Modified Interleaver Update 2/11/19

Benjamin Davis
Computer Engineering
California Polytechnic State University
San Luis Obispo, CA
benjamin.j.davis96@gmail.com

I. Problems With Current Interleaving Algorithm and Previous Proposal

The interleaving jamming scheme for 802.11 proposed by Vo-Huu et al. takes advantage of the current 802.11 interleaving scheme. This scheme is developed in such a way as to ensure that any two bits which were adjacent in the original coded data sequence are going to be a multiple of three subcarriers apart. This allowed for effective jamming of the entire signal through only jamming a third of the entire set of digital subcarriers (DSC). In order to prevent the attack from being performed this easily it is necessary to develop an interleaving scheme which does not have the same interval repetition property.

In earlier work an interleaver was developed which relied on the array of numbers generated by a linear congruential generator. This provided promising results as it spread the bits more effectively and required the jammer to use a 50% higher jamming power to receive the same results. However, this still allowed the interleaver to be targeted as a jamming attack surface at a 50% power benefit over broad spectrum jamming. In order to remedy this another novel jamming technique which relies on a pre-shared secret is desired so that the interleaver cannot be attacked.

II. Proposal

To accomplish this it is necessary to find a deterministic method of using the pre-shared secret to generate a random permutation of subcarriers for each bit within an individual subcarrier. This permutation would then be used in the same way as the array generated by the LCG in the previous interleaver proposal.

The generation of the permutation is done using a NIST approved Deterministic Random Number Generator (DRNG) for Computer Security (NIST 800-90a Rev. 1). This DRNG will be used in two steps. The first step uses the DRNG to attain the correct number of subchannels to use based on the subcarrier bit size. The second step uses the DRNG to perform a Fisher-Yates shuffle on the selected DSC to further randomize the order. This process produces a static array which will be used in the actual Interleaver.

The pseudo-code for the algorithms follows.

```
Permutation Generator Pseudo-Code
Input:
  Pre-Shared Secret: secret
  Bits per Subcarrier: b
Output:
  Permutation Array of size b: perm
Start:
  perm = []
  DRNG_init(secret)
  # Get DSCs to use
  i = 0
  while i < b
     randInt = DRNG_get_64_bit_int()
     randDbl = randInt / MAX64Bit
     randDSC = floor(randDbl * 48)
     if not perm contains randDSC
          perm.append(randDSC)
          i = i + 1

  # Fisher yates shuffle
  for i = 1 to b-1
  randInt = DRNG_get_32_bit_int()
  randDbl = randInt / MAX32Bit
  j = floor(randDbl * (b – i + 1)) + i
  temp = perm[j]
  perm[j] = perm[i]
  perm[i] = temp
```

```
Interleaver Pseudo-Code
Input:
  Bits per Subcarrier: b
  Coded Bit String: $X_0X_1…X_m$
    Where m = b*48
  Offset Array of size b: offset
Output:
  DSC Array: newDSC
Start:
  initDSC[48]
  for i = 0 to m-1
    initDSC[floor(i/b)] <= $X_i$

  newDSC[48]
  for s = 1 to 48
    for i = 1 to b
        dsc = (s + offset[i]) % 48
        newDSC[dsc] <= initDSC[s][i]
```

III. Analysis

Running this algorithm using Hash_DRNG (NIST 800-90a Rev. 1) as the DRNG yields promising results. Across a billion runs with different seeds for a b=8 the presence of any DSC within any given position of the permutation array is statistically equally probable. This demonstrates that at minimum this algorithm produces an even distribution of DSC for the permutation array.

Additional testing is ongoing regarding the collision frequency of this protocol. It will be impossible to ensure that all 2^44 possible permutations are found as this exceeds the memory limit of accessible computers. However, random sets of a million unique seeds should provide insight into any series collision errors that may occur.