Dynamic Reconfigurable Puncturing for Secure Wireless Communication

Liang Tang*, Jude Angelo Ambrose†, Akash Kumar*, Sri Parameswaran†
*Department of Electrical and Computer Engineering, National University of Singapore, Singapore
to.liang.tang@gmail.com, akash@nus.edu.sg
†School of Computer Science and Engineering, University of New South Wales, Australia
{ajangelo,sridevan}@cse.unsw.edu.au

***Abstract*—The ubiquity of wireless devices has created security concerns on the information being transferred. It is critical to protect the secret information in every layer of wireless communication to thwart any type of attacks. A dynamic reconfigurable puncturing based security mechanism, named *RePunc*, is proposed in this paper to provide an extra level of security at the physical layer. *RePunc* utilizes the puncturing feature of Forward Error Correction (FEC) to insert the secure information in the punctured positions of the standard information encoded data. The punctured patterns are dynamically changed and passed as a secret key from the sender to the receiver. An eavesdropper will not be able to detect the transmission of the secure information since the inserted secure information will be processed as channel noise by the eavesdropper's receiver. However, the rightful receiver will be able to successfully decode the secure packets by knowingly differentiating the secure information and the standard information before the FEC decoding. A case study of *RePunc* implementation for WiFi communication is presented in this paper, showing the extreme high security complexity with low hardware overhead.**

I. Introduction

The need for wireless technology to perform secure transactions has significantly increased. It is typical to encrypt the data before transmission at the MAC layer or at the application layer, using cryptographic algorithms such as AES, ECC, etc. However, the open nature of the wireless channel still allows the unintended receiver (or eavesdropper) to receive this encrypted information correctly without any notice. The information/data transferred through the wireless medium is referred to as *physical message* from this point onwards. Successful capture of the physical message will allow attackers to leisurely explore security vulnerabilities in the encryption, and use these vulnerabilities to extract the secure information. For example, eavesdropping on Bluetooth links [1] has been quite successful in recent years.

Existing physical layer security techniques attempt to reduce the chances of the physical message being captured by the adversary. Thus, any possibilities of offline attacks would be completely refrained. One of the methods used is frequency hopping. By hopping channels according to a pattern known to only the transmitter and the legitimate receiver, an eavesdropping adversary can be confounded. Frequency hopping comes under the broader umbrella of spreading. Spreading [2] in wireless protocols has an inherent ability to hide the physical message. However, it is vulnerable to cross correlation analysis on the spreading code [3]. Custom secret modulation mapping schemes were attempted in [4] as another physical layer security mechanism. Even though this would confuse the adversary, an exhaustive attack on the physical message to find the correct modulation mapping scheme is not difficult, since there are very few modulation mapping schemes.

In current literature, there exist proposals to change the physical layer signal processing parameters associated with the spreading [5], modulation mapping [4], puncturing [6], and convolutional coding [7] signal processing function blocks for security applications. To the best of our knowledge, for the first time, we propose a method to dynamically change the puncturing pattern on standard packets' (we refer to standard packets as the packets which do not require security) convolutional encoded bits, called *codeword* bits, and insert the secure codeword bits into the puncturing pattern positions, in order to make the secure packet invisible to eavesdroppers. Although the standard packets can be correctly received at the rightful receiver and eavesdropper, such a dynamic secure codeword insertion will make the exhaustive hacking analysis harder on the secure packets.

II. Related Work

Previous techniques to provide security at the physical layer for wireless communication can be categorized into four: 1), spread spectrum based techniques [2], [5], [8]; 2), modulation mapping constellation based techniques [4]; 3), eavesdropper channel degrading techniques [9]; and 4), reconfigurable wireless signal processing circuit based techniques [6], [7].

Spread spectrum techniques are typically utilized to disperse the information over a range of frequencies, thereby increasing the transmission bandwidth while reducing the interference from the other signals. Realizing the actual information bit using a secret spreading pattern is an effective way to hide the information over wireless channels. Direct sequence spread spectrum (DSSS) [5], Frequency hopping spread spectrum (FHSS) [2] and Hybrid Spread Spectrum (HSS) [8], combining FHSS and DSSS, are used for physical level secure communication.

The authors in [4] present a modulation mapping constellation approach, which uses a secret custom constellation mapping between the intended transmitter and receiver. The mapping relationship between the constellation points (i.e., real and imaginary) are customised in [4]. There are $M!$ variations for an M-symbol modulation constellation. However the value of M is restricted by the modulation scheme. For example, M is only four in the QPSK modulation. Thus, the modulation mapping constellation approach in [4] can be easily compromised.

Phased array enables secure wireless transmissions by enhancing signal transmission in the direction of the desired receiver, while degrading the signal in most of the other directions. Such directional transmission requires an adversary to use sophisticated sensitive receivers to recover the suppressed signals [9].

A reconfigurable puncturing pattern is proposed to encrypt the Turbo encoding data in [6]. A pseudorandom sequence is used to generate a customized puncturing pattern, which is only shared between the transmitter and the rightful receiver. Since the eavesdropper does not know the puncturing pattern, the received packet cannot be successfully decoded. However, the original uncoded information data are transmitted directly over-the-air and the puncturing pattern only affect the padding parity bits in Turbo encoding [6]. Thus, it is trivial to decrypt, especially in a good SNR channel condition, where the original uncoded information could be decoded without the help of the padding parity bits.

978-3-9815370-4-8/DATE15/©2015 EDAA

All the previous mentioned approaches send the secure information packets with updated physical layer signal processing parameters. An expert eavesdropper can recognize if these secure approaches are applied, especially when the received Packet Error Rate (PER) is abnormally high while the channel condition is good.

We propose to use a reconfigurable design for puncturing to insert the convolutional encoded secure packet codeword into the standard packet codeword. The standard packet would be successfully received in the both legitimate receiver and eavesdropper. Thus, the eavesdropper cannot realize the transmission of the secure information and the further hacking efforts can be refrained.

The attacking complexity on our methodology is high because of the large number possible puncturing parameters in our methodology. The details are given in Section III-C.

III. *RePunc* Methodology

In wireless communication, puncturing is an effective technique to improve the data rate by eliminating certain bits of an Forward Error Correction (FEC) encoded data packet before transmitting. The de-puncturing circuit in the receiving path inserts dummy bits at the punctured positions of the codeword to cover the punctured bits, before the FEC decoding. Due to the FEC decoder's error correction capability, the punctured data packet still can be decoded correctly. We present the puncturing procedure for WiFi as a case study in this paper. The puncturing works tightly with the convolutional encoding in WiFi. The convolutional encoding is used in WiFi as the FEC mechanism, which converts each 1-bit information symbol into a 2-bit coded symbol, called *codeword*, where the information symbol and coded symbol ratio $1/2$ is the coding rate (CR) [10]. To improve the data rate in WiFi, two puncturing patterns, $CR = 3/4$ and $CR = 2/3$, are introduced in WiFi specification [10].

A. *RePunc* Secure Communication

The proposed *RePunc* uses the puncturing signal processing feature to insert the secure packet codeword into the standard packet codeword. Instead of eliminating the punctured bits in WiFi, we propose to replace the standard packet punctured codeword bits by the secure codeword bits. The operations of two *RePunc* packets, one has $CR = 3/4$, another has $CR = 3/5$, are illustrated in Fig. 1. The Viterbi decoders at both the rightful receiver and the eavesdropper can decode the standard packet correctly. Thus, the eavesdropper would think that the all monitored data packets are properly collected, unaware of the transmission of the secure packets. An extra buffer is needed at the rightful receiver to store the secure codeword bits according to the defined puncturing pattern. Once a whole secure packet is stored in the buffer, a Viterbi decoding can be performed to decode the secure packet information bits.

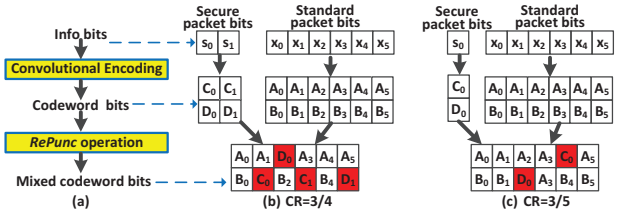

Fig. 1. RePunc Secure Communication

Even without any evidence, a smart eavesdropper may suspect that secure packets are transmitted by the *RePunc*, and may try to abstract the secure packet codeword bits according to the estimated puncturing patterns. Thus, the number of different puncturing patterns in *RePunc* should be large enough to make the exhaustive eavesdropper's puncturing pattern estimation nearly impossible.

In theory, the punctured positions for the secure codeword can be in any position of the whole standard packet codeword to increase the number of possible puncturing patterns. Although there are some forbidden puncturing patterns as these patterns will generate errors at the Viterbi decoding for the standard packet, a large number of different puncturing patterns still exists.

The transmitter and receiver exchange a seed of the secret key random generator offline (a common standard practice in wireless applications [11]). We propose the secret key S, which is the definition of the puncturing pattern to insert the secure packet codeword, for our solution as shown in Equation 1. L is the length of the *puncturing unit block*, or PUB, which is the number of information bits for a minimum codeword block whose puncturing pattern is repeated in the whole packet. For example, WiFi CR $3/4$ in [10] has the PUB length of three, since the puncturing patterns are repeated for each three source information bits, or six codeword bits. In *RePunc*, the L can be large, where the entire packet can be treated as a single PUB. Alternatively, the L can be small, such as two or three as the WiFi specification. N is the number of puncturing bits in a single PUB. For example, WiFi $CR = 3/4$ has $N = 2$ since two bits are punctured in each PUB. P defines the positions of the punctured bits, from p_0, p_1 to p_{N-1}, for N punctured bits.

$$S = \{L, N, P\}; P = \{p_0, p_1, p_2, p_3, ..., p_{N-1}\} \quad (1)$$

The larger L can have larger N, which has higher security complexity. However, the *RePunc* hardware implementation overhead will increase as a result. We let the user to decide the maximum L with the trade-off between the security complexity and the hardware implementation overhead.

B. *RePunc* Secure Communication Algorithm

To effectively utilize *RePunc*, we propose to dynamically change the puncturing pattern for the secure packet codeword insertion. The secure packet transmission throughput is defined by the number of secure packet information bits transmitted in one second, it can be derived from the *RePunc* standard packet CR, which is referred as *RePunc* CR from this point onwards. The secure packet transmission throughput is proportional to the standard packet Packet Error Rate (PER). To keep the same receiving performance as WiFi, the highest *RePunc* CR is set to $3/4$, i.e., the highest CR in WiFi. Although the standard packets are not the target of the *RePunc* mechanism, a smart eavesdropper may be alerted by the high PER.

The flow of the *RePunc* transmitter and receiver is depicted in Algorithm 1. Aligning with the standard practice in wireless communication, we allow the transmitter and the receiver to use the same pseudo random number generator (*PRNG*) which will generate the same random number from a given seed (*SD*). The *SD*, the period for security key updating M, the maximum hardware supported PUB size L_m, and the user defined PER degradation threshold T, are exchanged offline, as well as the *PRNG* between the transmitter and receiver. The seed *SD* will be utilized to initialize the working seed $SD\prime$ in both transmitter and receiver to generate the security key S, defined in Equation 1. Iterations after that will align at both ends to create the same pseudo random numbers at every iteration.

For every M standard packet, the PER_gap, which is the PER difference between the secure and standard packets, is checked. The RX sets $flagT$ to TRUE when the PER_gap is greater than the T, otherwise sets FALSE. The $flagT$ is sent back to the transmitter to decide the new security key S by the

S_gen function. When the $flagT$ is FALSE, the L and N are generated by the *SEL_0* and *SEL_1* functions directly. When the $flagT$ is TRUE, the new generated L must not smaller than the previous value, and the new N must be not greater than the previous one. Thus, the *RePunc* CR can be reduced and the PER performance can be improved. Once the L and N are decided, the positions of N secure bits can be generated by the function *SEL_2*.

Algorithm 1: *RePunc* Secure Communication

Offline: exchange SD, M, L_m, T;
Offline: decide *PRNG*;
Transmitter (TX):
$SD\prime$ = SD;
repeat
  for every M standard packets;
    Get $flagT$ from receiving;
    S = *S_gen*($SD\prime$, L_m, $flagT$);
    $SD\prime = SD\prime + 1$;
    SET(L, N, p_0, p_1, ..., p_{N-1});
  RePunc secure bits insertion;
  packet sending;
until *transmission end*;
function: **S_gen**($SD\prime$, L_m, $flagT$)
  r = *PRNG*($SD\prime$);
  if $flagT$ is TRUE;
    L = MAX(*SEL_0*(L_m, r), L);
    N = MIN(*SEL_1*(L, r), N);
  else;
    L = *SEL_0*(L_m, r);
    N = *SEL_1*(L, r);
  $for(i = 0; i < N, i++)$;
    p_i = *SEL_2*(L, r, i);
  S = L, N, p_0, p_1, ..., p_{N-1}
Receiver (RX):
$SD\prime$ = SD;
repeat
  for every M standard packets;
    if $PER\_gap >$ T;
      Set $flagT$ to TRUE;
    else;
      Set $flagT$ to FALSE;
    send $flagT$ to TX;
    S = *S_gen*($SD\prime$, L_m, $flagT$);
    $SD\prime = SD\prime + 1$;
    SET(L, N, p_0, p_1, ..., p_{N-1});
  store the secure bits base on pattern;
  standard packet decoding;
  if the whole secure packet is stored;
    secure packet decoding;
until *transmission end*;

C. RePunc Security Complexity

The *RePunc* security complexity depends on the PUB size. In WiFi, the maximum length of the data packet can be 4096 bytes, i.e., 32768 bits. Provided the PUB is the whole data packet, the maximum number of codeword bits is 65536. In theory, the secure codeword bits can be inserted at any position within the standard packet codeword. Since the highest *RePunc* CR is $3/4$, for a 4096 bytes standard information packet, the maximum number of the punctured bits is 21845. To reduce the Viterbi decoding error, forbidden positions are defined in the standard packet to avoid the secure codeword insertion: First, only two bits can be replaced by the secure bits in every six continuous codeword bits; Second, the codeword bits generated from the same information bit cannot be all replaced.

In the above example, the first secure bit can be inserted at any position of the 65536 standard codeword bits, hence there are n_p0 65536 possible patterns. The second secure bit can be inserted at any remaining position of 65535 bits, except the forbidden positions. The number of the forbidden positions for the current secure codeword bit is named n_fbd, which is one for the second secure bit, or four for the third secure bit. Thus, the second secure bit's number of possible puncturing patterns n_p1 is 65534 = 65535 - 1. Similarly, n_p2 can be calculated as 65530.

The total number of possible puncturing patterns, $2.8{*}10^{95733}$, can be computed by multiplying the number of possible puncturing patterns for all secure bits, showing in Equation 2.

$$NUM_{comb} = n\_p_0 * n\_p_1 * n\_p_2 * ... * n\_p_{21844} \quad (2)$$
$$n\_p_0 = 65536; n\_p_1 = 65534; n\_p_2 = 65530; ..., n\_p_{21844} = 4;$$
$$complexity = NUM_{comb} * cyc\_pkt \quad (3)$$
$$cyc\_pkt = NUM_{bits} * cyc\_bit$$

The security complexity of *RePunc*, shown in Equation 3, is formulated by the number of clock cycles (i.e., time spent at the Viterbi decoder) to try all possible puncturing patterns. We define that the cyc_pkt is the number of clock cycles to decode a packet in Viterbi decoding, which is determined by the number of bits in the secure packet (NUM_{bits}), and the number of clock cycles to decode a single bit in Viterbi decoding (cyc_bit). We consider the attacker can use a hardware based reconfigurable Viterbi decoder for hacking the secure bits. The cyc_bit is 12 clock cycles in our reconfigurable Viterbi decoder design. For the 1000 bytes secure packet example, 96000 clocking cycles are needed to attack one key with the hardware circuit. With $2.8 * 10^{95733}$ puncturing pattern combinations, an eavesdropper needs $4.26 * 10^{95721}$ years to attack the key by brute force with a 2GHz working clock.

D. RePunc Hardware Design

An efficient parallel processing *RePunc* hardware implementation, supporting the normal puncturing and secure bits insertion, is presented in this section. We design our *RePunc* hardware implementation as a reconfigurable ASIC circuit for better performance.

The PUB length is determined by trading-off the security complexity and the hardware implementation cost. The longer PUB, the more hardware resources are needed. In our *RePunc* implementation, the PUB length is set to 40 bits.According to Equation 2, the number of different puncturing patterns is reduced from the theoretic $2.8 * 10^{95733}$ to $1.77 * 10^{40}$, however, it is still larger than the number of parameter combinations in AES, i.e., $2^{128} = 3.40 * 10^{38}$.

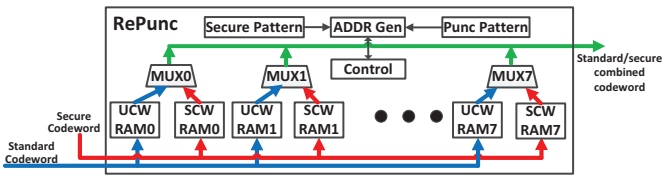

Fig. 2. *RePunc* Hardware Block Diagram

The *RePunc* hardware consists of eight standard packet codeword RAMs, eight secure packet codeword RAMs, eight MUXes, one RAM address generator logic, a control logic, and buffers to hold the normal puncturing pattern and the secure bits insertion pattern, as shown in Fig. 2. Although one set of codeword RAMs is enough for the serial implementation, the parallel eight codeword RAMs improve the throughput eight times higher. The codeword bits from the standard packet are stored in eight standard packet codeword RAMs, from *UCW RAM0-7*, each of these RAM stores the corresponding bits from an entire PUB. Since the PUB length is 40 bits, the size of each of these RAMs is 80 bits. The reordered secure bits for eight PUBs are stored in eight secure packet codeword RAMs, from *SCW RAM0-7*, each of these RAMs has the size 26 bits, which is in line of the highest *RePunc* $CR = 3/4$.

The address generator logic, *ADDR Gen*, generates RAM read addresses for *UCW RAM0-7*, and *SCW RAM0-7*, according to the *Punc Pattern* and *Secure Pattern* buffers which store the predefined patterns for the normal puncturing and secure codeword insertion patterns. The multiplexers *MUX0-7* select the signal either from the standard packet codeword RAMs, or the secure packet codeword RAMs, according to the patterns defined in *Secure Pattern*.

The *RePunc* hardware can support the normal puncturing by setting *Punc Pattern* and clearing *Secure Pattern*.

Due to space limitations the reconfigurable de-puncturing hardware implementation at the receiver side is not presented. However, it is similar with the *RePunc* structure shown in Fig. 2.

IV. EXPERIMENTAL SETUP

RePunc is implemented in Matlab and RTL for the security protocol verification and hardware performance evaluation respectively. The whole WiFi 802.11a baseband is designed in

Matlab at first, then the puncturing module is replaced by the *RePunc* according to Algorithm 1.

Clearly, the inserted secure packet codeword bits worsen the receiving performance of the standard packet. We use PER to analyze the reception performance. The packet length for all PER analysis is 1024 bytes.

The *RePunc* FEC, including the convolutional encoder, Viterbi decoder, and de-puncturing, is designed in RTL, and compared with the WiFi FEC. Area and timing results are generated by the Synopsys Design Compiler with TSMC 65nm technology, and power consumption is estimated by Synopsys Prime Time. Same synthesis settings (such as input/output delay, fan-out, etc.) are applied to both *RePunc* and WiFi hardware designs.

V. Results and Analysis

A. PER Impact

The impact on the standard packet reception PER, caused by *RePunc*, is analyzed by injecting different level of AWGN channel noises, represented by SNRs. Three different WiFi CR scenarios, named WF0 - WF2, for CR $1/2$, $2/3$ and $3/4$, are analyzed as a benchmark. Nine *RePunc* scenarios, named RP2, RP5 - RP26, represent the number of inserted secure bits from 2, 5 to 26 with a step size three in a 40 bits PUB. The RP26 has *RePunc* $CR = 40/(40 * 2 - 26) = 0.74$, which is closest to the highest CR in WiFi (i.e., WF2 $CR = 3/4 = 0.75$).

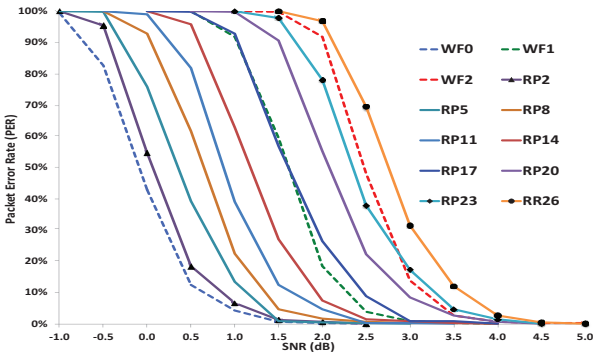

Fig. 3. *RePunc* PER Impact on Standard Packets

The measured PERs of these 12 test scenarios are plotted in Fig. 3. The WF0, shown in the blue dashed curve, has the best PER performance since it is the WiFi transmission without any puncturing. The WF2, shown in the red dashed curve, has nearly the worst PER performance since it has the most number of punctured bits. The RP2, RP5, - RP23 PER performances are in between WF0 and WF2, since the numbers of punctured bits are all larger than WF0, but smaller than WF2. Although the RP26 *RePunc* CR (0.74) is less than WF2's, it has worse PER than WF2. The reason is that the WF2 has an even distributed punctured pattern, while the random generated RP26 puncturing pattern has uneven distribution.

The measured PER results show that the impact of *RePunc* on the reception performance of the standard packet is lower than the WiFi CR $3/4$, except the RP26. The worst scenario RP26 only has 3dB PER degradation, compared to the non-puncturing WiFi $CR = 1/2$. Thus, the eavesdropper cannot identify if the secure packet is inserted into the standard packet.

B. Hardware performance analysis

The hardware implementation area, throughput and power consumption results from the *RePunc* and the standard WiFi FECs, are summarized in Table I.

The gate count of *RePunc* logic is 12.8K, which is about 10% higher than WiFi (11.6K). This result shows that the proposed *RePunc* has very low hardware implementation overhead. Please note the area for RAMs in both the *RePunc* and WiFi are excluded since we do not have a formal RAM simulation model.

TABLE I
RePunc HARDWARE OVERHEAD

	Gate Count (K)	Throughput (Gbps)	Power (uW)
RePunc	12.8	3.64	86
WiFi	11.6	4.82	77

The throughput, showing the maximum signal processing ability, is determined by the maximum working frequency, and the number of cycles to process an information bit. The maximum working frequency is determined by the delay of the critical path in the design.

We show how the throughput is calculated using an example. The maximum working frequency of *RePunc* is 990 MHz, i.e., 1.01ns period for each clock cycle. 87 cycles are used to process eight PUBs in *RePunc*. Each of these PUBs processes 40 information bits. By knowing the time for these 87 cycles and the number of processing bits, the *RePunc* throughput can be calculated as 3.64 Gbps. Similarly, the throughput of WiFi implementation is 4.82 Gbps.

Clearly, the *RePunc* throughput is lower than the WiFi implementation. This is due to the longer critical path delay in *RePunc* and the more number of clock cycles for each information bit processing. Although the *RePunc* throughput is about 24% lower than WiFi, the *RePunc* throughput is still much higher than the required highest 480 Mbps data rate from WiFi protocol [10].

Power consumption is a critical factor in wireless communication devices. The power consumption of *RePunc* and WiFi are measured by executing the 6Mbps data rate WiFi FEC. *RePunc* consumes 86 uW while WiFi consumes 77 uW. The power consumption overhead is about 12% for *RePunc* compared to WiFi.

VI. Conclusion

In this paper, a physical layer secure communication methodology is proposed by a low overhead, high performance reconfigurable puncturing signal processing. Using our methodology, the secure packets can be transmitted wirelessly without being identified by the eavesdropper. Moreover, even if the eavesdropper tries to hack the secure packet in *RePunc*, the theoretic complexity is extremely high to perform an exhaustive search.

References

[1] A. Lindell, "Test Attacks on the Pairing Protocol of Bluetooth v2.1," in *BlackHat*, 2008.

[2] J. Cho, Y. Kim, and K. Cheun, "A novel frequency-hopping spread-spectrum multiple-access network using m-ary orthogonal walsh sequence keying," *IEEE Trans. on Computers*, vol. 51, no. 11, pp. 1885–1896, 2003.

[3] T. Kang, X. Li, C. Yu, and J. Kim, "A survey of security mechanisms with direct sequence spread spectrum signals," *JCSE*, 2013.

[4] M. I. Husain, S. Mahant, and R. Sridhar, "CD-PHY: Physical Layer Security in Wireless Networks through Constellation Diversity," *CoRR*, vol. abs/1108.5148, 2011.

[5] M. Nutzinger, C. Fabian, and M. Marschalek, "Secure Hybrid Spread Spectrum System for Steganography in Auditive Media," in *IIH-MSP*, 2010.

[6] Q. Mao and C. Qin, "A novel turbo-based encryption scheme using dynamic puncture mechanism.," *JNW*, vol. 7, no. 2, pp. 236–242, 2012.

[7] L. Tang, J. A. Ambrose, S. Parameswaran, and S. Zhu, "Reconfigurable Convolutional Codec for Physical Layer Communication Security Application," in *MILCOM*, 2014.

[8] X. Ma, M. M. Olama, T. Kuruganti, S. F. Smith, and S. M. Djouadi, "Security of Classic PN-spreading Codes for Hybrid DS/FH Spread-Spectrum Systems," in *MILCOM*, 2013.

[9] M. Daly and J. Bernhard, "Directional Modulation Technique for Phased Arrays," *IEEE Trans. on Antennas and Propagation*, 2009.

[10] "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 2011. Available at: http://standards.ieee.org/.

[11] "Bluetooth specification version 1.2," 2003. Available at: http://www.bluetooth.org/.