ∼5 m range. Each node is comprised of a Jetson Orin interfaced to an ADALM-Pluto SDR mounted on a robotic agent as depicted in Fig. 3. We built a Docker container on each of the robotic agents which contains all requirements for running the gr-PHYSEC blocks. The PlutoSDR at each agent was tuned to a carrier frequency of 2.485 GHz, and sample rate of 1 MSps. We also developed a key performance indicator (KPI) monitor app which triggers the nodes to initiate execution of the secret key generation protocol and collects and visualizes information from each stage such as IQ samples and spectrograms from channel probing, quantized channel feature vectors, and reconciliation success. We use the collected information to calculate bit disagreement rate (BDR), key generation success rate, and key generation time.

5. Results

5.1. Bit disagreement and reconciliation

Across 100 real-time key generations, BDR between Alice/Bob keys typically lies in the 2–15% range. Changes in orientation and motion between the two robotic agents increase BDR to $\sim$ 56% (the representative median is $\sim$ 12.51%) well within the RS correction capabilities as seen in Fig. 4. With RS$(255, 128)$, reconciliation is successful in most attempts, resulting in a key generation success rate of 80% as depicted in Fig. 5. The effects of changes in orientation and motion after epoch 60 are shown in both Fig. 4 and 5. Shorter RS codes (e.g., RS(191,128)) reduce parity exchange overhead but also success rate, as demonstrated in our prior physical layer security studies (Yıldırım et al., 2025; Pelekanakis et al., 2021; Sklivanitis et al., 2021).

5.2. Key randomness (NIST STS)

We evaluate the randomness of the generated keys using the NIST Statistical Test Suite (Rukhin et al., 2010) to verify whether the SHA-3 output is correlation-free and does not contain any exploitable patterns among the constituent bits. The NIST software is composed of fifteen distinct tests. In this work, we report on a representative subset commonly used in key generation studies: Monobit (Frequency), Frequency Within a Block, Runs, Longest Run of Ones in a Block, Discrete Fourier Transform (Spectral), Non–Overlapping Template Matching, Serial, Approximate Entropy, and Cumulative Sums.

Each test treats a reconciled, privacy–amplified cryptokey as one binary sequence. We average p-values across tests and calculate key pass rates over 100 independent generated keys. Results are summarized in Table 1.

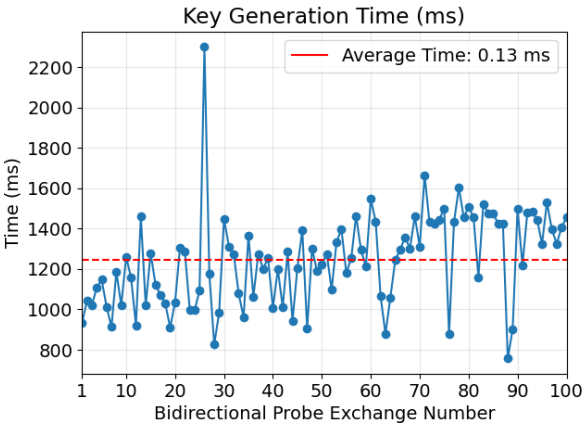


Figure 6. Key generation time in milliseconds.

Table 1. NIST statistical test suite scores.

Average test scores and generated keys		
Test Name	Average Test Score	Passing Percentage
Monobit	0.51	99%
Frequency Within Block	0.524	98%
Runs	0.508	100%
Longest Run Ones in a Block	0.211	84%
Discrete Fourier Transform	0.477	98%
Non Overlapping Template Matching	0.587	89%
Serial	0.527	99%
Approximate Entropy	0.517	100%
Cumulative Sums	0.508	99%

5.3. Latency and key generation rate

The average key generation time is ≈ 1299 ms in our embedded setup using a Jetson Orin as a compute platform. The estimated time include wait times of ≈400 ms that are put in place to implement bidirectional probe exchanges. Faster/optimized control links, parallelized probe exchange, or TensorRT/quantized CNNs could reduce latency (ONNX Runtime, 2025; O'Shea & Hoydis, 2017).

6. Reproducibility

The OOT blocks, example flowgraphs, and ONNX model are available at https://github.com/C2A2-at-Florida-Atlantic-University/gr-PHYSEC. We include installation notes for the gr-PHYSEC library and ONNX Runtime on Jetson platforms, example flowgraphs with control flow for running Alice and Bob SDR nodes as well as the KPI monitor which provides real-time information on probe exchanges and key performance indicators such as BDR, key generation success rate and key generation time. We developed a Docker container with the requirements to run on an ARM processor, and block-level parameter defaults. Blocks are self-contained and can be swapped (e.g., use different waveforms for channel probing).

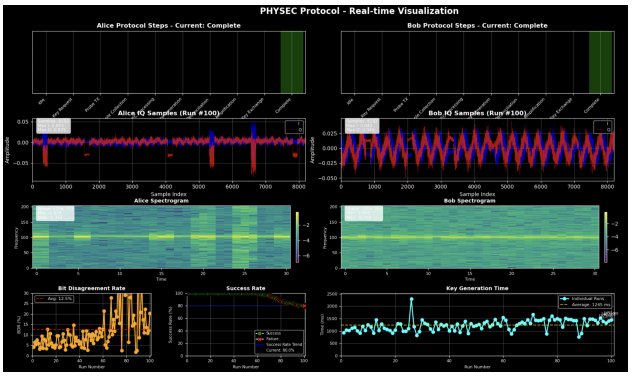


Figure 7. Key Performance Indicators (KPI) monitor.

7. Conclusion and Future Work

gr-PHYSEC delivers a practical, reproducible real-time physical layer key generation pipeline implemented in GNU Radio. A secret key generation protocol was implemented to generate a cryptokey between two authenticated nodes, Alice and Bob. Experimental bidirectional links were designed on low-cost COTS SDRs interfaced with edge AI inference platforms. The established links experienced a variety of channel conditions due to node mobility. Future work will focus on increasing key generation success rate and reducing key generation time through appropriate MAC-layer design as well as assessing protocol security that could be challenged by the relative position of Eve to Alice/Bob and the sophistication of Eve.

References

Allen, J. Short term spectral analysis, synthesis, and modification by discrete fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25 (3):235–238, 1977.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE*

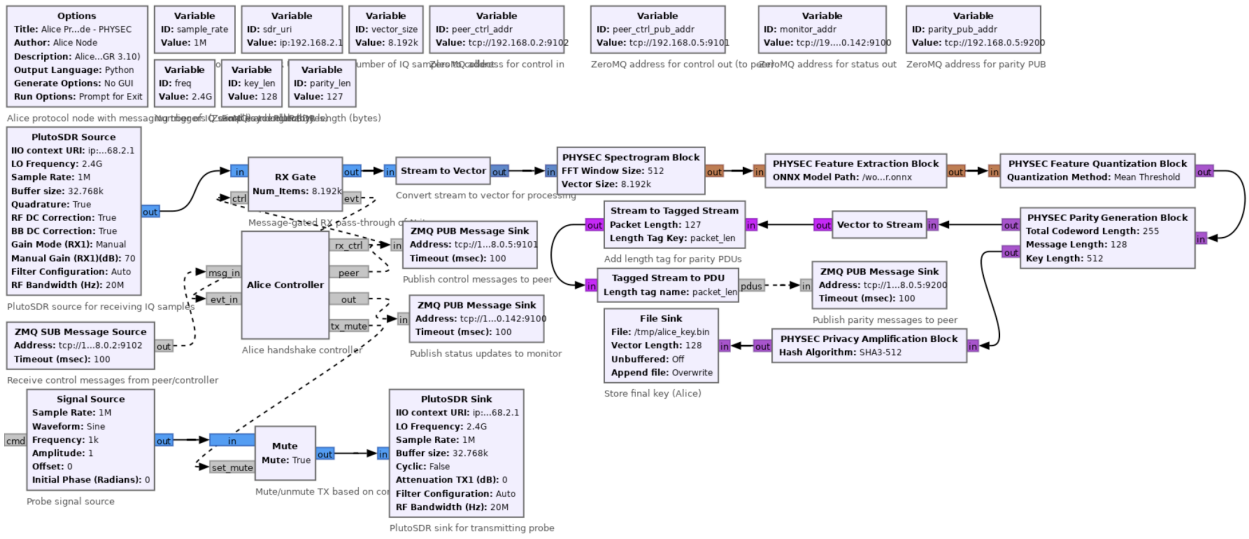


Figure 8. Alice's GNU Radio flowgraph.

Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016.

Hu, Yingying, Xu, Dongyang, and Zhang, Tiantian. Implementation and evaluation of physical layer key generation on sdr based lora platform. In *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, pp. 1–5, 2023. doi: 10.1109/VTC2023-Fall60731.2023.10333718.

Liu, H., Wang, Y., Yang, J., and Chen, Y. Fast and practical secret key extraction by exploiting channel response. In *Proceedings of IEEE INFOCOM*, pp. 3048–3056, 2013.

MacWilliams, F. J. and Sloane, N. J. A. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, 1977.

Mathur, S. et al. Radio telepathy: Extracting a secret key from an unauthenticated wireless channel. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp. 128–139, 2008.

ONNX Runtime. Onnx runtime, 2025. URL https://onnxruntime.ai/. Accessed 2025.

O'Shea, T. and Hoydis, J. An introduction to deep learning for the physical layer. *IEEE Transactions on Cognitive Communications and Networking*, 3(4):563–575, 2017.

Pelekanakis, Konstantinos, Yıldırım, Seçkin Anıl, Sklivanitis, Georgios, Petroccia, Roberto, Alves, João, and Pados, Dimitris. Physical layer security against an informed eavesdropper in underwater acoustic channels: Feature extraction and quantization. In *2021 Fifth Underwater Communications and Networking Conference (UComms)*, pp. 1–5, 2021. doi: 10.1109/UComms50339.2021.9598102.

Rukhin, A. et al. A statistical test suite for random and pseudorandom number generators for cryptographic applications. Technical Report SP 800-22 Rev. 1a, National Institute of Standards and Technology, 2010.

Ryland, Kevin, Lichtman, Marc, and Clancy, T. Implementation of two physical layer security techniques in an ota system. *Proceedings of the GNU Radio Conference*, 2(1):9, 2017. URL https://pubs.gnuradio.org/index.php/grcon/article/view/24.

Shen, G. et al. Radio frequency fingerprint identification for LoRa using deep learning. *IEEE Journal on Selected Areas in Communications*, 39(8):2604–2616, 2021.

Sklivanitis, George, Pelekanakis, Konstantinos, Yıldırım, Seçkin Anıl, Petroccia, Roberto, Alves, João, and Pados, Dimitris A. Physical layer security against an informed eavesdropper in underwater acoustic channels: Reconciliation and privacy amplification. In *2021 Fifth Underwater Communications and Networking Conference (UComms)*, pp. 1–5, 2021. doi: 10.1109/UComms50339.2021.9598159.

Wang, T. et al. Deep learning for wireless physical layer: Opportunities and challenges. *China Communications*, 14(11):92–111, 2017.

Yıldırım, Seçkin, Pelekanakis, Konstantinos, Sklivanitis, George, Pados, Dimitris A., Paglierani, Pietro, Petroccia, Roberto, Alves, João, Molfese, Francesco, and Cuomo, Francesca. Secret underwater acoustic key generation

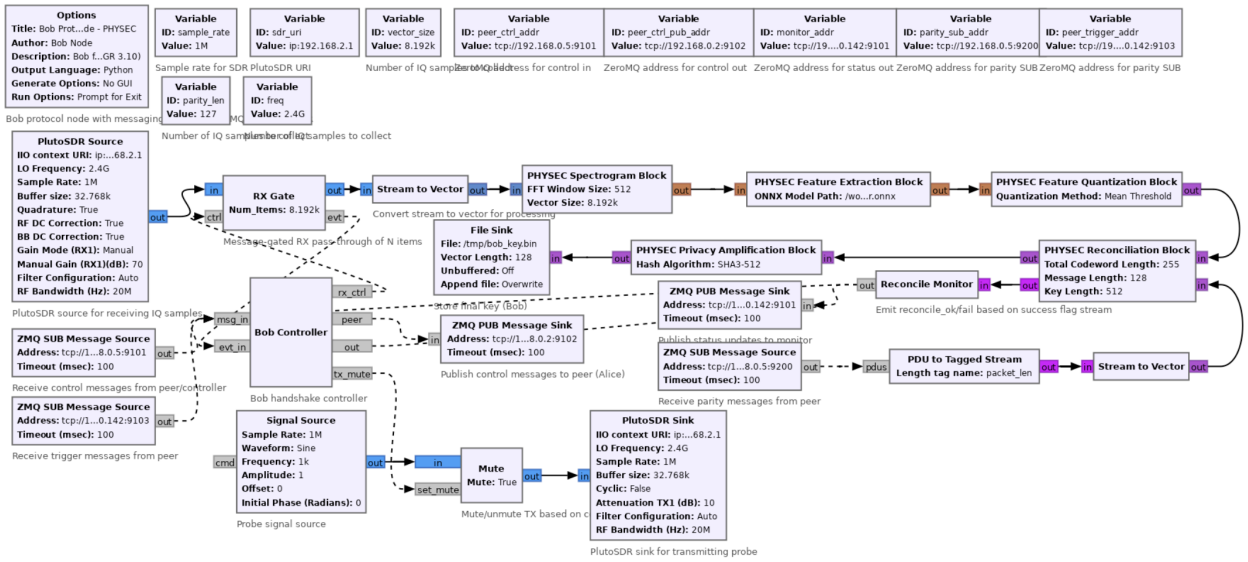


Figure 9. Bob's GNU Radio flowgraph.

challenged by eve's simulator. *IEEE Journal of Oceanic Engineering*, 50(2):489–506, 2025. doi: 10.1109/JOE.2023.3281978.

Zeng, K., Wu, D., Chan, A., and Mohapatra, P. Exploiting multiple antenna diversity for shared secret key generation in wireless networks. In *Proceedings of IEEE INFOCOM*, pp. 1–9, 2010.

Zhang, J., Duong, T. Q., Marshall, A., and Woods, R. Key generation from wireless channels: A review. *IEEE Access*, 4:614–626, 2016a.

Zhang, J. et al. Experimental study on key generation for physical layer security in wireless communications. *IEEE Access*, 4:4464–4477, 2016b.