An Encryption System for Securing Physical Signals

Yisroel Mirsky[1,2(✉)], Benjamin Fedidat[3], and Yoram Haddad[3]

[1] Georgia Institute of Technology, Atlanta, GA, USA
[2] Ben-Gurion University, Beer Sheva, Israel
yisroel@post.bgu.ac.il
[3] Jerusalem College of Technology, Jerusalem, Israel
ben@fedidat.com, haddad@jct.ac.il

Abstract. Secure communication is a necessity. However, encryption is commonly only applied to the upper layers of the protocol stack. This exposes network information to eavesdroppers, including the channel's type, data rate, protocol, and routing information. This may be solved by encrypting the physical layer, thereby securing all subsequent layers. In order for this method to be practical, the encryption must be quick, preserve bandwidth, and must also deal with the issues of noise mitigation and synchronization.

In this paper, we present the Vernam Physical Signal Cipher (VPSC): a novel cipher which can encrypt the harmonic composition of any analog waveform. The VPSC accomplishes this by applying a modified Vernam cipher to the signal's frequency magnitudes and phases. This approach is fast and preserves the signal's bandwidth. In the paper, we offer methods for noise mitigation and synchronization, and evaluate the VPSC over a noisy wireless channel with multi-path propagation interference.

Keywords: Physical channel security · Vernam Cipher · Harmonic encryption · FFT · Signal encryption · Waveforms

1 Introduction

Knowledge is power. It is in the interest of two communicating parties to secure their communication channel to the degree that no information about the channel or the communication is revealed. Today it is common practice to encrypt the payloads of higher level protocols in the communication protocol stack (*Layer 4* and above in the OSI model). This is analogous to encrypting the content of a letter but not the envelope itself. Doing so allows onlookers (eavesdroppers) to see the frame's header information in plaintext and even modify it. This is an even more serious problem for radio transmissions over public domains [22]. For

Y. Mirsky—Part of this author's work was done in the Jerusalem College of Technology.

N. Park et al. (Eds.): SecureComm 2020, LNICST 335, pp. 211–232, 2020.
https://doi.org/10.1007/978-3-030-63086-7_13

instance, by targeting specific bits, an attacker is able to perform energy-efficient jamming [14].

Solutions to this problem have been investigated thoroughly [3,18,27]. A common solution is to encrypt the data-link layer (*Layer 2*) before it is passed on to the physical layer (*Layer 1*). MACSec [20] is an example of such a protocol for multi-hop wired networks while the most common wireless encryption protocols are WEP and WPA/2. However, encrypting the *Layer 2* bit-stream does not provide secrecy for all information obtainable from the physical channel's characteristics. Examples of this information include traffic statistics, data rates, number of physical channels, service priorities, data size, data packet frequency, baud rate, modulation, and channel bandwidth. Knowledge of this information can be used to infer the transmission equipment, message importance, channel content, and channel capacity. This information leakage is analogous to writing letters in code and mailing them to a friend. One who sees these letters cannot explicitly determine the content, however the shape (bandwidth), address (protocol) and transmission frequency (bit-rate) can reveal significant information. Another advantage to securing the physical signal is that doing so protects the channel from all attacks that require observation of the bits. For example, protocol manipulation, timing inference, and replay attacks.

Therefore, in order to make a communication channel completely secure, the channel in its entirety should be protected, like a curtain over the entire operation. By extension, it should also be impossible to determine whether the intercepted signal was originally of a digital or analog origin. This level of security can only be achieved by acting at *layer 1* of the OSI protocol stack. For this reason, research on the topic of physical layer security (*Layer 1*) has gained attention over the years [3].

In this paper, we propose an application of the Vernam Cipher to analog signals, which we call the Vernam Physical Signal Cipher (VPSC). The VPSC is unique because it encrypts waveform signals on the frequency domain while achieving a high degree of secrecy on *Layer 1*.

There are several notable advantages to working on the frequency domain:

1. **Complete Information Privacy:** By encrypting the raw signal itself, no information about the channel is exposed. Regardless of waveform, the encrypted signal appears as white noise.
2. **Bandwidth Preservation:** This aspect is particularly desirable for radio applications where spectrum is a commodity. This is in contrast to performing modulo-based encryption on samples from the temporal plane, since doing so adds energy to all frequencies in the spectrum. In order to preserve the original signal's bandwidth, transformation in the frequency domain is necessary.
3. **Selective Band Encryption:** This process is similar to a band-pass filter since an entire signal's spectrum can be presented to the VPSC, but only selected frequency bands will be encrypted (regardless of the bands' content).
4. **Hardware Parallelization:** The signal's spectrum can be split and then encrypted in parallel by independent processors in real-time. This is useful when dealing with very large bands and weak processors. This modularity

makes the technology highly scalable to the consumer's needs. When operating directly on the temporal plane, this type of parallelization cannot be achieved when targeting specific frequency bands.

Although some of these advantages are present in current wireless security channels, the advantages altogether are unavailable in any single one [18,22,27]. The aim of the VPSC is to provide a secure connection (one that does not leak any information, even about the channel itself) between two communicating parties over a single physical link – such as a radio channel or an optical trunk line. The VPSC can also be applied to a multi-hop network if each link is protected separately, and the routing nodes are considered trusted.
Altogether, this paper has three main contributions:

1. **A Generic Physical Layer Cipher:** A method for applying the One-Time Pad [21]/Vernam Cipher [24] to the frequency domain, enabling all of the advantages listed above.
2. **Noise Mitigation for Modulo Operation on Waveforms:** We propose two noise mitigation techniques which enable modulo-based signal encryption (on both the time and frequency domains), and a method which combines the two into one. These noise mitigation techniques are necessary because modulo operations on analog signals are extremely sensitive to the presence of noise.
3. **Simulations, Source Code, and Prototyping:** We evaluate the VPSC's practicality by simulating a realistic wireless channel with distortions, interference, and noise. We also provide the Python source code for researchers to reproduce our work.[1] Finally, we demonstrate the technology by implementing the VPSC over two Arduino Duo development boards.

2 Related Work

In 1919, Gilbert Vernam patented a XOR-based cipher known as the Vernam Cipher [24]. This cipher works by applying the XOR operation between a message and a secret pseudo-random key. In 1949, Claude Shannon published a historical paper [21] in which communication secrecy was studied from the perspective of information theory. He proved the theoretical significance of the Vernam Cipher and proposed the one-time pad (OTP), also known as the Shannon Cipher System (SCS), a cipher capable of perfect secrecy. The OTP is essentially a Vernam Cipher which uses a truly random key.

Later in 1975, Wyner wrote his seminal paper that describes a degraded wiretap channel and provides information-theoretic concepts needed for the domain [25]. Loosely speaking, a wiretap channel (WTC) is one where the sender (Alice) transmits a signal to the legitimate receiver (Bob) while an eavesdropper (Eve) intercepts it. However, the signal Eve intercepts is noisier than Bob's, allowing Bob to obtain information that Eve cannot.

[1] The Python source code to the VPSC can be found online: https://github.com/ymirsky/VPSC-py.

We can relate the SCS to the WTC because both can be directly applied to quantized signal samples. However, there is an inherent difference between the SCS and the WTC. The SCS's secrecy is solely based on information theory, while the WTC's secrecy is based on exploiting the physical traits of communication media [5,25]. In [26], the authors find the secrecy capacity of a shared-key WTC in the presence of noise. Later, the authors in [12] generalized [26] by considering any channel (not necessarily noisy). It can be seen in these works that a secret-key WTC without the presence of noise and with maximum secrecy is essentially an SCS. Therefore, in our work, we focus on the SCS since its theory forms the basis for the VPSC.

In [13], the author proposes two methods for encrypting a physical signal: amplitude log masking (ALM) and sample-wise RSA encryption. In ALM, each signal sample in the time-plane is encrypted by taking the logarithm of the sample multiplied by a random value (a key). However, the ALM method does not provide a high degree of secrecy since ALM simply masks (obfuscates) the samples with a key, and attacks such as correlation analysis [10] can be used to reveal the masked message. Moreover, ALM is very sensitive to noise since errors are exponentially multiplied during the decryption process. This makes ALM impractical to use in noisy channels.

In the RSA method, the RSA [19] cipher is applied to each sample from the time-plane. The RSA method requires that the samples be quantized to discrete values. This is necessary in order to perform the power operation of RSA without float-point overflows. As a result, the RSA method is highly susceptible to even the slightest amount of noise. This is because RSA has a non-linear relationship between the cipher-text and the plain-text. As a consequence, every single rounding error in the cryptogram results in a completely different deciphered value. This behavior is similar to how the output of a hash map is sensitive to changes in its key. This means that the RSA method is not a practical solution for real-world channels.

Both the ALM and RSA methods use a significant amount of energy over the entire spectrum. These are undesirable side effects, especially for wireless channels. The VPSC, on the other hand, uses the same amount of bandwidth as the original signal, and a similar amount of energy as well. Furthermore, the VPSC is much more robust to noise and interference, since the modulo operation of the SCS maps noise close to the original sample's value (discussed further in Sect. 8).

In [16], the authors propose frequency component scrambling (FCS) via a Fast Fourier Transform (FFT) to protect audio channels. Their method is to scramble the frequency components (f.c.) of the given signal to obfuscate its contents. However, scrambling does not provide a high level of secrecy. Regardless of the number of f.c.s, it is possible to descramble the signal by analyzing the correlation of the f.c. magnitudes, similarly to what was done in [10]. For example, if a 16-QAM modulation is applied to the carrier frequency f_c, FCS is used to encrypt the band surrounding f_c, then FCS would simply move the contents of f_c to a neighboring bin (see Fig. 1). This is similar to frequency hopping

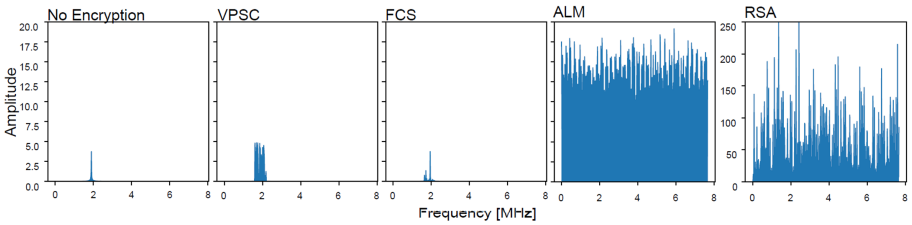

Fig. 1. The spectrum of a 16-QAM channel, with a carrier wave of 1.9 MHz, encrypted by the various methods.

except applied to a much smaller band, and with only one hop. Therefore, FCS is only applicable to the prevention of casual eavesdroppers.

Figure 1 shows the spectrum of an encrypted 16-QAM signal carried on 1.9 MHz wave, using each of the methods. It can be seen that the methods either do not sufficiently secure the channel (FCS), or use the entire spectrum with a large amount of energy (ALM and RSA).

Therefore, to the best of our knowledge, the VPSC is the first physical layer encryption system capable of perfect secrecy that operates directly on the frequency domain, and is also robust to noise.

3 Cryptographic Model

In this Section, we introduce the notations which will be used throughout this paper. We also apply the standard cryptographic model to the waveform message space.

In this paper, we assume that we are dealing with discretely sampled and quantized real signals. Let there be Q discrete levels of quantization such that the highest level is Q_U and the lowest is Q_L. Let $S \subseteq \mathbb{Z}^N$ be the collection of all possible signal segments to be encrypted from some particular channel T, where $N = 2^k, k \in \mathbb{N}$ and $\boldsymbol{s} \in S$ has the form

$$\boldsymbol{s} = \begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix} \tag{1}$$

where $Q_L \leq v_i \leq Q_U$ for every i. In other words, $\boldsymbol{s}$ is a frame of N quantized samples, taken from the time domain, which we want to encrypt.

Let f_s be the sample rate of the system such that $f_s \geq 2B$, where B is the essential bandwidth of the selected signals from S (Nyquist rate). Let the message space M (the collection of all possible plaintext messages) of the cryptosystem be defined as the collection of all discrete Fourier transforms (DFT) of the vectors in S, in polar form, such that

$$M = \left\{ (\boldsymbol{m}_m, \boldsymbol{m}_a) \middle| \begin{matrix} \boldsymbol{m}_m = |DFT[\boldsymbol{s}]| \\ \boldsymbol{m}_a \angle DFT[\boldsymbol{s}] \end{matrix} \right\} \tag{2}$$

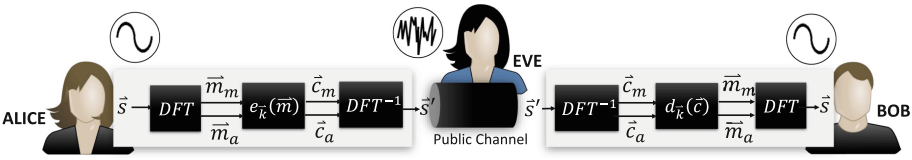

Fig. 2. The cryptosystem model for analog signals. Alice sends a waveform signal to Bob, and Eve's interception provides her with no information about it.

It is helpful to view the message $\boldsymbol{m} \in M$ as the polar form of the DFT of N consecutive samples of a real-time signal segment found in S. In other words, $\boldsymbol{m}_m$ represents the frequency magnitudes and $\boldsymbol{m}_a$ denotes the frequency angles (phases) of the signal segment $\boldsymbol{s}$.

Let ϕ be a scalar parameter which defines the maximum frequency magnitude of the cryptosystem. It is restricted to the inequality

$$\phi \geq max(\boldsymbol{m}_m[i] + \varepsilon), \forall i, \boldsymbol{m}_m \tag{3}$$

where ε is a small value. Let the key space K (the collection of all possible keys) of the cryptosystem be defined as a collection of all possible tuples in the form

$$\boldsymbol{k} = (\boldsymbol{k}_m, \boldsymbol{k}_a) \tag{4}$$

where $\boldsymbol{k}_m$ and $\boldsymbol{k}_a$ are random N-length vectors which are used to encrypt magnitudes and angles respectively. Since we are dealing with real signals, $\boldsymbol{k}_m$ and $\boldsymbol{k}_a$ must be structured to conform to the DFT output from real signals.

Specifically, $\boldsymbol{k}_m$ has the structure

$$\boldsymbol{k}_m = concatenate\left(0, \boldsymbol{v}, \texttt{mirror}\left(\boldsymbol{v}\left[2 : \frac{N}{2}\right]\right)\right) \tag{5a}$$

where $\boldsymbol{v}$ is a $^N/_2$ length vector of random values on the range $[0, \phi)$, `refl` is the mirror rearrangement operation on the values of some vector, and the symbol ":" indicates a range of indexes. Similarly, $\boldsymbol{k}_a$ has the structure

$$\boldsymbol{k}_a = concatenate\left(\boldsymbol{a}, -\texttt{refl}(\boldsymbol{a})\right) \tag{5b}$$

where $\boldsymbol{a}$ is also a $^N/_2$ length vector of random values, but on the range $[-\pi, \pi)$.

Let the key space K (the collection of all possible keys) of the cryptosystem be defined as the collection of all possible $\boldsymbol{k}$.

The cryptogram space of the system C is equivalent to the collection of all possible real signals found in M. This is necessary in order to obtain perfect secrecy, since it must be possible to map any cryptogram $\boldsymbol{c} \in C$ back to any message $\boldsymbol{m} \in M$ [21].

Let the inverse-DFT (DFT^{-1}) of the cryptogram $\boldsymbol{c}$ be referred to as $\boldsymbol{s}'$, such that

$$DFT^{-1}(\boldsymbol{c}) = \boldsymbol{s}' \tag{6}$$

Let the general encryption function be defined as

$$e_{\boldsymbol{k}}(\boldsymbol{m}) = \boldsymbol{c} \tag{7a}$$

and the general decryption function be defined as

$$d_{\boldsymbol{k}}(\boldsymbol{c}) = \boldsymbol{m} \tag{7b}$$

where the key $\boldsymbol{k} \in K$ is used to encrypt the message $\boldsymbol{m} \in M$ and decrypt the ciphertext $\boldsymbol{c} \in C$.

Now that some notation has been defined, we can present the cryptographic model used in this paper, as depicted in Fig. 2. Consider a case in which Alice wants to transmit $\boldsymbol{s}$ (a segment of some analog signal) securely to Bob so that Eve cannot obtain any information about $\boldsymbol{s}$ as it travels across the public medium. First, Alice obtains the tuple $\boldsymbol{m} = (\boldsymbol{m}_m, \boldsymbol{m}_a)$ by converting the DFT of $\boldsymbol{s}$ into polar form. Next, Alice encrypts the frequency components ($\boldsymbol{m}$) with (7a) by performing $e_{\boldsymbol{k}}(\boldsymbol{m}) = \boldsymbol{c}$. Finally, Alice performs a DFT^{-1} on the cryptogram $\boldsymbol{c}$ and transmits the result $\boldsymbol{s}'$ over the public medium towards Bob.

Once Bob has received $\boldsymbol{s}'$ he can obtain the original $\boldsymbol{s}$ from it by performing the same steps which Alice performed, while using the decryption function (7b) instead.

This process is repeated continuously in real-time for each set of N samples that Alice wishes to send. However, each key $\boldsymbol{k}$ is selected at random from K (key generation is further described in the next Section).

4 The Vernam Physical Signal Cipher

In this Section, we define the VPSC by detailing its implementation of the encryption and decryption functions (7a, 7b). We also introduce two methods of noise mitigation which are essential for the VPSC to work in the real world. Afterwards, we review possible key-sharing options.

Let the VPSC encryption function be defined as

$$e_{\boldsymbol{k}}(\boldsymbol{m}) = \begin{Bmatrix} e_{\boldsymbol{k}_m}(\boldsymbol{m}_m)+\lambda \\ e_{\boldsymbol{k}_a}(\boldsymbol{m}_a) \end{Bmatrix} = \begin{Bmatrix} ((\boldsymbol{m}_m+\boldsymbol{k}_m) \bmod \phi)+\lambda \\ \boldsymbol{m}_a+\boldsymbol{k}_a \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{c}_m+\lambda \\ \boldsymbol{c}_a \end{Bmatrix} = \boldsymbol{c} \tag{8a}$$

and let the VPSC decryption function be defined as

$$d_{\boldsymbol{k}}(\boldsymbol{c}) = \begin{Bmatrix} d_{\boldsymbol{k}_m}(\boldsymbol{c}_m)-\lambda \\ d_{\boldsymbol{k}_a}(\boldsymbol{c}_a) \end{Bmatrix} = \begin{Bmatrix} ((\boldsymbol{c}_m-\boldsymbol{k}_m) \bmod \phi)-\lambda \\ \boldsymbol{c}_a-\boldsymbol{k}_a \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{m}_m-\lambda \\ \boldsymbol{m}_a \end{Bmatrix} = \boldsymbol{m} \tag{8b}$$

where mod is the element-wise modulo operation, $\boldsymbol{k}$ is a purely random vector selected from K, and λ is a required amplification of the encrypted signal. The parameter λ is a constant defined by the user such that $\lambda > 0$. The purpose of λ is to ensure that the phase of component n will not be lost, in the chance that $\boldsymbol{c}_m[n] \approx 0$. This can legitimately occur at random, based on $\boldsymbol{k}_m$. We note that λ does not affect the secrecy of $\boldsymbol{c}$ because we are simply amplifying the final signal.

4.1 Noise Mitigation

Since the VPSC is a physical signal cipher, it must operate according to physical constraints. One of those is ϕ; the maximum frequency magnitude of the cryptosystem. This parameter must be larger than the largest possible frequency magnitude in M, by a small amount ε, as described in (3). Setting ϕ to a value less than the largest magnitude will result in a loss of information due to the modulo operation.

The functions (8a, 8b) can provide a high level of security since they are essentially an OTP (discussed later in Sect. 5). However, their implementation in reality (as-is) does not function. This is because every communication medium adds some noise to the channel, whether it is natural noise or some other signal interference. Therefore, under normal circumstances, some energy always gets added or subtracted from some of the frequency magnitudes in $\boldsymbol{c}_m$. This incurs an undesirable effect in the decryption process.

Depending on the amount of energy, the subtraction and then modulo of the cryptogram in (8b) can send values in $\boldsymbol{m}_m$ that were close to 0 or ϕ to the opposite extreme.

To illustrate this issue we can track the usage of the VPSC over some noisy channel. Let's say that $\boldsymbol{m}_m[n]$ is the n$^{\text{th}}$ frequency magnitude from the original signal segment $\boldsymbol{s}$, and that $\boldsymbol{m}_m[n] = \phi - \varepsilon$, where ε is some relatively small number. Suppose that the encryption key to be used on the magnitude $\boldsymbol{m}_m[n]$ is $\boldsymbol{k}_m[n] = \alpha$, where $0 \leq \alpha < \phi$. Encrypting $\boldsymbol{m}_m$ with (8a) results in:

$$\boldsymbol{c}_m[n] = (\boldsymbol{m}_m[n] + \boldsymbol{k}_m[n]) \bmod \phi = (\phi - \varepsilon + \alpha) \bmod \phi \tag{9a}$$

Now $\boldsymbol{c}_m$ is converted into $\boldsymbol{s}'$ by (6) and transmitted over the communication medium. Assume that by doing so, the magnitude $\boldsymbol{c}_m[n]$ (of $\boldsymbol{s}'$) receives some additional energy γ from noise in the channel, where $\gamma > \varepsilon$. The result is that Bob now receives a noisy cryptogram,

$$\boldsymbol{c}_m[n]^* = (\phi - \varepsilon + \alpha) \bmod \phi + \gamma \tag{9b}$$

and Bob cannot analytically determine γ since the original message magnitude $\boldsymbol{m}_m[n] = \phi - \varepsilon$ is unknown to him. When Bob tries to decrypt (9b) using (8b), assuming $\gamma - \varepsilon < \phi$ the following will occur:

$$d_{\boldsymbol{k}_m[n]}(\boldsymbol{c}_m[n]^*) = (\boldsymbol{c}_m[n]^* - \boldsymbol{k}_m[n]) \bmod \phi \tag{9c}$$

If $\alpha < \varepsilon$ then (9c) evaluates to

$$(\phi - \varepsilon + \alpha + \gamma - \alpha) \bmod \phi = (\phi - \varepsilon + \gamma) \bmod \phi = \gamma - \varepsilon \text{ since } \gamma > \varepsilon \tag{9d}$$

If $\alpha \geq \varepsilon$ then (9c) evaluates to

$$(\alpha - \varepsilon + \gamma - \alpha) \bmod \phi = (\gamma - \varepsilon) \bmod \phi = \gamma - \varepsilon. \tag{9e}$$

In both cases (9d, 9e), Bob will interpret $\boldsymbol{s}$'s n$^{\text{th}}$ frequency magnitude to be a near-zero value, as opposed to the correct near maximum value (ϕ). Similarly,

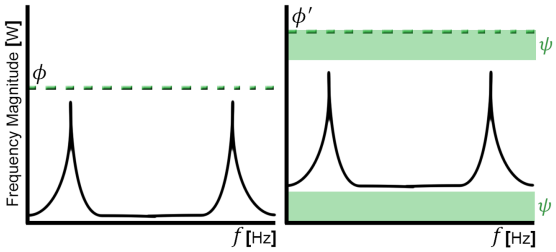

Fig. 3. A sine wave signal undergoing the PR technique. The image on the right shows the new signal after the procedure where the shaded areas provide a modulo-error "buffer zone" with a width of ψ watts each.

the same issue can be shown for near-zero values being interpreted as maximum values as well.

These unavoidable errors add a tremendous amount of noise to the decrypted signal. Therefore, since a small amount of noise energy γ can cause a large signal to noise ratio (SNR), it is impractical to implement the VPSC as-is by simply using the encryption and decryption functions (8a, 8b) without any noise mitigation. Therefore, we propose two methods of noise mitigation for the VPSC: preemptive-rise and statistical-floor.

Preemptive-Rise. The preemptive-rise (PR) technique is implemented both in the encrypter (transmitter) and decrypter (receiver). The idea is to make a buffer zone above and below the original signal's range of frequency magnitudes. This ensures that the addition of random noise will not cause any of the magnitudes to fall out of bounds during the subtraction step of (8b) as depicted in Fig. 3. This is not a conventional signal boost since non-relevant frequencies within the encrypted band will be boosted as well.

Let ψ be the width of each buffer zone in watts where $\psi \equiv u \times \sigma_0$ such that $u \in \mathbb{N}$ and σ_0 is the standard deviation of the channel's noise energy.

In order to implement PR, the encrypter and decrypter must use a larger ϕ than previously required due to the larger range of magnitude values. This larger version ϕ' can be defined as

$$\phi' = \phi + 2\psi \tag{10}$$

where ϕ is determined from (3). Furthermore, the range from which the magnitude keys can be selected (5a) must be changed to $[0, \phi')$.

The implementation of PR is equivalent to modifying the magnitude encryption function in (8a) to

$$e_{\boldsymbol{k}_m}(\boldsymbol{m}_m) = (\boldsymbol{m}_m + \boldsymbol{k}_m + \psi) \bmod \phi' \tag{11}$$

and magnitude decryption function in (8b) to

$$d_{\boldsymbol{k}_m}(\boldsymbol{c}_m) = (\boldsymbol{c}_m - \boldsymbol{k}_m) \bmod \phi' - \psi \tag{12}$$

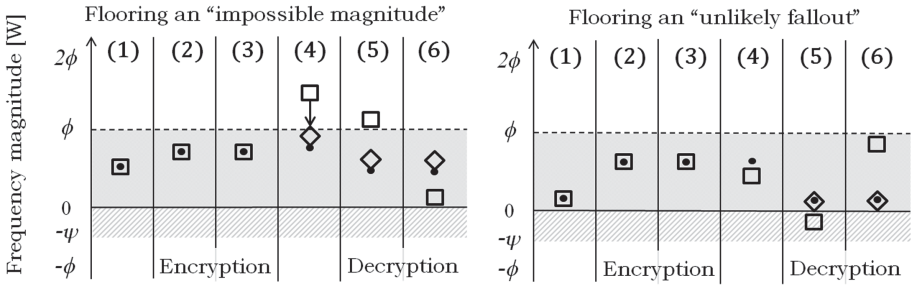

Fig. 4. A single frequency magnitude undergoing SF. Each step of the VPSC encryption/decryption is shown for when the system has no noise (filled circle), noise but no SF (square frame), and noise with SF (diamond frame). Steps are **(1)** original value, **(2)** key added, **(3)** modulo step, **(4)** transmission over channel, **(5)** key removed, and **(6)** modulo step.

Although PR can completely eliminate the noise distortions, its cryptogram (12) requires a greater transmission power than the original cryptogram (8a) due to the power change in (10).

Statistical-Floor. Unlike the PR technique, statistical-floor (SF) is implemented in the decrypter alone. The idea is to try and correct those values which have erroneously been shifted over the boundaries by the noise energy. The method tries to clean the signal by correcting erroneous fallouts before and after the subtraction step in (8b).

There are two cases which we consider erroneous: impossible magnitudes and unlikely fallouts. When there is no added noise to the signal, it is impossible to receive an encrypted magnitude above a certain value. Therefore, when we receive these "impossible magnitudes" the only conclusion we can have is that they were affected by some positive noise. More specifically, when there is no noise, the largest frequency magnitude possible is $\phi - \varepsilon$.[2] When there is noise, it is possible to receive a magnitude above ϕ. Therefore, we can conclude that any received magnitudes greater than or equal to ϕ should be floored to $\phi - \varepsilon$ before the subtraction step (8b). An illustration of this technique can be found at the top of Fig. 4.

This flooring procedure can be done without any prior knowledge about the original signal. However, after the subtraction step, some values end up just above or below 0. Without knowing the original signal, it is impossible to determine if the values had initially been just above 0 or had been altered as a result of noise. As discussed earlier, these values could add a large amount of noise to the signal after the modulo step. Fortunately, if we have some statistical information about the original signal, then we can attempt to correct these values (unlikely fallouts).

[2] The largest magnitude of the system is $\phi - \varepsilon$ and not *phi*, similar to how in *nmodm*, the largest n can be is $m - 1$.

For instance, let us assume that it is known that the original signal has most of its magnitude values close to 0 (like the signal in Fig. 3). In this case, after the subtraction step, we will assume that all values in the range $[-\psi, 0)$ were supposed to be just above 0 but were shifted down by noise. To correct them, we will floor them to 0. The same idea can be applied if we have prior knowledge that the original signal is mostly made up of large magnitudes. An illustration of this technique can be found at the bottom of Fig. 4. This figure shows that the error is minimized by SF in both cases. Furthermore, in the case of "impossible magnitudes", a potentially high level of noise can be mitigated.

In Algorithm 1 the pseudo-code for the SF magnitude decrypter modified from (8b), where ε is a very small number, and we assume that the original signal has mostly small magnitudes.

Algorithm 1. Pseudo-code for the magnitude decryption function using the SF technique, under the assumption that the original signal has mostly small frequency magnitudes.

function STATISTICAL-FLOOR($\boldsymbol{m}_m$, $\boldsymbol{k}_m$)
  for $i \leftarrow 1$ to N **do**
    if $\boldsymbol{m}_m[i] \geq \phi$ **then** ▷ impossible magnitude
      $\boldsymbol{m}_m[i] \leftarrow \phi - \varepsilon$
    end if
    $\boldsymbol{m}_m[i] \leftarrow \boldsymbol{c}_m[i] - \boldsymbol{k}_m[i]$
    if $-\psi \leq \boldsymbol{m}_m[i] < 0$ **then** ▷ unlikely fallout
      $\boldsymbol{m}_m[i] \leftarrow 0$
    end if
    $\boldsymbol{m}_m[i] \leftarrow (\boldsymbol{m}_m[i]) \bmod \phi$
  end for
end function

Algorithm 2. Pseudo-code for the VPSC's magnitude encryption, using the combined noise mitigation technique.

function $e_{\boldsymbol{k}_m}(\boldsymbol{m}_m)$
  $\boldsymbol{c}_m \leftarrow \boldsymbol{m}_m$
  for $i \leftarrow 1$ to N **do**
    $\boldsymbol{c}_m[i] \leftarrow \boldsymbol{c}_m[i] + \lambda$ ▷ add energy buffer
    $\boldsymbol{c}_m[i] \leftarrow \bmod(\boldsymbol{m}_m[i] + \boldsymbol{k}_m[i], \phi + 2\lambda)$ ▷ encrypt
    $\boldsymbol{c}_m[i] \leftarrow \boldsymbol{c}_m[i] + \lambda$ ▷ add carrier energy
  end for
end function

Algorithm 3. Pseudo-code for the VPSC's magnitude decryption, using the combined noise mitigation technique.

```
function d_{k_m}(c_m)
    m_m ← c_m
    for i ← 1 to N do
        if c_m[i] > φ + 3λ then                ▷ remove impossible magnitudes
            c_m[i] ← φ + 3λ
        end if
        c_m[i] ← c_m[i] − λ                    ▷ remove carrier energy
        if c_m[i] < 0 then
            c_m[i] ← 0
        end if
        m_m[i] ← mod(c_m[i] − k_m[i], φ + 2λ)  ▷ decrypt
        m_m[i] ← m_m[i] − λ                    ▷ remove energy buffer
    end for
end function
```

Although SF does not require more power to transmit a cryptogram (unlike PR), it sometimes incorrectly floors values that happen to be below 0 that should not have been modified. Since this error is unavoidable when using this technique, it is impossible for SF to completely eliminate the modulo noise distortions without adding some of its own.

Combined Method. It is clear that the PR and SF techniques have their advantages and disadvantages, as mentioned above. To obtain the *best of both worlds*, one may use a combination of both PR and SF techniques. Such a combination can eliminate almost all distortions while using less transmission power than PR to achieve the same noise reduction. The pseudo-code for performing encryption and decryption with the combined method can be found in Algorithms 2 and 3.

4.2 Key Sharing

Since the VPSC is a specific case of the OTP, the length of the VPSC's key must equal the length of the streamed message. Sharing this key as a prior secret between two parties is impractical. In this type of situation it is common for both parties to agree upon a secret seed to initialize a key-stream generator. This makes the shared key finite as opposed to virtually infinite in length. Another consideration is how two communicating parties with only public channels can share a secret key (or seed) before any secure channel has been established. This subject has been well researched [8] and there are several common solutions. One solution is to form a hybrid cryptosystem using public-key cryptography [23]. In hybrid cryptosystems, a symmetric-key system (such as the VPSC) is initiated with an asymmetric key exchange.

5 Cryptanalysis

The VPSC is unconditionally secure (unbreakable) when using truly random keys. Claude Shannon proved that if truly random numbers are used to generate this cipher's encryption keys, it then becomes what is known as a One-Time-Pad (OTP), which is unconditionally secure [21].

The typical OTP operates by performing bitwise XOR operations on two binary vectors of equal length: the message and the key. The cryptographic behavior of the XOR operation, performed on each bit in the vector, can be applied to each element in an n-ary vector as well. This is because the XOR operation can be viewed as a modulo-2 operation. For instance, if $a, b \in \{0, 1\}$ then the XOR operation $a \oplus b$ is equivalent to performing $(a+b) \bmod 2$. Similarly, with an n-ary vector, if $c, d \in \{0, \ldots, n-1\}$ then the same cryptographic behavior is observable from $(a + b) \bmod n$.

This means that the OTP cipher can be applied to any system which uses n-ary vectors such as $\boldsymbol{m}_m, \boldsymbol{k}_m, \boldsymbol{c}_m$ and $\boldsymbol{m}_a, \boldsymbol{k}_a, \boldsymbol{c}_a$ from our cryptographic model described in Sect. 3. Therefore, the VPSC can be viewed as an extension of the OTP, and if the selection of $\boldsymbol{k} = (\boldsymbol{k}_m, \boldsymbol{k}_a) \in K$ is purely random then the VPSC is unconditionally secure. The complete proof can be found in appendix 11.

Since no amount of cryptograms $\boldsymbol{c}[i]$ may provide any information about the original plaintext $\boldsymbol{m}[i]$, an encrypted signal $\boldsymbol{s}'$ has the appearance of random noise (completely random samples with zero correlation).

It is also important to note that public knowledge of ϕ or ϕ' does not compromise the system; rather it is the secrecy of the key which protects the message (see the proof in appendix 11). Therefore, knowledge of the setting of ϕ may not compromise the system. This conforms with Kerckhoffs's principle since ϕ is part the cryptosystem and does not belong to the secret key.

Finally, the strength of symmetric stream ciphers depends on random number generator [15]. Therefore, to secure the VPSC, one must use a cryptographically secure pseudo random number generator (CSPRNG) [17] such as elliptic curve generators [2] or the use of integer factorization like the Blum-Blum-Shub number generator [4].

6 Signal Synchronization

In this Section, we propose a direct analytical method of synchronizing a VPSC decrypter to an encrypter at any arbitrary point in its key sequence. We will assume that the VPSC has been implemented with a cipher in counter (CTR) mode, such as AES-CTR [6]. This is usually done in the manner shown in [9]. The counter mode, while being secure and fully parallelizable (which is a performance advantage) [7], also offers the ability to access any value in the key stream independently from previous ones.

In order to initialize their cryptographic systems, both the encrypter and decrypter must use the same secret configuration. Let D be the collection of all possible initial configurations for a system such that

$$D = (sc, st, g) | sc, st, g \in N \tag{13}$$

where sc is the CSPRNG's initial counter (seed counter), st is the start time of the encryption relative to the encrypter's world-clock t_{tx}, and g is the key generation rate (values per time unit). This initial configuration can be used by a decrypter to calculate the current CSPRNG-counter (cc) which can be used to generate the current key-frame $\boldsymbol{k}$.

To synchronize using time references, the decrypter must take into account the propagation delay ρ and the drift between its world-clock (t_{rx}) and the encrypter's (t_{tx}). This time delay δ can be described as $\delta = (t_{rx} - t_{tx}) + \rho$. Once δ is known, the decrypter can calculate the current CSPRNG-counter value as follows

$$cc = [((t_{rx} + \delta) - st) \times g] \bmod P \tag{14}$$

where P is the counter's period for the CSPRNG.

To account for δ, we propose that the decrypter seek it out from the received signal. This can be achieved by (1) using (14) without δ to find a nearby counter, and then (2) finding the spot in the current signal where the autocorrelation of the decrypted signal is the least like white noise. In order to measure the similarity of a decrypted signal to white noise we use the metric

$$\alpha = \frac{\sum_{k=1}^{N-1} R_{\boldsymbol{d},\boldsymbol{d}}[k]}{R_{\boldsymbol{d},\boldsymbol{d}}[0]} \tag{15}$$

where $R_{\boldsymbol{x},\boldsymbol{x}}[n]$ is the discrete autocorrelation of the vector $\boldsymbol{x}$ at index n, and $\boldsymbol{d}$ is the decrypted frame.

Since we have temporally found the frame for the respective key, we can now calculate δ. This can be done by solving

$$cc_{\boldsymbol{k}} = [((t_{usedRx_k} - t_{peak_k}) - st) \times g] \bmod P \tag{16}$$

where $cc_{\boldsymbol{k}}$ and t_{usedRx_k} are the counter and "current time" used when creating $\boldsymbol{k}$ and t_{peak_k} is the timestamp where (15) is maximized. Moreover, one can increase the precision by averaging the results over several consecutive counters. Figure 5 demonstrates this on a signal captured between two Arduino development boards.

7 Complexity and Performance

As mentioned in Sect. 4, the operations that take place at each end are: DFT, DFT^{-1}, modulo-add or modulo-subtract and key generation. The most computationally expensive operations are the DFT and DFT^{-1} operations performed on each frame. Since the VPSC operates on real-valued signals, a trick can be performed to reduce complexity. The trick is to put $N/2$ samples into each both real and complex inputs of a $\frac{N}{2}$ DFT [11]. Therefore, the complexity of the encryption function $e_{\boldsymbol{k}}$ and decryption function $d_{\boldsymbol{k}}$ is $O\left(N\log\left(\frac{N}{2}\right)\right)$ Today it is possible to find inexpensive hardware-accelerated DSP chips capable of performing them at high speeds. For example, the processor shown in [1] calculates a

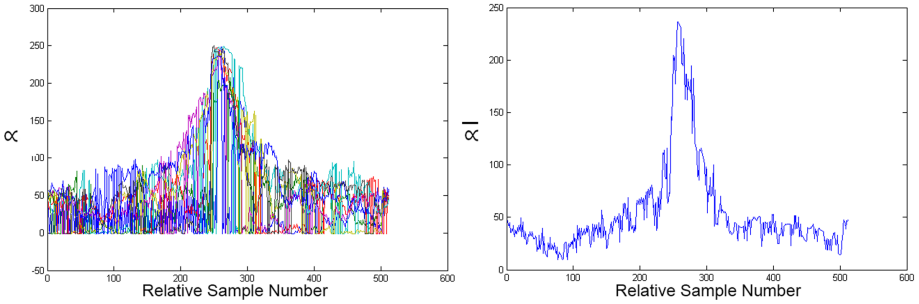

Fig. 5. Receiver synchronization to handle propagation and sender-receiver world clock delays though autocorrelation over several consecutive counters (left) and when averaged together (right).

1024-point complex FFT at a 32-bit precision in 23.2 µs. Using optimizations for real signals, it is possible to achieve even lower processing times. The same processor can also perform the division required for the modulo operations in 8.75 ns, which is negligible compared to the FFT processing time. Therefore, a frame of 1024-samples could undergo encryption and decryption on this chip in about 92.8 µs

To increase the speed further, or handle larger frames, one can use several VPSC encrypters and decrypters in parallel. This would be done by having each pair operate on a slice of the signal's spectrum and by combining the slices back together in the time plane.

8 Evaluation

In this Section, we verify the VPSC's practicality in realistic scenarios, such as both wired and wireless channels. We accomplish this by first evaluating the VPSC in a realistic channel simulator, and then by implementing the VPSC on actual hardware as a proof-of-concept.

8.1 Wireless – Simulation

The most practical use for a physical signal cipher, is to protect channels which (1) are easily accessible by an eavesdropper/attacker, and (2) do not require third parties to interpret and relay the signal (e.g. switches and routers). For these reasons, we evaluate the VPSC's performance in a wireless channel. We also note that wireless channels are significantly more challenging than in wired channels due to multi-path propagation and other distortions.

Experiment Setup: To evaluate VPSC in a wireless channel, we developed a simulator in Python which uses the configuration of an LTE OFDMA

Table 1. The parameters taken for the simulations.

	Parameter	Value
LTE Channel	Multiplexing	OFDMA
	Modulation	QAM 16
	Channel (f_c)	1900 MHz
	Channel Bandwidth	1.4 Mhz
Noise: AWGN	SNR [dB]	10
Interference: Multi-path propagation + Doppler	Number of paths	4
	Path Delays [μsec]	0.0, 0.5, 1.0 , 0.2
	Path Gains [dB]	0, -9, -12, -6
	Relative velocity [m/s]	1.4 (walking speed)
Signal Processing	Sample Rate (f_s)	15.3 Mhz
	Frame size (N)	1024
	VPSC/FCS Encryption band	1899.3 – 1900.7 MHz
Simulation	Number of bits	40,000
	Number of QAM symbols	10,000

mobile wireless channel.[3] The simulator can apply additive Gaussian white noise (AWGN), multi-path propagation (MPP) interference based on Rayleigh fading, and the Doppler effect (assuming a mobile receiver). The default configurations are listed in Table 1 unless mentioned otherwise. Furthermore, in all simulations, we applied the VPSC's combined method of noise mitigation from Algorithms 2 and 3, and set $\lambda = (N_0)^{-10}$, where N_0 represents the mean of the AWGN energy.

As a baseline comparison, we evaluated the VPSC against an unencrypted channel and three other physical signal ciphers: FCS, ALM, and RSA (see Sect. 2). For the VPSC and FCS, we encrypted the channel's bandwidth only, while the other methods had to encrypt the entire spectrum.

For each simulation, the following steps are performed until 40k bits are sent: (1) four random bits are modulated into a QAM-16 symbol, (2) the corresponding signal is generated on carrier frequency f_c for 66.7 µs (the LTE OFDMA symbol duration), (3) the raw signal is encrypted and sent over the noisy channel, (4) the received signal is decrypted, (5) the symbol is demodulated, and (6) the bit error rate (BER) is updated.

Results–Signal Quality: In Fig. 6, we present the BER plots for all methods of encryption after passing through an AWGN channel, and a channel with both AWGN and MPP. The figure shows that the VPSC is more robust to AWGN than the other methods up to about 8dB SNR. Moreover, the VPSC is more robust than all other methods in the case of MPP, making the VPSC a better choice for wireless channels. The reason why the VPSC performs better than an unencrypted signal up to 6dB SNR is because the noise mitigation adds some energy for the buffers.

Note that FCS is robust to AWGN, but not to MPP. This is because FCS shifts the majority of the signal's energy to the left or right of the carrier wave,

[3] The Python source code to the VPSC can be found on online: https://github.com/ymirsky/VPSC-py.

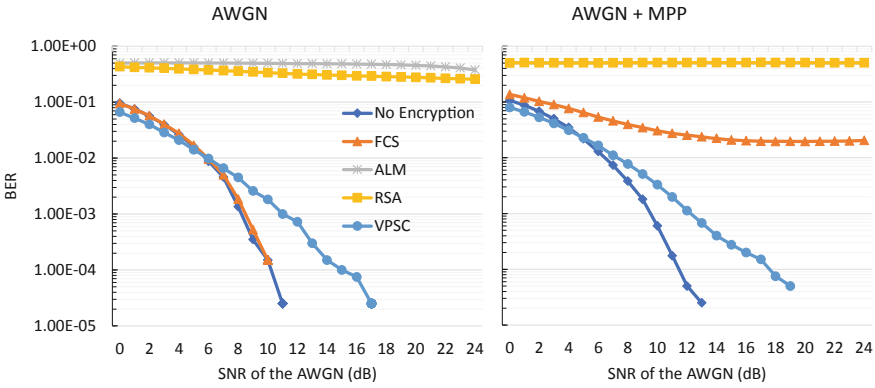

Fig. 6. The bit error rate plots for all ciphers when introduced to AWGN (left) and both AWGN with multipath propagation and Doppler effect (right).

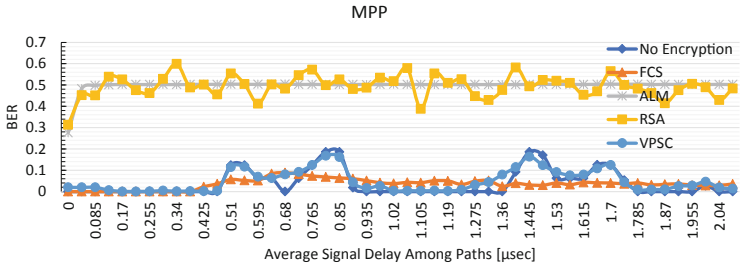

Fig. 7. The bit error rate plots for all ciphers when introduced to multipath propagation and Doppler effect, with increasing propagation delays.

thus increasing the interference of MPP and the Doppler effect across the channel's band. The ALM and RSA methods fail completely even in the presence of a minute amount of noise (24 dB SNR), requiring 300 dB SNR for zero errors. ALM fails because the logarithm operations increase the noise. RSA fails because the function acts like a hash map, resulting in large discrepancies for small input variations. To visualize the effect of the encryption, we provide a 16-QAM constellation plot of the demodulated symbols for various channels encryption methods in Fig. 10 of the appendix.

In Fig. 7, we examine the effect of the propagation delay in an MPP channel. In this simulation, we scaled up the propagation delays of the signal paths (Table 1). As a result, interference increases when there are symbols that overlap (ISI) which can be seen in the plot. The figure shows that the VPSC is nearly as robust to MPP as the original signal (without encryption). We note that FCS never has a 'peak' of interference, but rather always stays at a non-zero BER. This is because most of the signal's energy falls on a single frequency component, and the random shift of this component's location helps mitigate ISI.

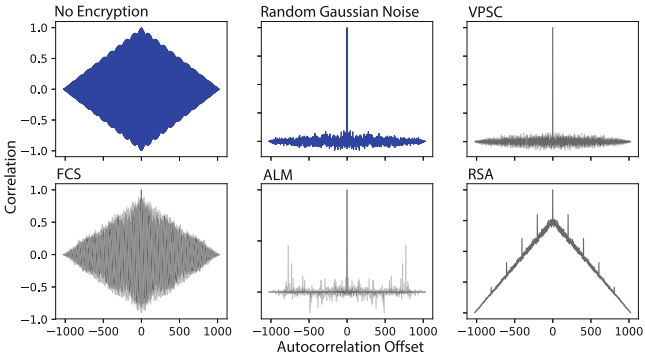

Fig. 8. The autocorrelation of a 16-QAM signal, encrypted by each of the ciphers three times (each time with a different key).

Results–Signal Secrecy: In order to measure the secrecy of the signal ciphers, we performed autocorrelations on the 16-QAM cryptograms (encrypted signals) for each of the ciphers. An autocorrelation measures the correlation of a signal with itself at various offsets. By measuring this self similarity, we can see evidence on whether or not a correlation attack may be performed. If a signal reveals no information about its contents, then the signal is essentially white noise. The autocorrelation of white noise is extremely low at every offset, except for when the signal overlaps itself completely.

In Fig. 8, we present the autocorrelation of a 16-QAM signal, encrypted by each of the signal ciphers three times: each time with a different key. The plots reveal that the VPSC has the same autocorrelation as random noise. This makes sense since each of the FFT's frequency components holds a random magnitude and phase as a result of the selected key. The figure also shows that the FCS does not protect the channel's content, but rather only obfuscates it. This is also apparent from the spectrum of the FCS's cryptograms, illustrated earlier in Fig. 1. The ALM cipher provides a relatively good encryption since it resembles white noise. However, the ALM has *spikes* in its autocorrelations. This means that some information is being leaked. This imperfection is likely due to the fact that ALM multiplies each sample with a value which is never zero. As a result, the cryptogram's distribution can reveal a portion of the contents. Given enough cryptograms, it may be possible to correlate out the ciphered signal. Finally, the RSA method fails to protect the signal's contents. This is due to an oversight in the cipher's implementation. Specifically, the RSA method encrypts the samples with the same private key, otherwise the complexity (of finding prime numbers and exchanging public keys in real-time) would be too high. As a result, all samples with the same value are mapped to the same location after encryption. Although this process obfuscates the ciphered signal, some of its original frequencies are still retained. An illustration of this effect can be seen in Fig. 11 in the appendix.

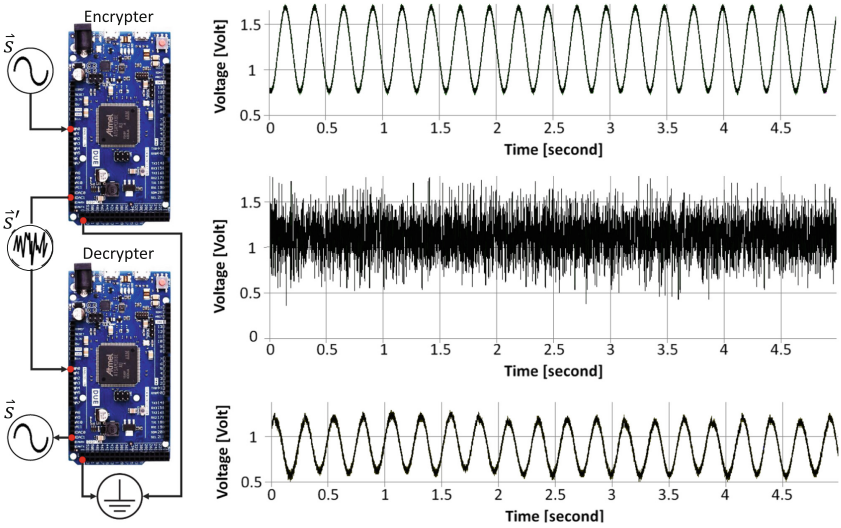

Fig. 9. The encryption and decryption of 4 Hz sine wave using the prototype, with the original signal, (top) the intercepted encrypted signal (middle), and the decrypted signal (bottom).

In summary, the VPSC is suitable for encrypting physical signals which traverse real-world channels. The VPSC is significantly more secure than other physical signal ciphers.

8.2 Wired – Proof of Concept

We implemented a hardware proof of concept for the VPSC to (1) illuminate any overlooked issues with the concept, and (2) demonstrate that the system works in practice. The VPSC prototype was implemented across two Arduino Due development boards. These boards embed 32-bit ARM core microcontrollers clocked 84 MHz with 512 KB of flash memory and numerous embedded digital I/O pins. One board was designated as the encrypter and the other as the decrypter (left side of Fig. 9). The boards contain both digital-to-analog converters (DAC) and analog-to-digital converters (ADC), which were used for transmitting and receiving signals respectively. We implemented the VPSC in C++ and open-source libraries were used for the Fast Fourier Transforms and the PRNG (SHA-1 as a counter-mode cipher).

In order to start the prototype, each board was given the same initial configuration as presented in (13). The time delay inference algorithm that uses (15) was verified offline using samples captured from the boards, plotted in Fig. 5. Using these samples, the receiver was able to synchronize and correctly decrypt every frame.

To test the prototype, we used a sine wave as the source signal, with an amplitude 1 V and a frequency of 4 Hz. The sample rate f_s was 1 kHz and the frame size N was 256. We chose a sine wave because many common modulation schemes are based upon it (e.g., ASK, FSK, QAM, etc.) As shown in Fig. 9, the prototype was able to successfully reconstruct the encrypted signal.

9 Conclusion

It is sometimes highly desirable to provide the highest level of security to communications systems. By using the VPSC, it is possible to encrypt any waveform signal to a high degree of secrecy while maintaining the same amount of bandwidth (an expensive commodity in RF). Moreover, operation on the frequency domain has many advantages such as parallelization on the hardware level. To ensure the stability of the VPSC, we have proposed two mitigation techniques when noise is introduced, and recommend using a combination of both.

To evaluate the VPSC, we implemented three other known physical signal ciphers. We then simulated an LTE OFDMA wireless channel, with noise and interference, and measured the ciphers' performance in carrying a 16-QAM modulation. Furthermore, we explored the secrecy of each of the ciphers by examining the autocorrelations of their respective encrypted signals. Our evaluations demonstrated that the VPSC is not only the most suitable physical signaling cipher in noisy channels, but also the most secure.

In summary, the VPSC offers a powerful method of encrypting any waveform signal (as well as complex), with a trade-off between security and power efficiency.

Acknowledgements. This research was partly funded by the Israel Innovations Authority under WIN - the Israeli consortium for 5G Wireless intelligent networks.

10 Appendix - Additional Figures

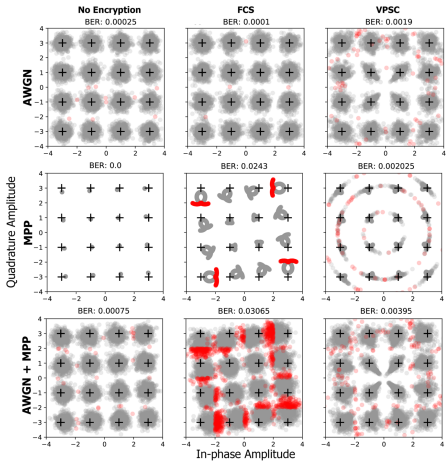

Fig. 10. QAM-16 constellation plots of the deciphered and demodulated symbols (1900 MHz LTE OFDMA), with various types of noise and interference, where red indicated incorrectly demodulated symbols. (Color figure online)

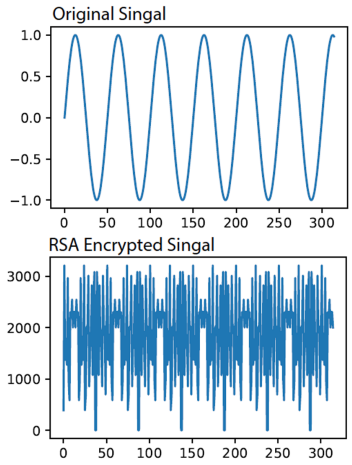

Fig. 11. The RSA method's failure demonstrated by a sine wave on the top (plaintext) and the encrypted RSA signal on the bottom (ciphertext).

The proof that the Vernam Cipher and OTP can be extended from binary to N-ary values with out loss of secrecy, can be found here: https://github.com/ymirsky/VPSC-py/blob/master/Additional%20Proofs.pdf.

References

1. Sharc processor adsp-21367 reference, datasheet (2013). http://www.analog.com/static/imported-files/data_sheets/ADSP-21367_21368_21369.pdf
2. Barker, E.B., Kelsey, J.M.: Recommendation for random number generation using deterministic random bit generators. NIST Special Publication 800–90A (2012)
3. Bloch, M., Barros, J.: Physical-Layer Security: From Information Theory to Security Engineering. Cambridge University Press, Cambridge (2011)
4. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. J. Comput. **15**(2), 364–383 (1986)
5. Csiszar, I., Korner, J.: Broadcast channels with confidential messages. IEEE Trans. Inf. Theory **24**(3), 339–348 (1978). https://doi.org/10.1109/TIT.1978.1055892
6. Dworkin, M.: Recommendation for block cipher modes of operation-methods and techniques. NIST Special Publication 800–30A (2001)
7. Ferguson, N., Schneier, B., Kohno, T.: Cryptography Engineering: Design Principles and Practical Applications, p. 70. Wiley, Hoboken (2012). Chap. 4

8. Garrett, P., Lieman, D.: Public-key Cryptography: Baltimore (Proceedings of Symposia in Applied Mathematics) (Proceedings of Symposia in Applied Mathematics). American Mathematical Society, Boston (2005)
9. Hudde, H.C.: Building stream ciphers from block ciphers and their security. Seminararbeit Ruhr-Universität Bochum (2009)
10. Jo, Y., Wu, D.: On cracking direct-sequence spread-spectrum systems. Wirel. Commun. Mob. Comput. **10**(7), 986–1001 (2010)
11. Jones, K.: Fast solutions to real-data discrete Fourier transform. In: Jones, K. (ed.) The Regularized Fast Hartley Transform, pp. 15–25. Springer, Dordrecht (2010). https://doi.org/10.1007/978-90-481-3917-0_2
12. Kang, W., Liu, N.: Wiretap channel with shared key. In: 2010 IEEE Information Theory Workshop (ITW), pp. 1–5, August 2010. https://doi.org/10.1109/CIG.2010.5592665
13. Khalil, M.: Real-time encryption/decryption of audio signal. Int. J. Comput. Netw. Inf. Secur. (IJCNIS) **8**, 25–31 (2016)
14. Law, Y.W., Palaniswami, M., Hoesel, L.V., Doumen, J., Hartel, P., Havinga, P.: Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols. ACM Trans. Sen. Netw. **5**(1), 6:1–6:38 (2009)
15. Marton, K., Suciu, A., Ignat, I.: Randomness in digital cryptography: a survey. ROMJIST **13**(3), 219–240 (2010)
16. Matsunaga, A., Koga, K., Ohkawa, M.: An analog speech scrambling system using the FFT technique with high-level security. IEEE J. Sel. Areas Commun. **7**(4), 540–547 (1989). https://doi.org/10.1109/49.17718
17. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. Discrete Mathematics and Its Applications. Taylor & Francis, Boca Raton (1996)
18. Nichols, R., Lekkas, P.: Wireless Security: Models, Threats, and Solutions. McGraw-Hill Telecom Professional. McGraw-Hill, New York (2002)
19. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM **21**(2), 120–126 (1978)
20. Romanow, A.: IEEE standard for local and metropolitan area networks-media access control (MAC) security. IEEE Std 802.1AE-2006, pp. 1–142 (2006). https://doi.org/10.1109/IEEESTD.2006.245590
21. Shannon, C.E.: Communication theory of secrecy systems. Bell Syst. Tech. J. **28**(4), 656–715 (1949)
22. Shiu, Y.S., Chang, S.Y., Wu, H.C., Huang, S.H., Chen, H.H.: Physical layer security in wireless networks: a tutorial. IEEE Wirel. Commun. **18**(2), 66–74 (2011). https://doi.org/10.1109/MWC.2011.5751298
23. Vacca, J.: Computer and Information Security Handbook. Elsevier Science, Amsterdam (2012)
24. Vernam, G.S.: Secret signaling system, July 1919. US Patent 1,310,719
25. Wyner, A.D.: The wire-tap channel. Bell Syst. Tech. J. **54**(8), 1355–1387 (1975)
26. Yamamoto, H.: Rate-distortion theory for the Shannon cipher system. IEEE Trans. Inf. Theory **43**(3), 827–835 (1997). https://doi.org/10.1109/18.568694
27. Zhou, X., Song, L., Zhang, Y.: Physical Layer Security in Wireless Communications. Wireless Networks and Mobile Communications. Taylor & Francis, Boca Raton (2013)