Mitigating Rate Attacks through Crypto-Coded Modulation

Triet D. Vo-Huu
vohuudtr@ccs.neu.edu

Guevara Noubir
noubir@ccs.neu.edu

College of Computer and Information Science
Northeastern University
Boston, MA 02115

ABSTRACT

Exposing the rate information of wireless transmissions enables highly efficient attacks that can severely degrade the performance of a network at very low cost. In this paper, we introduce an integrated solution to conceal the rate information of wireless transmissions while simultaneously boosting the resiliency against interference. The proposed solution is based on a generalization of Trellis Coded Modulation combined with Cryptographic Interleaving. We develop algorithms for discovering explicit codes for concealing any modulation in BPSK, QPSK, 8-PSK, 16-QAM, 64-QAM. We propose a 2-pass frequency correction and phase tracking mechanisms that enables the proposed schemes to reach their potential. We demonstrate that in most cases this rate hiding scheme has the side effect of boosting resiliency by up to 7dB (simulations) and 4dB (SDR experiments).

1. INTRODUCTION

Wireless communication is the key enabling technology of the Mobile Revolution that we are currently enjoying. Beyond enabling Mobile Phones, Wireless Sensor Networks, and the Internet of Things devices, it is also key to Cyber-Physical Systems such as SCADA Wireless Remote Terminal Units [46].

Achilles' Heel of Wireless. The broadcast nature of the wireless medium makes it vulnerable to two types of major attacks *denial of service*, and *information leakage*. Designing countermeasures to wireless DoS attacks before they become widespread is very important for both military and commercial applications. Due to a series of recent incidents, the FCC has stepped up its education and enforcement effort [12], rolled out a new jammer tip line (1-855-55NOJAM), and issued several fines [13]. At the same time jammers are becoming a commodity and are growing in sophistication and convenience of use and deployment. Beyond degrading a critical communication infrastructure, wireless DoS can also be the prelude to more sophisticated attacks where the adversary deploys rogue infrastructure [8]. Evidence of such attacks in the real world started emerging in the last few months [14, 42].

Weaknesses of Rate Adaptation Algorithms. We are interested in the impact of exposing the rate information, defined by the combination of Modulation and Coding Scheme (MCS), on enabling denial of service attacks. Recent work [33] showed that knowledge of the transmission rate enables selective jamming of packets resulting in very efficient attacks on all the Wi-Fi rate adaptation protocols investigated. Rate Adaptation Algorithms (RAA) adjust the physical layer transmission rate to the channel characteristics; ideally selecting a low rate MCS for a low SNR channel and a high rates for high SNR [5, 25, 26, 31, 34, 48, 50]. To illustrate the criticality of hiding the communication rate, we first provide some insights into rate adaptation attacks. The key idea in the highly efficient RAA-attack is to force a device to use a low rate (e.g., 1Mbps), by jamming all packets of higher rates. This allows: (1) a reflection attack where the victim is occupying the channel for very long time preventing other transmissions (i.e., 54 times channel occupancy for 802.11g to over 300 times for 802.11n), (2) a saturation of the network that induces a higher collision probability resulting in some RAA maintaining a network-wide low rate even when the adversary stops jamming. This self-sustained interference is called a congestion collapse. The RAA-attack is aggravated by the fact that jamming high rate packets is easier than jamming low rate packets and only requires interfering with few symbols. Furthermore, the comeback to higher rates is slow. Finally, the equiprobability of transmission among devices enables the adversary to focus on a single link to degrade the whole network [7]. Previous work [33, 36, 37] demonstrated both analytically and experimentally that smart RAA-aware jamming can lead to highly efficient attacks. The adversary only needs to jam a small carefully selected fraction (less than 5%) of the packets to achieve orders of magnitude more efficiency than blind jamming [33].

The fundamental reason why an adversary can selectively jam packets is because the rate information is either explicit (e.g., SIGNAL field of the IEEE 802.11 PLCP header) or implicit (analysis of I/Q constellation). Before describing our approach, we provide some context for our work in terms of related research.

Mitigating Jamming. Jamming at the physical layer has been extensively studied (cf. [39] for a recent monograph on this subject). Reliable communication in the presence of adversaries regained significant interest in the last decade, as new jamming attacks as well as the need for more complex applications and deployment environments have emerged (cf. [10, 38] and references therein). Specifically crafted attacks and counter-measures have been studied for packetized wireless data networks [20, 27, 28, 49], multiple access resolution [1, 2, 4], multi-hop networks [27, 44, 51], wireless sensor networks [52], spread-spectrum without shared secrets [23, 30, 43].

Related Work on RAA Attacks and Mitigations. Characterizing the vulnerabilities of RAA algorithms to jamming and developing countermeasures received increasing interest over the last

MobiHoc'15, June 22–25, 2015, Hangzhou, China.

http://dx.doi.org/10.1145/2746285.2746319.

few years from the first observations and mitigations through fixed rate communication during jamming [7, 37], to randomized mitigations [35, 36], to game theoretic randomization strategies [15–19]. However, previous work assumes that either the adversary is not able to selectively jam packets based on their rates [7, 35–37] (e.g., due to the slow reaction time of the jammer), or because the transmitter is able to effectively hide the rate [16, 18] (without proposing a concrete mechanism for rate-hiding). Unfortunately, over the last couple of years several recent work demonstrated that rate-selective reactive jammers are feasible on Ettus Software Defined Radio platforms, either on the host [33], or on the FPGA [32, 49]. RAA attacks are therefore fairly easy to implement on custom chips. As we will discuss in the next section, a key unresolved challenge in preventing RAA attacks is how to prevent an adversary from guessing the rate (Modulation, Coding information) during the transmission and therefore from selectively interfering with a packet.

A very recent work [41] proposed a modulation level encryption technique to hide the rate of communications. In essence, this technique always transmits with the highest order modulation, but the communicating nodes cryptographically agree on a subset of the constellation points to be used for each symbol. For example, BPSK modulation can be hidden in 16-QAM by only considering eight pairs of points. For every symbol to be transmitted a pair is cryptographically selected by the transmitter and is also known to the receiver through a shared key. The information bit of BPSK determines which element of the pair is sent. Since the eight pairs cover the whole constellation points, the adversary cannot distinguish between a BPSK communication embedded within 16-QAM or a true 16-QAM. While this scheme conceals the rate information, it does so at the cost of degrading the robustness of the communication. First, one can analytically show that 1-2dB are lost because of the constrained selection of the constellation pairs. Several additional dB are lost due to the poor performance of frequency offset correction, and phase tracking techniques.

Approach and contributions. One plausible approach to hide the implicit rate information, leaked by the constellation points, consists of always using the highest order modulation and combining it with a matching rate (e.g., uncoded BPSK can be hidden by using 16-QAM combined with a 1/4 rate coding scheme). As will be discussed and demonstrated in the next sections, such approach does not perform well, mainly because: (1) At low SNR several key components of the communication chain perform poorly, therefore degrading the performance of the system. These include the coding schemes, the frequency offset, and the phase tracking. This is one of the key reasons why most standards (both WLAN and Cellular) still rely on BPSK as a fallback solution for low SNR regimes. (2) Traditional codes maximize the Hamming distance between codewords and not the multi-dimensional Euclidean distance necessitated by the I/Q constellation of coded high order modulations. (3) An adversary can still guess the rate information by trying all possible modulation schemes. Traditional encryption schemes cannot prevent this attack since if they are applied post-coding/modulation they would render the error correction properties of the code useless (a single bit error would be amplified by the decryption process).

We introduce an integrated solution – Conceal and Boost Modulation (CBM) – to conceal the rate information of wireless transmissions while simultaneously boosting the communication resiliency against interference. The adversary sees all communications using the highest order modulation. The proposed solution is based on a Generalization of Trellis Coded Modulation (GTCM) with an integrated frequency offset and phase tracking technique combined with Cryptographic Interleaving. We developed novel algorithms for efficiently discovering and validating new trellis codes capable of upgrading any modulation constellation to any higher order constellation. While previous work on TCM was restricted to $\frac{k}{k+1}$ rate uniform trellis due to many reasons including complexity of searching over all possible trellis. We devise an efficient algorithm that finds the explicit codes for concealing any modulation in BPSK, QPSK, 8-PSK, 16-QAM, 64-QAM. In addition to the analytically proven properties of these codes, we show through simulations that in most cases this modulation hiding scheme has the side effect of boosting resiliency by up to 7dB over uncoded modulation and over 8dB in comparison with prior work [41]. However, in real experiments using a testbed of USRP N210 [11], we show that with a standard frequency offset correction and phase tracking, the performance of the proposed schemes is severely degraded. We develop a 2-pass frequency offset correction and phase tracking mechanisms that integrates within the GTCM decoder. This technique exploits the capability of today's baseband chips and SDR platforms to store a substantial number of samples therefore using information from the GTCM decoder to accurately track and correct the transient phase errors. Our final system achieves an experimentally measured performance boost of up to 4dB.

(a) 8-PSK (b) 16-QAM

Figure 1: Constellation guessing on USRP: Received symbols can be distinguished clearly after carrier synchronization and tracking.

2. RATE INFORMATION

2.1 Rate Information Leakage

Explicit rate information: In many communication protocols, the rate information of a transmission is unprotected. For instance in IEEE 802.11 networks, the rate is explicitly specified in the SIGNAL field of the physical layer's frames. An adversary can easily synchronize with the communication between two parties, analyze the data frames and extract the rate. This attack is very practical as demonstrated by [33]. Similarly, the rate information of a transmission in LTE cellular systems is exposed in the Modulation and Coding Scheme (MCS) field within the Downlink Control Information (DCI), which is itself encoded using a publicly known fixed rate $1/3$ convolutional code and QPSK modulation.

Modulation guessing: Even if the rate information is not explicitly provided within the packet header, the adversary can analyze the received signal in complex I/Q form. After performing the carrier synchronization, frequency and phase offset correction, the adversary can trace the received constellation pattern and determine the modulation in use. This method does not require the knowledge of the protocol's frame structure. We demonstrate the guessing attack by implementing a modulation detector on USRP, which can in real time identify the transmission's modulation (Figure 1). It can be easily extended to build a practical rate-aware jammer [33, 49] to selectively jam the high rate packets.

Code guessing: With more sophisticated techniques, an adversary could manage to identify not only the transmission's modulation, but also the codes in use. One such technique is to track the sequence of received symbols to guess the codes based on the fact that different codes produce different transitions from one coded symbol to another. Since most communication standards specify

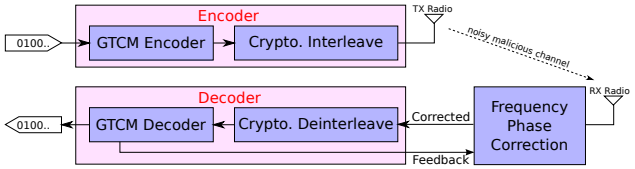


Figure 2: Overview of our CBM system.

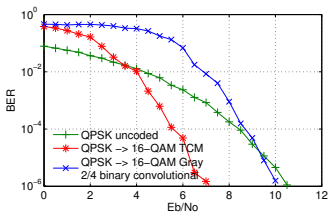


Figure 3: Performance comparison between (1) our TCM code with standard 16-QAM, and (2) best traditional binary code (from [9]) of rate $2/4$ with Gray coded 16-QAM.

a limited set of modulations and codes, guessing by matching and trial-and-error is efficient for the adversary.

2.2 Challenges to Rate Concealing

We now provide some insights why it is hard to hide the rate information.

Data Encryption: First, to hide the explicit rate information in the packet's protocol header, a straight-forward solution would be to encrypt the header. However, doing so does not prevent the modulation guessing attack, as a transmission still exposes the modulation to the complex I/Q analyzing adversary, who does not even need to decrypt the header to learn the rate.

Using Single Modulation: Alternatively, in order to avoid modulation guessing, a naive solution could be always using only one modulation for communications. This, however, lacks the flexibility and adaptivity to the environment, such as preventing the user from benefiting from high data rates at high SNR.

Modulation Level Encryption: Recently, the modulation level encryption technique developed by Rahbari and Krunz [41] can hide the rate from modulation guessing by a random mapping into a higher-order modulation based on a shared secret between two parties. However, their solution sacrifices the resiliency by 1-2dB in simulation, and several dB in real world due to imperfect frequency offset correction.

Coding: To improve the system robustness, one can propose using binary error correction codes combined with the highest order modulation. However without a careful design of the codes, the transmission performance is not guaranteed even with the best binary codes (as we show later in Section 4). Moreover, simply applying coding with modulation level encryption still leaks the rate information to code guessing attacks.

3. APPROACH

Our scheme – Conceal and Boost Modulation (CBM) – is depicted in Figure 2. The General Trellis Coded Modulation (GTCM) module's functionalities are two-fold. First, it makes the constellation pattern indistinguishable to the adversary, therewith countering the modulation guessing attacks. Second, it boosts the system resiliency against interference. The Cryptographic Interleaving (CI) module conceals the rate information from explicit rate exposing and implicit code guessing attacks.

Our idea for hiding the constellation is to always use a *single unifying* modulation (the highest order) to transmit data in order to create the same constellation observed by the adversary. To preserve the bit rate and robustness supported by the original modulation, the GTCM module encodes the data by a suitable code of rate matching the bit rate ratio between the original modulation and the target modulation. To be precise, let's consider a system that supports a set of different modulations ordered by the number of bits per symbol (bps) $b_1 \leq \ldots \leq b_{\mathcal{N}}$, where $\mathcal{N}$ denotes the highest-order modulation of bit rate $n = b_{\mathcal{N}}$. Assume that the on-going transmission is desired to be carried at a bit rate $k = b_{\mathcal{K}}$ bps for some modulation $\mathcal{K}$. In order to conceal the constellation, we encode the data using an adequate code of rate k/n and transmit the encoded data using the target modulation $\mathcal{N}$. Since the adversary will always observe the same constellation $\mathcal{N}$, the actual rate is concealed from modulation guessing attacks.

To counter the code guessing attacks, we develop a cryptographic module, which interleaves the modulated symbols before transmission. We emphasize that the interleaving process is performed at the baseband samples level, i.e., complex symbols produced by the GTCM module are interleaved per block of transmitted symbols. We note that straightforward encryption of data *before* modulating does not conceal the rate information, as the adversary can clearly observe the constellation of encrypted data. We derive a specific method to efficiently generate cryptographic interleaving functions used for permuting the output symbols from the GTCM module in such a way that the transmit stream does not leak the rate information. For the receiver to be able to decode the data, the rate information is embedded into the packet in an encrypted form such that only the receiver, who shares the secret key with the transmitter, can decrypt the information.

It is important to understand the implications of rate hiding. On one hand, the highest-order modulation creates redundancy by the constellation expansion. On the other hand, the constellation points' pair-wise distances are closer than in the original constellation. Without good design specifically targeting to the upgraded modulation's constellation, the system can become less resilient against interference. For example, the modulation unification technique [41] results in loss of 1-2dB of robustness compared to regular rate-exposing systems. This is because no coding is used in their system. However, even using good traditional binary codes cannot guarantee the robustness of the system because they maximize the Hamming distance between codewords and are not designed for coded modulation. An illustration is shown in Figure 3. We take the best code $\begin{pmatrix} 17 & 13 & 05 & 02 \\ 10 & 03 & 17 & 15 \end{pmatrix}$ of rate $2/4$ from Table VII in [9], and use it with Gray coded 16-QAM modulation. Comparing it with our derived TCM code $\begin{pmatrix} 01 & 12 & 16 & 11 \\ 01 & 13 & 16 & 11 \end{pmatrix}$ of the same rate and constraint, we see a gain of about 4dB is achieved with our code (Figure 3), while the binary code almost gives no advantage over uncoded QPSK at BER $= 10^{-6}$. Therefore, with good codes designed for the target modulation, we can gain instead of losing.

Searching for good codes for the rate-hiding systems must take into account the constellation description defined by the highest-order modulation. This idea is rooted in the Trellis Coded Modulation (TCM) technique introduced by Ungerboeck [45], who focused on devising modulation codes of rate $k/(k+1)$. In the literature, finding good TCM codes is a challenging problem, for which only heuristic solutions have been studied such as the set partitioning rules established in [45]. Unfortunately, there is no polynomial time algorithm for constructing the optimal general TCM codes. In this work, we introduce a new heuristic approach for upgrading

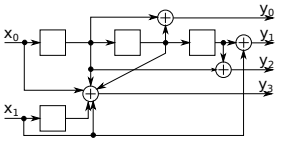


Figure 4: Our best TCM code of rate $2/4$ (QPSK $\rightarrow$ 16-QAM) and constraint length 4. Its boosting gain over uncoded QPSK is 3.8dB.

arbitrary modulations. Our heuristic solution is not based on the conventional set partitioning technique. Instead, we generate the code by randomizing the mapping between the inputs, shift registers, and the outputs of the general code structure. Our results show that this approach can find codes at least as good as the ones found in [45]. In practice only applying codes to an existing system is not sufficient due to imperfections caused by frequency and phase offset. We address these issues by devising a two-pass synchronization mechanism, which will be discussed in Section 6.

4. GENERAL TCM

In this section, we describe the fast search procedure for TCM codes of arbitrary rate k/n, which are used to encode data originally modulated by a modulation $\mathcal{K}$ of order 2^k (bit rate k) to the highest-order modulation $\mathcal{N}$ of order 2^n (bit rate n), without any uniformity restriction. For convenience, we give a brief overview of TCM codes. A TCM code is a convolutional code (k, n) defined by a set of k shift registers storing the code's k input bits, and a generator matrix which specifies the input-output relation (e.g., Figure 4). Deeper shift registers have higher potential gain and decoding complexity. Thus, we classify the codes by their constraint length v defined as $v = \sum_{i=1}^{k} v_i$, where v_i is the length of the i-th shift register. In our search procedure, we only consider the feed-forward construction of convolutional codes, because any construction with feedback can be transformed into a feedback-free construction that produces equivalent codewords [29]. To represent a code, we use the conventional generator polynomial form $G(D) = \{g_{ij}(D), i = 1 \ldots k, j = 1 \ldots n\}$, where $g_{ij}(D) = \sum_{l=0}^{v_i} a_l D^l$ is a univariate polynomial, and the indeterminate D represents the delay of the input bit in the corresponding shift register. If $a_l = 1$, the i-th input's current value (for $l = 0$) and past values (for $l > 0$) are $GF(2)$ added (exclusive-or) to the j-th output. For example, the convolutional code $(2, 4)$ in Figure 4 has the generator matrix/polynomials $G = \begin{bmatrix} D+D^2 & D^3 & D+D^3 & 1+D+D^2 \\ 0 & 1 & 0 & 1+D \end{bmatrix}$.

Unlike binary convolutional codes whose performance depends on the Hamming distance of the binary output symbols, TCM codes' performance is determined by the free *Euclidean* distance d^∞, which is the minimum Euclidean distance of any two distinct complex-symbol sequences produced by the code and modulation $\mathcal{N}$. Since binary codes are not designed for coded modulation, they do not take into account the constellation mapping. The best binary code with regards to Hamming distance can have a significantly small Euclidean distance between transmitted complex symbols and result in poor performance when combined with a specific modulation. For example, Figure 3 shows that our TCM code (QPSK$\rightarrow$16-QAM) outperforms the best binary $2/4$ convolutional code in [9] applied to a Gray-coded 16-QAM modulation, by 3dB at BER=10^{-6}. In the search for good TCM codes, we use as comparison metrics the asymptotic coding gain ratio measured by $\beta = d^\infty_{\mathcal{N}}/\Delta_{\mathcal{K}}$, where $\Delta_{\mathcal{K}}$ is the minimum Euclidean distance between constellation points in the original modulation $\mathcal{K}$. Good TCM codes must have high β ratio.

4.1 Code search algorithm

We introduce a new heuristic approach for searching for good TCM codes. For a given code specification $(k, n, \{v_i\})$, the coefficients of the generator polynomials g_{ij} are randomly selected. Since each combination of coefficients corresponds to a unique code construction, we check the generated code for its free Euclidean distance and if it satisfies additional critical properties such as structure information leakage. The search is performed for a fixed number of trials independent of the code specification, thus it is substantially faster than a full search which evaluates all possible codes. Yet, as shown in Section 4.3, our randomization approach can achieve the same results as a full search. The search procedure is illustrated in the RANDOMCODESEARCH algorithm below.

Our random search is characterized by the number of trials T performed by the algorithm. A randomly generated code, might be (1) *catastrophic*, i.e., there exists a non-zero input sequence that can produce all-zero output sequence; or (2) *non-equiprobable*, i.e., the output values are not uniformly distributed, which can help the adversary deploy statistical attacks to distinguish the mapping we aim to conceal. Therefore, the generated code is first validated against above properties, then its free distance d^∞ is computed.

```
RANDOMCODESEARCH(k, n, {v_i}, M, T)
1  d^∞ = 0            // free distance of current best code
2  for i = 1 to T
3      C = generateCode(k, n, {v_i})
4      if valid(C)  // non-catastrophic and equiprobable check
5          d = COMPUTEDISTANCE(C, M, d^∞)
6          if d > d^∞
7              d^∞ = d      // update free distance
8              C* = C       // store new best code
9  return (C*, d^∞)
```

4.2 Free distance computation algorithm

The computational bottleneck of the code search lies in the computation of the Euclidean free distance, since it is performed for every generated code. In the conventional TCM code construction method based on set partitioning rules [45], computing the free distance only involves finding the minimum distance to the all-zero sequence. However, our GTCM approach does not restrict the search by the set partitioning rule, as we consider a higher-dimension space so that better codes can be found (including non-uniform ones). As a result, computing the free Euclidean distance involves all pairs of output sequences. Nevertheless, we devise an efficient algorithm – COMPUTEDISTANCE – whose running time is on average less than 2ms, on a 3GHz CPU desktop computer, for the modulations and depths we consider. COMPUTEDISTANCE's algorithm consists of traversing the trellis of the code and appropriately updating the *state-distances*, which is defined below.

First, we introduce some convenient notations. Let $I = \{0, \ldots, 2^k - 1\}$ be the set of inputs, $O = \{0, \ldots, 2^n - 1\}$ the set of output symbols, and $\Lambda = \{0, \ldots, 2^v - 1\}$ the set of possible states corresponding to a code C. A path P of length L is defined as a sequence of 3-tuples $P = \{(S_i, x_i, y_i), i = 0 \ldots L-1\}$, where $S_i \in \Lambda, x_i \in I$ are respectively the state and input of the code at time i, and $y_i \in O$ is the output symbol due to input x_i at state S_i. The encoding can start from any initial state S_0. Since the output symbol is mapped to a specific constellation $\mathcal{M}$, the Euclidean distance between two paths P and $\tilde{P}$ of length L is dependent on $\mathcal{M}$ and computed by $d_{\mathcal{M}}(P, \tilde{P}) = \sum_{i=0}^{L-1} d_{\mathcal{M}}(y, \tilde{y})$, where $d_{\mathcal{M}}(y, \tilde{y})$ gives the Euclidean distance between two points y and $\tilde{y}$ on the target coded modulation $\mathcal{M}$'s constellation. Now, we define the state-distance $D[S, \tilde{S}] \triangleq \min\{d(P, \tilde{P})\}$ of two states S and $\tilde{S}$ as

the minimum Euclidean distance between all possible paths of the same length ending at state S and $\tilde{S}$, respectively.

The key idea of our algorithm is that we update $D[S, \tilde{S}]$ gradually when traversing the trellis with increasing L. When two paths P and $\tilde{P}$ merge at the same state $S = \tilde{S}$, the free distance d^∞ is checked and updated with $D[S, \tilde{S}]$.

```
COMPUTEDISTANCE(C, M, d^∞_best)
 1  D[S, S̃] = ∞ for all (S, S̃) ∈ V²          // state-distances
 2  d^∞ = ∞                                  // C's free distance
 3  for each S ∈ Λ, (x, x̃) ∈ I², x ≠ x̃
 4      UPDATEDISTANCE(S, x, S, x̃)
 5  repeat
 6      for each (S, S̃) ∈ Λ², S ≠ S̃, D[S, S̃] < d^∞
 7          for each (x, x̃) ∈ I²
 8              UPDATEDISTANCE(S, x, S̃, x̃)
 9              if d^∞ ≤ d^∞_best
10                  return d^∞               // not the best, return now
11  until (S, S̃) not found in line 6
12  return d^∞                               // we found a better code
```

The algorithm COMPUTEDISTANCE starts by initializing the state-distances to the distance between any path P and $\tilde{P}$ starting from any *same* state S (line 1–4). We make the paths diverge from the same state (line 3), then compute the distance between them (line 4). In the main loop (line 5–11), the state-distances are repeatedly updated for each new segment added (line 7) to the paths until there exist no more state pairs $(S, \tilde{S})$ whose state-distance $D[S, \tilde{S}]$ is less than d^∞ (line 11). The maintenance and update of state-distances in both the initialization and the main loop are performed by UPDATEDISTANCE, which keeps records of $D[S, \tilde{S}]$ for all $S, \tilde{S}$. Whenever two paths P and $\tilde{P}$ merge at a state S, the corresponding state-distance $D[S, S]$ is checked to update d^∞.

```
UPDATEDISTANCE(S, x, S̃, x̃)
 1  T = C.nextState(S, x); y = C.output(S, x)
 2  T̃ = C.nextState(S̃, x̃); ỹ = C.output(S̃, x̃)
 3  if S = S̃
 4      d = d_M(y, ỹ)                        // update at initialization
 5  else
 6      d = d_M(y, ỹ) + D[S, S̃]              // update loop
 7
 8  if d < D[T, T̃]
 9      D[T, T̃] = d
10      if d < d^∞ and T = T̃                 // two paths merge
11          d^∞ = d
```

Furthermore in RANDOMCODESEARCH, where each generated code is computed for the free distance, we speed up the search by storing the best free distance d^∞_{best} associated to the best code C^* discovered so far in order to quickly eliminate codes of free distance shorter than d^∞_{best} (line 9 in COMPUTEDISTANCE).

Correctness. To prove the correctness of the algorithm, we show that the state-distances $D[S, \tilde{S}]$ keep records of the distances of all possible "close" paths. The proof is based on the following lemma.

LEMMA 1. *At any time i on the code trellis, for any pair of paths P and $\tilde{P}$, if there exists another pair of paths Q and $\tilde{Q}$ such that $S_i^{(P)} = S_i^{(Q)}$, $S_i^{(\tilde{P})} = S_i^{(\tilde{Q})}$, and $D[Q, \tilde{Q}] < D[P, \tilde{P}]$, then P and $\tilde{P}$ can be eliminated.*

PROOF (SKETCH). By the lemma's assumption, P merges with Q and $\tilde{P}$ merges with $\tilde{Q}$ at time i. It is followed that at time $i + 1$, any new pair evolved from P and $\tilde{P}$ will find a similar new pair evolved from Q and $\tilde{Q}$. Therefore, the pair $(P, \tilde{P})$ cannot have shorter distance and can be eliminated from searching. □

Proof of correctness. At initialization of COMPUTEDISTANCE, $D[S, \tilde{S}]$ are set to non-infinity values only for pairs of paths starting from the same state (line 3). This means that $D[S, \tilde{S}]$ properly reflect the distances of paths at initial states. In the main loop, the algorithm traverses every transition of the trellis and updates the state-distances. By Lemma 1, the macro UPDATEDISTANCE will discard paths corresponding to greater distance $D[S, \tilde{S}]$ and keep the ones corresponding to the shortest distance so far (line 9). Therefore, no closest pairs are eliminated by the algorithm.

To see that the algorithm terminates, we show that there exists a time such that $D[S, \tilde{S}] \geq d^\infty$ for all state pairs. It is enough to show that $D[S, \tilde{S}]$ are increasing while d^∞ is decreasing. The former is correct because evolving paths always contain transitions that results in positive increment in distance. The latter is due to the update in UPDATEDISTANCE. This concludes the proof.

Computational Complexity. The time complexity $g(t)$ of COMPUTEDISTANCE depends on the length L of paths where the free distance is found. We estimate $g(t)$ in the worst case as follows. First, since UPDATEDISTANCE requires a constant number of operations, we judge the time complexity in terms of number of calls to UPDATEDISTANCE. The initialization of $D[S, \tilde{S}]$ requires 2^{v+2k} updates (line 3–4). At each iteration of time i in the main loop (line 5–11), the number of updates is at most 2^{2v+2k}. Therefore, the worst-case complexity of COMPUTEDISTANCE is $g(t) = 2^{v+2k} + 2^{2v+2k}L = O(2^{2(v+k)}L)$. In our experimental search results, we observe that the value of L can be bounded by $L \leq 3v$ for any code. The running time of the algorithm on a 3GHz CPU desktop computer is less than 2ms, which is significantly faster than the naive approach that compares all pairs of paths.

4.3 Search results

In this section, we list the GTCM codes found by our randomization approach (Tables 1 to 3), and discuss their performance in comparison with traditional uniform TCM codes, as well as with a full search approach. The lists are compiled for codes of constraint length up to $v = 10$ and for each pair of original modulation $\mathcal{K}$ and target coded modulation $\mathcal{N}$ in {BPSK, QPSK, 8-PSK, 16-QAM, 64-QAM}. We note that previous work [45] discovered codes for only 1 bit constellation expansion. The asymptotic coding gain β is measured in dB, and the generator matrix G is presented in the standard octal form adopted from [9, 29]. Our symbol mapping of m-PSK constellations is $p(s) = e^{j2\pi s/m}$, where $s = 0 \ldots m-1$ is the transmitted symbol, $j = \sqrt{-1}$. For square m-QAM constellations, our mapping is $p(s) = (x_0 + s_L\Delta_m) + j(y_0 + s_H\Delta_m)$, where Δ_m is the minimum separation in m-QAM, (x_0, y_0) are coordinates of the zero symbol ($s = 0$) located at the bottom-left corner of the constellation, and $s_L = s \bmod (\sqrt{m})$, $s_H = (s - s_L)/\sqrt{m}$ correspond to low-order and high-order bits of s. We note that (1) for some cases of short constraint length, there are no good codes with positive boosting gain; (2) our code search algorithm is independent of symbol mapping, thus can be used to find good generalized TCM codes for any constellation required by the application.

Comparison with uniform codes search. As an example shown in Table 2 for QPSK → 8-PSK, we achieve better codes than the ones in [45] for constraint lengths $v = 6, 8, 10$, which confirms the intuition that better codes can be discovered if the uniform mapping property of the set partitioning rules is relaxed.

Comparison with full search. To verify that the codes discovered using our randomization techniques are actually the best for each category, we also perform a full search for some "small" tuples[1] $(\mathcal{K}, \mathcal{N}, v)$ and compare the results with codes found by RANDOM-

[1]Larger values of $\mathcal{K}, \mathcal{N}$ or v make the *full* search intractable.

BPSK → QPSK			BPSK → 8-PSK			BPSK → 16-QAM			BPSK → 64-QAM		
v	β	G	v	β	G	v	β	G	v	β	G
1	1.76	(3 1)	2	3.72	(5 2 4)	3	3.42	(17 4 5 10)	5	3.94	(63 10 40 47 4 2)
2	3.98	(7 2)	3	4.77	(10 2 17)	4	4.15	(33 10 27 2)	6	4.63	(117 10 40 55 104 100)
3	4.77	(13 4)	4	5.36	(31 4 2)	4	4.15	(33 10 27 2)	6	4.63	(117 10 40 55 104 100)
4	5.44	(35 4)	5	6.02	(10 2 71)	5	5.05	(67 10 55 4)	7	4.91	(227 4 60 371 44 210)
5	6.02	(64 33)	6	6.53	(107 20 12)	6	5.31	(31 100 167 2)	8	5.17	(573 12 52 745 100 362)
6	6.99	(135 56)	7	6.99	(251 102 4)	7	5.8	(212 20 327 40)	9	5.35	(1747 16 344 1277 60 10)
7	6.99	(374 147)	8	7.24	(661 102 30)	8	6.13	(725 14 757 200)	10	5.61	(2473 12 3436 3341 62 6)
8	7.40	(457 142)	9	7.63	(1715 336 400)	9	6.33	(1453 346 1137 30)			
9	7.78	(1312 665)	10	7.78	(3575 1400 14)	10	6.63	(2653 16 3103 774)			
10	8.45	(2175 1256)									

Table 1: BPSK → QPSK/8-PSK/16-QAM/64-QAM TCM codes.

QPSK → 8-PSK			QPSK → 16-QAM			QPSK → 64-QAM		
v	β	G	v	β	G	v	β	G
1	1.12	(3 2 1) (1 0 0)	2	2.55	(3 0 1 1) (1 2 2 1)	4	3.01	(1 0 2 3 0 1) (12 4 6 1 6 6)
2	3.01	(1 5 2) (1 0 0)	3	3.42	(5 2 0 6) (1 0 3 0)	5	3.41	(5 0 2 1 0 4) (12 4 4 15 2 10)
3	3.6	(1 4 2) (2 1 0)	4	3.8	(7 2 2 7) (2 0 5 0)	6	3.68	(1 0 2 3 0 0) (75 30 76 72 2 40)
4	4.13	(13 10 6) (3 1 0)	5	4.15	(1 2 3 0) (37 14 10 2)	7	3.94	(5 0 0 3 1 0) (66 44 20 71 26 6)
5	4.59	(1 4 0) (13 17 4)	6	4.47	(7 16 13 1) (11 16 12 1)	8	4.1	(61 54 34 46 32 34) (15 7 6 3 4 0)
6	5.01	(20 1 12) (3 4 0)	7	5.05	(17 0 4 0) (2 5 25 12)	9	4.26	(12 2 4 7 0 4) (73 62 6 110 16 24)
7	5.01	(7 2 0) (30 75 10)	8	5.05	(10 24 35 12) (25 12 3 15)	10	4.63	(220 46 140 231 102 330) (17 0 4 7 2 12)
8	5.75	(1 6 0) (134 165 42)	9	5.56	(10 24 35 12) (25 12 3 15)			
9	5.75	(311 250 122) (6 1 0)	10	5.56	(116 10 27 70) (23 30 31 5)			
10	6.02	(763 227 376) (7 6 0)						

Table 2: QPSK → 8-PSK/16-QAM/64-QAM TCM codes.

$\mathcal{K}$	$\mathcal{N}$	$T_{\text{full}}(s)$	$T_{\text{random}}(s)$
BPSK	QPSK	1	≈ 0
BPSK	8-PSK	4	≈ 0
BPSK	16-QAM	263	15
QPSK	8-PSK	57	44
QPSK	16-QAM	7101	54

Table 4: Running time comparison for searching codes of constraint length $v = 5$: random search is significantly faster than full search.

CODESEARCH. As we conjectured, the full search does not find any code better than RANDOMCODESEARCH. Yet, RANDOMCODESEARCH is extremely fast (cf. Table 4). The search results indicates that good codes are distributed randomly in the search space, thus randomized searching is a viable approach, and especially useful for large constraint length and high-order modulations.

Asymptotic coding gain. The search results show that with a large enough constraint length there are codes such that in addition to modulation hiding, the resiliency of the system can be boosted up. For example, concealing BPSK in QPSK is about 8.5dB more robust than uncoded BPSK systems.

5. CRYPTOGRAPHIC INTERLEAVING

While encoding the data with the highest-order constellation can hide the modulation, information about the trellis codes still leaks. A powerful adversary can attempt to decode the received stream of samples with all possible codes (which are publicly known) in parallel. The attacker can then infer the rate from the code that has the highest likelihood (i.e., lowest errors). It is necessary to prevent the adversary from being able to try the decoding. Our key idea is to randomize the transitions between the coded symbols. However, we note that naive randomization does not work. For instance, simply applying conventional encryption algorithms on the coded symbols (post-GTCM) will significantly reduce the system performance, as an error occurring during the transmission can spread out to many errors after decryption (by definition of a good encryption algorithm), which would exceed the code's error correcting capability. We therefore propose "Cryptographic Interleaving" as a solution for the code concealing problem. While cryptographic interleaving has been proposed to alleviate the impact of malicious jamming [28], its use for concealing code information, as far as we are aware of, is first identified in this work. Based on cryptographic functions, we design an efficient mechanism to permute the order of GTCM-coded baseband symbols without changing their values.

Crypto-Interleaving Process: In order to prevent an adversary from distinguishing between sequences encoded by different codes, we require: (1) the transmitted symbols should be indistinguishable from a sequence produced by a random code, (2) symbols belonging to different packets are permuted differently, and (3) the user identity is not revealed. We assume a shared key between the transmitter and receiver, which will be used as the seed to generate a secret random interleaving function. For convenience, we assume that the coded symbols produced by the GTCM Encoder can be divided into multiple blocks, each has m symbols, and each frame contains b blocks. A block is identified by the tuple (K, s, i), where K is the shared secret belonging to the communication session, s is the frame number, and i denotes the block index within the frame.

Let's consider a pool of N generated interleaving functions $\mathcal{P} = \{f_0 \ldots, f_{N-1}\}$, where N is the security parameter, and each interleaving function $f_n : I \to I, I = \{0, \ldots, m-1\}, n \in \{0, \ldots N-1\}$ is an invertible index-mapping of symbols within a block. The permutation of symbols in a block (K, s, i) is performed in two steps. First, an interleaving function $f = f_n$ is selected from the pool $\mathcal{P}$ by $n = h_K(s|i) \bmod N$, where h is a key-hashed pseudo-random function (e.g., truncated HMAC-SHA3). Then, the symbol sequence $(y_0, \ldots, y_{m-1})$ is permuted to $(y_{f^{-1}(0)}, \ldots, y_{f^{-1}(m-1)})$.

Generating Crypto-Interleaving Functions: The efficiency of the crypto-interleaving process depends on the computation of the permutation. While a naive solution can precompute the pool of index-mapping, it does not scale with the block size. Moreover, significant memory is required for the precomputation. In our CBM system, we propose an efficient method for generating the interleaving functions. Our technique assumes that the number of symbols per block, m, is a prime. We note that while this constraint can be easily overcome to support various frame size, e.g., by padding, it has the side effect of preventing length-based attacks, which exploit the observations of different frame size to infer the code information. We define the following linear interleaving functions: $f_{A,B}(x) = Ax + B \bmod m$, where $A \in \{1 \,..\, m-1\}, B \in \{0 \,..\, m-1\}$. It is easy to see that any pair (A, B) corresponds to a bijective function $I_{A,B}(x)$ with respect to x; therefore, $f_{A,B}(x)$ is a proper interleaving function (i.e., invertible). The interleaving process is carried out by first generating the coefficients A, B based on the block identifier (K, s, i) by

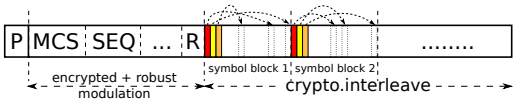
$$A = (h_K(|s|i|0) \bmod (m-1)) + 1, \quad B = h_K(s|i|1) \bmod m.$$

8-PSK → 16-QAM			8-PSK → 64-QAM			16-QAM → 64-QAM		
v	β	G	v	β	G	v	β	G
1	3.11	(1 3 0 1) (0 0 1 0) (1 0 1 0)	3	4.15	(5 1 7 5 3 2) (2 3 3 1 2 0) (1 0 0 0 0 0)	2	3.31	(0 0 1 2 3 0) (1 3 1 1 3 2) (1 0 0 1 0 0) (1 0 0 0 0 0)
2	4.36	(3 3 2 3) (1 0 3 1) (1 0 0 0)	4	4.66	(0 14 1 0 15 4) (2 0 0 1 1 0) (1 0 0 1 0 0)	3	3.31	(1 2 1 1 3 2) (0 3 0 1 0 2) (2 0 1 2 3 2) (0 0 0 1 0 0)
3	5.33	(3 5 1 7) (0 1 3 1) (1 0 1 0)	5	5.12	(37 21 13 24 35 4) (3 1 0 1 0 0) (1 0 0 1 0 0)	4	4.18	(0 5 6 11 10 1) (1 2 0 0 3 0) (1 0 0 1 0 0) (0 0 0 1 0 0)
4	6.12	(5 6 6 4) (4 1 2 3) (1 0 1 0)	6	5.53	(71 75 36 70 53 12) (2 2 0 1 2 0) (1 0 0 1 0 0)	5	4.56	(10 3 4 17 24 14) (1 2 1 2 3 0) (0 0 0 1 0 0) (1 0 0 1 0 0)
5	6.12	(5 4 2 2) (0 3 7 3) (2 0 1 0)	7	5.73	(6 47 4 10 27 12) (3 2 0 1 2 0) (2 2 0 3 2 0)	6	4.91	(63 57 30 12 14 32) (0 2 1 1 3 0) (0 0 0 1 0 0) (1 0 0 0 0 0)
6	6.79	(16 3 6 7) (16 17 13 13) (1 0 1 0)	8	5.91	(124 157 6 270 254 172) (3 0 0 0 0 2) (1 0 0 1 0 1)	7	5.23	(10 3 4 17 24 14) (11 16 5 12 17 14) (0 0 0 1 0 0) (1 0 0 1 0 0)
7	7.37	(23 17 11 17) (2 16 15 10) (1 0 1 0)	9	6.26	(26 21 300 303 272 1) (5 3 0 1 2 0) (1 0 0 1 0 0)	8	5.53	(24 0 15 26 23 0) (31 37 15 5 27 26) (1 0 0 1 0 0) (1 0 0 0 0 0)
8	7.37	(21 20 15 16) (15 5 17 3) (1 0 2 0)	10	6.42	(471 475 236 670 653 12) (6 6 0 1 2 0) (1 0 0 1 0 0)	9	5.81	(73 177 42 123 25 55) (10 7 4 2 12 1) (1 0 0 1 0 0) (1 0 0 0 0 0)
9	7.37	(77 336 10 332) (2 1 5 1) (1 0 1 0)				10	5.81	(36 737 302 641 151 317) (4 7 2 6 6 1) (1 0 0 1 0 0) (1 0 0 0 0 0)
10	7.37	(15 35 33 5) (13 16 10 15) (16 10 0 15)						

Table 3: 8-PSK → 16-QAM/64-QAM TCM codes.

The interleaving function f is selected as $f = f_{A,B}$. The symbol sequence $\{y_0, \ldots, y_{m-1}\}$ is accordingly permuted to $\{y_{f^{-1}(0)}, \ldots, y_{f^{-1}(m-1)}\}$.

Header Format and Encoding: Since the interleaving processing on the coded symbols of the user data involves using not only the secret key K, but also the packet number s and block index i, the transmitter needs to embed this information along with the rate information into the transmitted frame.



At the packet beginning, the preamble P assists with the frame detection and synchronization at the receiver. The MCS field identifies the modulation and coding scheme for the payload. The packet number required for cryptographic interleaving is specified by SEQ. The R field stores a random number generated per packet by the transmitter. The frame header is encrypted by $E_K(MCS|SEQ|\ldots|R)$ using AES encryption E with the shared secret key K. The header is encoded by a public robust coding scheme and along with the preamble is modulated by a public robust modulation.

Security: Since the interleaving functions are generated using a key-hashed pseudorandom function h with secret key K applied on the packet number s and block index i, the coefficients A and B are indistinguishable [3], thus the interleaved symbol sequences are also indistinguishable.The header is also semantically secure due to the use of random R with AES encryption.

6. SYNCHRONIZATION MECHANISMS

As we will show in Section 7, the simulation results exhibit a significant gain in system robustness while the rate information is concealed from the adversary. However, when evaluating our CBM scheme over a real radio system with a standard set of synchronization mechanisms, we experience a substantial performance loss. Chief among the mechanisms impacted by the imperfections of the radio front end are the frequency and phase correction. At low SNR, such imperfections impact high order modulation much more severely than low order modulation. As in many other communication systems, synchronization with transmitter is required at the receiver in order to decode the data. Traditional synchronization techniques for symbol timing, frequency and phase recovery are realized by means of phase locked loop (PLL) circuits [22, 40]. However, in the context of our work, where the channel SNR can be very low due to adversarial interference, conventional methods perform poorly, making synchronization mechanisms bottleneck of the system. To make our CBM system robust, we developed a set of new efficient digital signal processing algorithms for coarse and fine frequency and phase offset correction relying on an integrated process of estimations based on preambles and iterative soft decoding. Our algorithms also exploit the fact that today's radio receivers can have ample memory to store a whole packet, and many standards already require multi-pass iterative soft-decoding, e.g., LDPC decoding in DVB-S2, IEEE 802.11n/ac. In Section 7, we show that our techniques achieve significantly better performance than traditional solutions. It is also worth mentioning that the impact of asynchronization attacks, which focus jamming energy on the beginning of each frame to destroy the synchronization information, is also alleviated in our system by using a long preamble with good correlation properties (cf., Section 6.2).

6.1 Overview of Transmitter and Receiver

We designed and implemented our transmitter and receiver on the USRP N210 SDR [11] using GNU Radio [6]. The transmit and receive chains of our CBM system are depicted in Figure 5.

Transmitter: The key components of the transmitter are the GTCM Encoder and Cryptographic Interleaving blocks. On transmission of a packet, the binary payload is encoded with an appropriate trellis code and target modulation (Tables 1 to 3) by the GTCM Encoder to produce coded complex symbols. The sequence of symbols is then permuted using the Cryptographic Interleaving block according to the shared key between the two parties. The packet modulating process is completed by prepending the payload with the header and preamble. Finally, it is resampled by the root-raised cosine TX filter before transmission by the RF front end.

Receiver: As in many communication systems, our receiver first obtains the sequence of symbols from the RF front end after they are preprocessed with the Automatic Gain Control (AGC) and symbol timing synchronizer to stabilize the attenuated input signal and lock to the receiver sampling clock.

The key to improving the robustness of our CBM system to overcome the RF front end imperfections consists of improving the accuracy of frequency offset and phase correction. Our mechanisms start with the Frame Synchronizer. First, we estimate the coarse frequency offset based on the preamble of each received packet. The header and payload symbols are then corrected with the estimates. To improve the estimation accuracy, we employ a phase locked loop (PLL) combined with a soft pre-decoding of the packet. This integration results into a 2-pass decoding process. The feedback from the soft pre-decoding re-encoding loop is applied to the PLL to improve the correction. Finally, the corrected sequence passes through the Cryptographic Deinterleaver and GTCM Decoder to recover the original data. In the following subsections, we discuss in details our efficient algorithms of those mechanisms.

6.2 Frame Synchronization

The Frame Synchronizer uses the preamble for both frame detection and frequency offset estimation. The principle of our synchronization mechanisms is to analyze the phase-difference sequence of the received symbols. While our technique for frequency offset estimation partly shares similarity with previous work [24], we improve the estimation by averaging over multiple packets. More importantly, we can identify the frame and estimate the phase offset at the same time, and fine-tune the estimation with feedback from the soft decoding process. We consider the received symbol

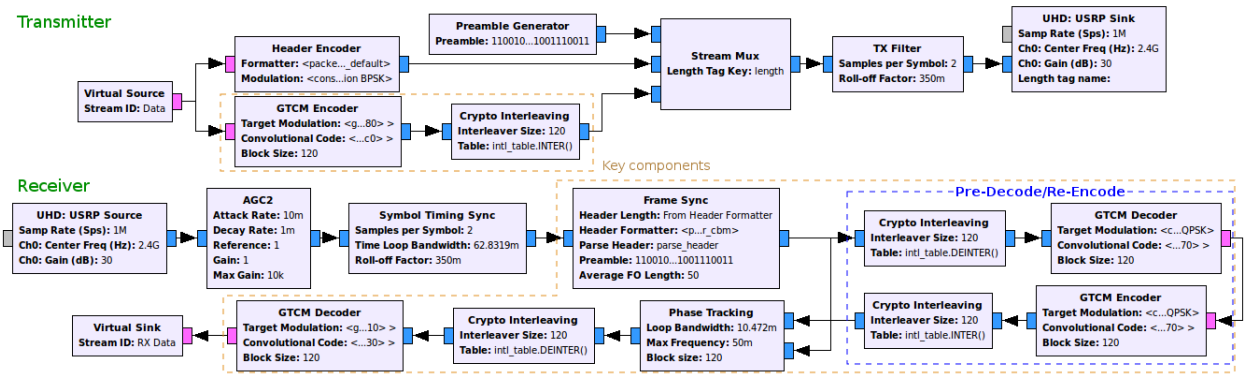


Figure 5: Transmitter and Receiver block diagram

$y_i = x_i e^{j(\theta i+\phi)} + w_i$ at discrete sampling time i, after the signal has been normalized by the AGC, and the symbol sampling period has been synchronized with the receiver clock. The transmitted symbol x_i is distorted by the unknown frequency offset θ, phase offset ϕ due to the mismatched clock between the transmitter and receiver, and interference w_i. The unknown parameters θ and ϕ are considered as constants during a short period of preamble, and zero-mean variables during the packet transmission period.

Frame Detection: Let $\{p_1, \ldots, p_L\}$ denote the complex valued preamble, and $\Delta p_i = p_{i+1} p_i^*$ be the phase-difference indicator between two adjacent preamble symbols. The transmitted stream consists of multiple frames: $\{x_i\} = \{p_1, \ldots, p_L, d_{1,1}, \ldots, d_{1,m}, \ldots, p_1, \ldots, p_L, d_{s,1}, \ldots, d_{s,m}, \ldots\}$, where $d_{s,j}$ denotes the s-th frame's j-th data symbol. To detect the start of each transmitted frame in the received signal, we first compute the phase-difference sequence of received symbols $\Delta y_i = y_{i+1} y_i^* = x_{i+1} x_i^* e^{j\theta} + z_i$, where z_i represents the phase-difference with interference. We then match $\{\Delta y_i\}$ with $\{\Delta p_i\}$, i.e., computing the cross-correlation

$$C = \sum_{i=1}^{L-1} \Delta y_i \Delta p_i^* = \sum_{i=1}^{L-1} x_{i+1} x_i^* \Delta p_i^* e^{j\theta} + \sum_{i=1}^{L-1} z_i \Delta p_i^*.$$

We select the preamble as a maximum length sequence such that $\sum p_i x_i^* = L$ only if $\forall i, x_i = p_i$. When the preamble is present in the received signal, i.e., $x_i = p_i$, we obtain $C \approx (\sum_i^{L-1} |\Delta p_i|^2) e^{j\theta}$. When it is not present, C becomes small due to uncorrelation between $\{p_i\}$ and $\{y_i\}$. Our experimental evaluation suggests that the frame be best detected, if $|C| \geq \alpha \sum_{i=1}^{L-1} |\Delta p_i|^2$ with $\alpha = 0.8$.

Frequency Offset Estimation: Once the frame is detected, the frequency offset is immediately estimated from the cross-correlation: $\tilde{\theta} = \angle C \approx \theta$. Under low SNR conditions, $\tilde{\theta}$ may not closely approximate the actual θ. Instead, we compute the average $\hat{\theta} = E[\tilde{\theta}]$ over multiple packets, and use $\hat{\theta}$ for succeeding processing.

Phase Offset Estimation: Knowing the frequency offset $\hat{\theta}$, we estimate the phase offset by correcting the received preamble y_i with $\hat{\theta}$: $\hat{y}_i = y_i e^{-j\hat{\theta} i}$, and computing the cross-correlation with the expected preamble p_i as: $A = \sum_{i=1}^{L} \hat{y}_i p_i^* \approx \sum_{i=1}^{L} p_i e^{j(\theta-\hat{\theta})i+\phi} p_i^* \approx \sum_{i=1}^{L} |p_i|^2 e^{j\phi}$. The phase offset $\hat{\phi}$ is estimated as $\hat{\phi} = \angle A \approx \phi$.

6.3 Phase Tracking

While preamble based frequency and phase offset estimation can provide accurate estimates at the frame beginning, the environment variations make these initial estimates deviated from the actual values over time, especially when reaching the end of the frame. To over come this issue, standard phase tracking mechanisms usually deploy decision-aided methods, which rely on the distance between the received symbols and the reference constellation points. BPSK modulation has less symbol errors and therefore the phase tracking works quite well. For higher order modulations, the density of the constellation makes the estimation of the phase errors harder at low SNR, as incorrect constellation points are guessed more frequently. These errors accumulate and end up exceeding the error correction capability of the GTCM code. This partially explains for low SNR scenarios why many current standards still rely on low order modulations instead of coded high order modulations. In pilot-aided synchronization systems such as OFDM, low SNR can result in loss of orthogonality of the carriers, preventing the receiver to recover the signal's phase and amplitude.

To improve the resiliency of our system, we first add a PLL logic in the Frame Synchronizer to keep track of minor changes of the frequency and phase. The loop bandwidth of the PLL is set to a reasonably small value in order to be sufficient to track the small variations, but not to be disturbed by interference.

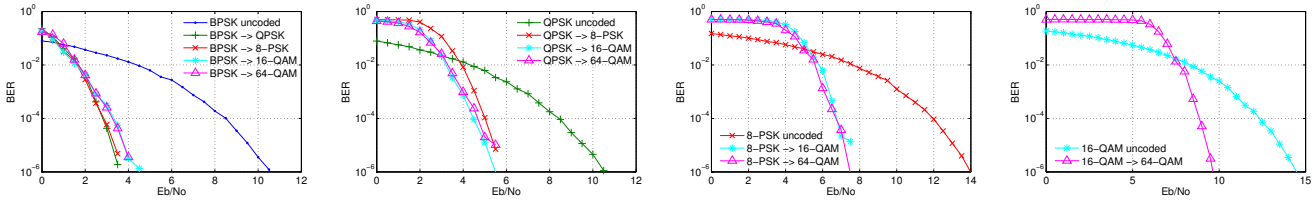
The key technique to improve the estimation accuracy, consists of a pre-decoding and re-encoding mechanism, in which we first decode the phase-corrected symbols (by running them one first time through the Cryptographic Deinterleaver and GTCM Decoder), then re-encode the resulted binary data back to complex symbols. Now by comparing the re-encoded symbols (which have few symbol errors) and the previously received signal, we can rerun the phase tracking mechanism overcoming synchronization mistakes of the first pass. During the second-pass phase tracking, we carefully skip all the symbols for which there is a discrepancy between what was received and what was decoded. We observe that this significantly reduced the phase tracking errors. This frequency/phase corrected sequence of symbols is then sent to the Cryptographic Deinterleaver and GTCM Decoder for the final decoding.

7. EVALUATION

We report on the evaluation of our CBM system with both simulation results in MatLab and experimental results in our USRP N210 SDR testbed. We use the bit error rate (BER) and normalized signal-to-noise ratio E_b/N_0 as metrics for the system robustness.

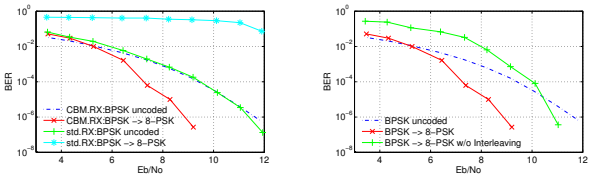
7.1 Simulation Results

First, we assess the robustness of our codes found in Section 4 by simulation in MatLab to avoid impact of imperfect RF front ends such as frequency offset or phase noise. We simulate the transmission of packets encoded by the best TCM codes (constraint length



(a) Conceal & Boost BPSK (b) Conceal & Boost QPSK (c) Conceal & Boost 8-PSK (d) Conceal & Boost 16-QAM

Figure 6: Simulation results: Resiliency Boost of Coded over Uncoded Modulations



(a) Synchronization: CBM Receiver vs. Standard Receiver.

(b) Robustness with and without Cryptographic Interleaving.

Figure 7: Performance of CBM's Synchronization and Cryptographic Interleaving mechanisms.

$v = 10$) for each pair of original modulation $\mathcal{K}$ and target modulation $\mathcal{N}$. The simulated noise is additive white Gaussian.

At BER $= 10^{-6}$, Figure 6a shows that, in addition to hiding the rate, our codes provide up to 7dB gain over uncoded BPSK modulation when concealing it into any higher-order modulation. Compared to the modulation level encryption technique proposed in recent related work [41] whose performance degrades by about 1.2dB for hiding BPSK modulation in 64-QAM modulation, we gain up to 8.2dB. In Figures 6b to 6d, we also achieve significant improvements when concealing QPSK, 8-PSK, and 16-QAM to higher-order modulations. It can be observed that when 64-QAM modulation is used for rate concealing, we obtain at least 5dB resiliency over any original uncoded modulation.

7.2 Experimental Results

As noted by previous work [45], the frequency offset and phase noise, due to channel variations and imperfect RF front ends, can severely reduce the decodability of TCM codes. To assess our system in realistic conditions, we implemented our solution in our USRP N210 SDR testbed to evaluate our system performance. The setup consists of one transmitter, one receiver, and one jammer. The experiments were both carried through an RF cable and a combiner adding the TX signal to the jammer signal (for a precise control of the SNR), and over the air as a second check.

Impact of Synchronization Mechanisms: As discussed in Section 6, the low SNR conditions prevent traditional synchronization techniques from performing well. Figure 7a shows the performance comparison between two systems: (1) one uses the standard synchronization techniques with band-edge filter combined with phase locked loop circuits [21, 22], (2) CBM receiver with our synchronization mechanisms. We observe that the coded modulation with standard synchronization techniques performs worse than the uncoded modulation due to low SNR. With our 2-pass synchronization, the accuracy of frequency and phase offset correction is much improved, resulting in the resiliency boost of the system.

Effect of Interleaving: While the main goal of Cryptographic Interleaving is to conceal the underlying modulations and codes from the adversary, it has a side effect of increasing the system robustness by scattering the burst errors over the whole block. To show the contributions of Cryptographic Interleaving in terms of resiliency boost, we compare the performance of our CBM system with and without interleaving. Figure 7b shows that interleaving increases the performance gain by roughly 2dB. We note that without our synchronization mechanism, the interleaving does not help.

CBM vs. Uncoded: Now we evaluate our CBM system in comparison with the uncoded system. We conduct the experiments for concealing any modulation $\mathcal{K}$ under any higher-order modulation $\mathcal{N}$ from the set of 5 different supported modulations. The results are shown in Figures 8a to 8d. First, we observe that in the real world environments, our system resiliency drops roughly $4 - 6$dB due to the unpleasant frequency and phase offset. Yet, comparing to the uncoded system, we obtain up to 4dB gain in resiliency while simultaneously concealing the rate information.

8. CONCLUSION AND DISCUSSION

We proposed a solution to the problem of hiding the rate of a communication while simultaneously increasing its robustness to interference. To the best of our knowledge, this is the first system that can achieve this goal. Our approach relies on algorithms for discovering new General TCM codes, and a cryptographic interleaving scheme. These algorithms include new efficient techniques to determine the free distance of non-uniform TCM codes. We explicitly derived 85 codes for upgrading any modulation in {BPSK, QPSK, 8-PSK, 16-QAM, 64-QAM} into any higher order modulation. These are the best codes among uniform and non-uniform TCM codes specifically designed for coded modulation and that conceal the underlying rate information. The GTCM codes and Crypto-Interleaving are complemented by new frequency correction and phase tracking techniques. We demonstrate analytically, and through simulations and experimentation, that beyond achieving rate-hiding, an order of magnitude improvement of energy efficiency is achieved in comparison with recent related work. The proposed solution is easily deployable in software defined radios. Our implementation source code is available at [47]. An interesting avenue of future research is to explore generalizations of turbo and LDPC codes to higher order modulations with an integrated frequency and phase correction mechanism.

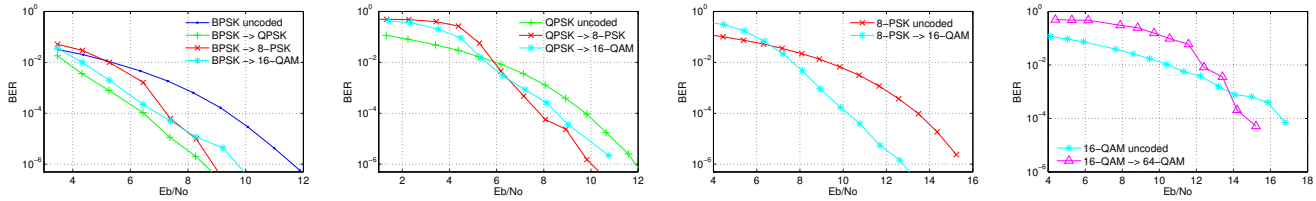
Acknowledgement. This material is based upon work supported by the National Science Foundation under Grant CNS-1409453.

References

[1] B. Awerbuch, A. W. Richa, and C. Scheideler. A jamming-resistant MAC protocol for single-hop wireless networks. PODC'08, 2008.

[2] E. Bayraktaroglu, C. King, X. Liu, G. Noubir, R. Rajaraman, and B. Thapa. On the performance of IEEE 802.11 under jamming. In *IEEE INFOCOM*, 2008.

[3] M. Bellare. New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In C. Dwork, editor, *Advances in Cryptology* -



(a) Conceal & Boost BPSK (b) Conceal & Boost QPSK (c) Conceal & Boost 8-PSK (d) Conceal & Boost 16-QAM

Figure 8: Experimental results: Resiliency Boost of Coded over Uncoded Modulations

CRYPTO 2006, volume 4117 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2006.

[4] M. A. Bender, M. Farach-Colton, S. He, B. C. Kuszmaul, and C. E. Leiserson. Adversarial contention resolution for simple channels. SPAA '05, pages 325–332, New York, NY, USA, 2005. ACM.

[5] J. C. Bicket. Bit-rate selection in wireless networks. Master's thesis, Massachusetts Intitute of Technology, 2005.

[6] E. Blossom. GNU Radio: Tools for Exploring the Radio Frequency Spectrum. *Linux J.*, 2004(122), June 2004.

[7] I. Broustis, K. Pelechrinis, D. Syrivelis, S. V. Krishnamurthy, and L. Tassiulas. Fiji: Fighting implicit jamming in 802.11 wlans. In *Security and Privacy in Communication Networks*. 2009.

[8] A. Cassola, W. Robertson, E. Kirda, and G. Noubir. A practical, targeted, and stealthy attack against wpa enterprise authentication. NDSS, 2013.

[9] J.-J. Chang, D.-J. Hwang, and M.-C. Lin. Some extended results on the search for good convolutional codes. *Information Theory, IEEE Transactions on*, 43, 1997.

[10] Y. Chen, W. Xu, W. Trappe, and Y. Zhang. *Securing Emerging Wireless Systems: Lower-layer Approaches*. Springer, 2009.

[11] Ettus Research. Universal Software Radio Peripheral.

[12] FCC. FCC enforcement bureau steps up education and enforcement efforts against cellphone and gps jamming, 2013.

[13] FCC. FCC fines jammers, 2013.

[14] FCC. Marriott hotels fined $600,000 by FCC for jamming Wi-Fi hotspots, October 2014.

[15] K. Firouzbakht, G. Noubir, and M. Salehi. Superposition coding in an adversarial environment. CISS, March 2011.

[16] K. Firouzbakht, G. Noubir, and M. Salehi. On the capacity of rate-adaptive packetized wireless communication links under jamming. WISEC '12. ACM, 2012.

[17] K. Firouzbakht, G. Noubir, and M. Salehi. On the performance of multi-layer superposition coding scheme under constrained jamming. CISS, March 2013.

[18] K. Firouzbakht, G. Noubir, and M. Salehi. On the performance of adaptive packetized wireless communication links under jamming. *Wireless Communications, IEEE Transactions on*, 2014.

[19] K. Firouzbakht, G. Noubir, and M. Salehi. Packetized wireless communication under jamming, a constrained bimatrix game. GLOBECOM, Dec 2014.

[20] S. Gilbert, R. Guerraoui, and C. Newport. Of malicious motes and suspicious sensors: On the efficiency of malicious interference in wireless networks. In *OPODIS*, 2006.

[21] F. Harris. Let's assume the system is synchronized. In *Globalization of Mobile and Wireless Communications*. Springer, 2011.

[22] F. J. Harris. *Multirate Signal Processing for Communication Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[23] T. Jin, G. Noubir, and B. Thapa. Zero pre-shared secret key establishment in the presence of jammers. MobiHoc, 2009.

[24] S. Kay. A fast and accurate single frequency estimator. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, Dec 1989.

[25] J. Kim, S. Kim, S. Choi, and D. Qiao. CARA: Collision-aware rate adaptation for IEEE 802.11 WLANs. In *INFOCOM*, 2006.

[26] M. Lacage, M. H. Manshaei, and T. Turletti. IEEE 802.11 rate adaptation: a practical approach. In *Proceedings of ACM MSWiM*, 2004.

[27] M. Li, I. Koutsopoulos, and R. Poovendran. Optimal jamming attacks and network defense policies in wireless sensor networks. In *INFOCOM*, 2007.

[28] G. Lin and G. Noubir. On link layer denial of service in data wireless lans. *Wiley Journal on Wireless Communications and Mobile Computing*, May 2005.

[29] S. Lin and D. J. Costello. *Error Control Coding*. 2 edition, 2004.

[30] A. Liu, P. Ning, H. Dai, Y. Liu, and C. Wang. Defending DSSS-based broadcast communication against insider jammers via delayed seed-disclosure. ACSAC'10.

[31] MadWifi-Project. Bit-rate selection algorithms, 2012.

[32] D. Nguyen, C. Sahin, B. Shishkin, N. Kandasamy, and K. R. Dandekar. A real-time and protocol-aware reactive jamming framework built on software-defined radios. SRIF '14, pages 15–22, New York, NY, USA, 2014. ACM.

[33] G. Noubir, R. Rajaraman, B. Sheng, and B. Thapa. On the robustness of IEEE 802.11 rate adaptation algorithms against smart jamming. ACM WiSec '11, pages 97–108, 2011.

[34] Onoe. MadWifi rate control, 2011.

[35] C. Orakcal and D. Starobinski. Jamming-resistant rate control in Wi-Fi networks. In *IEEE GLOBECOM*, 2012.

[36] C. Orakcal and D. Starobinski. Jamming-resistant rate adaptation in wi-fi networks. *Performance Evaluation*, 75-76(0):50–68, 2014.

[37] K. Pelechrinis, I. Broustis, S. V. Krishnamurthy, and C. Gkantsidis. Ares: an anti-jamming reinforcement system for 802.11 networks. In *Proceedings of ACM CoNEXT*, 2009.

[38] K. Pelechrinis, M. Iliofotou, and S. V. Krishnamurthy. Denial of service attacks in wireless networks: The case of jammers. *Communications Surveys & Tutorials, IEEE*, 2011.

[39] R. Poisel. *Modern communications jamming principles and techniques*. Artech House Publishers, 2011.

[40] J. G. Proakis and M. Salehi. *Digital Communications*. 5 edition, 2007.

[41] H. Rahbari and M. Krunz. Friendly CryptoJam: A Mechanism for Securing Physical-layer Attributes. WiSec '14, 2014.

[42] B. Schneier. Fake cell phone towers across the US, Sep. 2014.

[43] M. Strasser, C. Popper, S. Capkun, and M. Cagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *IEEE SP*, 2008.

[44] P. Tague, D. Slater, G. Noubir, and R. Poovendran. Linear programming models for jamming attacks on network traffic flows. In *WiOpt*, 2008.

[45] G. Ungerboeck. Channel coding with multilevel/phase signals. *Information Theory, IEEE Transactions on*, 28(1):55–67, Jan 1982.

[46] vMonitor and ICL. SCADA wireless systems, 2011.

[47] T. D. Vo-Huu and G. Noubir. CBM source code. http://www.ccs.neu.edu/home/noubir/projects/cbm.

[48] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *Proceedings of ACM SIGCOMM*, 2009.

[49] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders. Short paper: reactive jamming in wireless networks: how realistic is the threat? In *Proceedings of ACM WiSec*, 2011.

[50] S. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *ACM MobiCom*, 2006.

[51] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *IEEE Network*, 20(3):41–47, 2006.

[52] W. Xu, W. Trappe, and Y. Zhang. Defending wireless sensor networks from radio interference through channel adaptation. *ACM Transactions on Sensor Networks*, 4, September 2008.