Received 29 July 2020; revised 9 September 2020; accepted 17 September 2020. Date of publication 23 September 2020; date of current version 15 October 2020.

Digital Object Identifier 10.1109/OJCOMS.2020.3025536

Physical Layer Security for Multiple-Input Multiple-Output Systems by Alternating Orthogonal Space-Time Block Codes

MICHAEL R. CRIBBS (Graduate Student Member, IEEE), RIC A. ROMERO (Senior Member, IEEE), AND TRI T. HA (Life Fellow, IEEE)

Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA 93943, USA

CORRESPONDING AUTHOR: M. R. CRIBBS (e-mail: mrcribbs@hotmail.com)

ABSTRACT Algorithms are provided to build a large set of unique complex-valued orthogonal space-time block codes (STBCs) from a known or standard STBC. A physical layer security (PLS) scheme is proposed to take advantage of this set by alternating the STBC in use over a multiple-input single-output (MISO) or multiple-input multiple-output (MIMO) communications link between base station (BS) and user equipment (UE). A practical procedure is proposed and demonstrated to build individual STBCs from the set without use of a lookup table. The sufficient statistic is given and proven to allow for maximal ratio combining (MRC) by the intended receiver for all STBCs in the set. An algorithm is offered for the UE to update the MRC matrix in use as the STBC alternates. Definitions are provided for cryptograms, key residues (KRs), key residue classes (KRCs), and message and cryptogram residue classes pertaining to STBC PLS schemes. These definitions are used to present analysis of the information-theoretic security of the proposed PLS scheme to include message and key equivocation. Theoretical expected bit error rate (BER) for a passive eavesdropper is proven and plotted along with Monte Carlo simulations for confirmation. Discussion of different attack models is provided. Cost and attack complexity are compared between the proposed scheme and two related techniques from the literature.

INDEX TERMS Cryptogram residue class, key equivocation, key residue class, message equivocation, physical layer security, space-time block code.

I. INTRODUCTION

PHYSICAL layer security is currently a major area of research for use with emergent 5G networks and as a potential mitigation strategy towards quantum computing's expected ability to rapidly break certain types of cryptography [1]–[5]. In his pioneering work in this area of research, Wyner introduced the wire-tap channel where secrecy capacity was achieved assuming the signal received over the wire-tap channel was a degraded version of that received over the main channel [6]. Further research loosened this assumption to allow for security without such disparity between channels [7]. Many physical layer security (PLS) approaches employ the use of forward error correcting codes that may be randomized or punctured to provide security [8]–[10]. Other common approaches over multiple-input multiple-output (MIMO) channels include use of precoding matrices or insertion of artificial noise [11]–[14]. Although less common in the literature, this article employs the use of space-time block codes (STBCs) to provide security as in [15]–[17].

Whereas some PLS techniques do not require any form of pre-shared secret [6], [12], there are a number of schemes that do [16]–[19]. Many PLS approaches use the channel state information (CSI), received signal strength indicator (RSSI), or other main channel statistics to establish an element of secrecy between the transmitter and intended receiver [4], [15], [20], [21]. In [15], RSSI was used as a secret key to seed a pseudorandom number generator (PRNG) used to generate phase rotations for each transmit (TX) antenna of a MIMO system after encoding with either the Alamouti STBC from [22] or the 3/4 rate Octonion orthogonal STBC from [23]. Similarly in [16], a pre-shared secret pseudorandom antipodal sequence was applied across the TX antennas of a MIMO system after

encoding with a real-valued orthogonal STBC. A generalization and extension of this technique was presented in [17] where a pre-shared secret STBC was drawn from a set of complex-valued orthogonal STBCs. This article extends [17] by building a much larger set of STBCs, alternating the STBC after each use, and providing a generic framework for analysis of similar STBC based PLS schemes.

The contributions of this article are:

- to propose and design an alternating orthogonal STBC PLS scheme for use over a multiple-input single-output (MISO) or MIMO communications link in the presence of passive eavesdropping,
- to provide new and updated algorithms from [17] to build a much larger set of unique orthogonal STBCs for use with the proposed PLS scheme,
- to offer a very practical technique to build individual STBCs without storing the full set which could be prohibitive, especially for low-storage devices, due to very large set cardinalities,
- to provide and prove a sufficient statistic for maximal ratio combining (MRC) for all STBCs in the set,
- to offer an algorithm for use by the intended receiver to update the MRC matrix as the STBC alternates,
- to adapt nomenclature common to secrecy systems for use with STBC PLS schemes,
- and to provide detailed theoretical analysis and discussion of complexity cost of the proposed scheme.

The benefits of the STBC PLS techniques presented in [15]–[17] carried forward to this work include no required CSI at the transmitter, no transmission power diverted to artificial noise, no increase to transmit signal peak-to-average power ratio (PAPR), full diversity for improved reception over fading channels, and linear decoding complexity for the intended receiver.

The remainder of this article is outlined as follows. In Section II, the base code for this work is presented. In Section III, details of two new set building algorithms are provided. In Section IV, we demonstrate the process to build individual STBCs from the set. In Section V, we present the proposed PLS scheme. In Section VI, common nomenclature for secrecy systems is adapted to STBC PLS schemes. In Section VII, information-theoretic security of the proposed PLS scheme is analyzed along with expected bit error rate (BER) for a passive eavesdropper (*aka* "Eve"). In Section VIII, different attack models are discussed, and a comparison of the cost and attack complexity of our scheme to that of two related techniques from the literature is given. In Section IX, we recap the contents of this article. In Appendix A, updated algorithms from [17] are included for completeness. In Appendix B, a proof is provided for the MRC details presented in Section V. In Appendix C, a tabular form key residue (KR) is included for the chosen base code. Finally, in Appendix D, a theoretical proof is given for Eve's expected BER.

II. BASE CODE

The base code employed in [17], referred to as single-pair orthogonal 3-3-4 (SPO334) from this point forward, is a complex orthogonal 3/4 rate STBC represented as

$$\mathbf{G} = \begin{bmatrix} s_1 & s_2 & s_3 \\ -s_2^* & s_1^* & 0 \\ -s_3^* & 0 & s_1^* \\ 0 & -s_3^* & s_2^* \end{bmatrix}, \tag{1}$$

consisting of three variations of three data symbols, s_1 through s_3, transmitted over four symbol time periods using three TX antennas. The asterisk, (*), represents complex conjugate operation.

The base code selected for this work, referred to as multi-pair orthogonal 4-6-8 (MPO468) from this point forward, is a complex orthogonal 6/8 rate STBC represented as

$$\mathbf{G} = \begin{bmatrix} s_1 & s_2 & s_3 & 0 \\ -s_2^* & s_1^* & 0 & s_6 \\ -s_3^* & 0 & s_1^* & -s_5 \\ 0 & -s_3^* & s_2^* & s_4 \\ s_4 & s_5 & s_6 & 0 \\ -s_5^* & s_4^* & 0 & s_3 \\ -s_6^* & 0 & s_4^* & -s_2 \\ 0 & -s_6^* & s_5^* & s_1 \end{bmatrix}, \tag{2}$$

consisting of four variations of six data symbols, s_1 through s_6, transmitted over eight symbol time periods using four TX antennas. MPO468 is based upon a STBC constructed in [24] but reordered to show how it extends SPO334. As the code rate of these two STBCs is equivalent, moving to the MPO468 STBC may not yield an advantage in terms of capacity; however, the additional columns, rows, and data symbols in MPO468 allow for a much larger set cardinality leading to improved PLS.

The number of columns in $\mathbf{G}$, denoted as c from this point forward, is equal to the number of TX antennas employed per codeword, where codeword refers to a single block of encoded symbols as dictated by $\mathbf{G}$. The number of rows in $\mathbf{G}$, denoted as r from this point forward, is equal to the number of symbol time periods used per codeword. Throughout this article, the data symbol vector for this STBC refers to $\mathbf{s} = [s_1 \ s_2 \ s_3 \ s_4 \ s_5 \ s_6]^T$, where T represents the transpose operation. An extended data symbol vector, $\mathbf{s}_{ext} = [\mathbf{s}^T \ \mathbf{s}^\dagger]^T$, is also referenced, where † represents the conjugate transpose operation.

III. EXTENSION OF SET BUILDING ALGORITHMS

Although the six set building algorithms presented in [17] are sufficient for SPO334, additional transformations are possible for MPO468.

A. SYMBOL PERMUTATIONS

One reason why the algorithms in [17] are sufficient for SPO334 is that the rows of $\mathbf{G}$ in (1) contain all $\binom{3}{2}$ combinations of the three data symbols, where $\binom{3}{2}$ represents the binomial coefficient 3 choose 2. With MPO468 however,

$r < \binom{6}{3}$; therefore, not all $\binom{6}{3}$ combinations of the six data symbols appear in rows of $\mathbf{G}$ in (2). Thus a symbol permutation algorithm is required to provide these additional symbol combinations.

Example 1: A symbol permuted version of (2) that cannot be built using algorithms from [17] alone is

$$\mathbf{G} = \begin{bmatrix} s_1 & s_2 & s_3 & 0 \\ -s_2^* & s_1^* & 0 & s_5 \\ -s_3^* & 0 & s_1^* & -s_6 \\ 0 & -s_3^* & s_2^* & s_4 \\ s_4 & s_6 & s_5 & 0 \\ -s_6^* & s_4^* & 0 & s_3 \\ -s_5^* & 0 & s_4^* & -s_2 \\ 0 & -s_5^* & s_6^* & s_1 \end{bmatrix}, \tag{3}$$

which is built by swapping symbols s_5 and s_6.

Not all symbol permutations are permitted in order to ensure uniqueness of codes within the set. An allowed symbol permutations vector, denoted as $\mathbf{v}_{asp}$, is used for this purpose within the symbol permutations algorithm. For any given base STBC, $\mathbf{v}_{asp}$ may be empirically derived by performing each of the $k!$ symbol permutations in lexicographic order and recording the first of each set of permutations that yields a new combination of symbols per row of $\mathbf{G}$. Through this process for MPO468,

$$\mathbf{v}_{asp} = [1\ \ 2\ \ 3\ \ 4\ \ 5\ \ 6\ \ 7\ \ 8\ \ 9\ \ 11\ \ 13\ \ 15\ \ 31\ \ 32\ \ 37]. \tag{4}$$

Whereas 15 allowed symbol permutations, including the original, are recorded in $\mathbf{v}_{asp}$, it can be shown that there are $\frac{k!}{15} - 1 = \frac{720}{15} - 1 = 47$ other permutations for each of the entries in $\mathbf{v}_{asp}$ that produces the same combination of symbols per row.

B. SYMBOL NEGATIONS

A second reason why the algorithms presented in [17] are sufficient for SPO334 is that orthogonality of any two columns of $\mathbf{G}$ relies on a single pair of symbols that must yield a zero inner product. By contrast, orthogonality of any two columns for MPO468 relies on two pairs of symbols to yield a zero inner product. To clarify, a symbol pair between columns of $\mathbf{G}$ refers to the four symbol elements contained in two rows consisting of two variations of two data symbols. When variations are constrained to original, negative, conjugate, or negative-conjugate data symbols, a single conjugate operator must appear between the two variations of each symbol, and an odd number of negatives must appear among the four symbol elements in order for the symbol pair to yield a zero inner product. Thus, there are 32 possible conjugation and negation combinations for any given symbol pair that yield a zero inner product. Each of these combinations results in a conjugate relationship between the two variations of one symbol and a negative-conjugate relationship between the two variations of the other symbol. These symbol pair relationships are important when defining KRs in Section VI-B.

Example 2: Two of the 32 combinations yielding a zero inner product for the symbol pair that exists between columns one and two in (1) and (2) are

$$\begin{bmatrix} s_1 & s_2 \\ -s_2^* & s_1^* \end{bmatrix} \quad \text{and} \tag{5a}$$

$$\begin{bmatrix} s_1 & -s_2^* \\ -s_2 & -s_1^* \end{bmatrix}. \tag{5b}$$

With only a single symbol pair of concern to maintain orthogonality between any two columns of SPO334, the algorithms in [17] cycle through all "compatible" methods of zeroing the inner product for each symbol pair. We use the term compatible to highlight the fact that symbol pairs are not independent of one another when any symbol belongs to more than one pair. Due to the number of symbols per row of $\mathbf{G}$, it can be seen that each of the three symbols on the top row of SPO334 belong to two symbol pairs whereas the remaining symbols in the matrix belong to only one. By contrast, every symbol within the MPO468 STBC belongs to two symbol pairs. With two pairs of concern to maintain orthogonality between any two columns of MPO468, a symbol negation algorithm is required to more finely affect the relationships within each symbol pair.

Example 3: A symbol negated version of (2) that cannot be built using algorithms from [17] alone is

$$\mathbf{G} = \begin{bmatrix} -s_1 & s_2 & s_3 & 0 \\ -s_2^* & -s_1^* & 0 & s_6 \\ -s_3^* & 0 & -s_1^* & -s_5 \\ 0 & -s_3^* & s_2^* & s_4 \\ s_4 & s_5 & s_6 & 0 \\ -s_5^* & s_4^* & 0 & s_3 \\ -s_6^* & 0 & s_4^* & -s_2 \\ 0 & -s_6^* & s_5^* & -s_1 \end{bmatrix}, \tag{6}$$

which is built by negating all variations of symbol s_1.

For the symbol negation algorithm, it is important to note, based upon the initial base STBC in use, which data symbols may be negated, individually or concurrently, without resulting in duplicate codes. An allowed symbol negations vector, denoted as $\mathbf{v}_{asn}$, is used for this purpose within the symbol negations algorithm. As previously explained for SPO334, the symbol negation algorithm is unnecessary; thus, to prevent building duplicate codes, $\mathbf{v}_{asn}$ is empty for that base code. Conversely, symmetry within the MPO468 code allows for a few combinations of symbols to be permitted for use with this algorithm. For instance, symbols s_1 and s_2 represent one combination whereas symbols s_4 and s_5 represent another. When negated individually or concurrently, and in combination with particular row and column negations, the permitted symbol combinations allow for any combination of data symbols to be negated. The choice as to which combination of symbols to permit with this algorithm is up to the user; however, whereas the same set of STBCs is built either way, the indexing of the set is slightly different.

On the contrary, there are other symbol combinations, e.g., s_1 and s_4, that are not permitted with the MPO468 base code.

Algorithm 6 Symbol Permutations

Inputs: $\mathbf{S}$ {Input set of STBCs},
PC_i {# of populated codes in input $\mathbf{S}$},
$\mathbf{A}_{syms}$ {Symbols record array},
$\mathbf{v}_{asp}$ {Allowed symbol permutations vector},
$\mathbf{s}_{ext}$ {Extended data symbol vector}
Outputs: $\mathbf{S}$ {Output set of STBCs},
PC_o {# of populated codes in output $\mathbf{S}$},
$\mathbf{A}_{syms}$ {Updated symbols record array}

$\mathbf{K}_{perms} = perms(k)$ {Symbol permutations array}
$PC_o = PC_i \cdot |\mathbf{v}_{asp}|$
{Operation limit = $|\mathbf{v}_{asp}|$}
for $it = 1$ to $|\mathbf{v}_{asp}| - 1$ **do**
  for $m = 1$ to PC_i **do**
    $\mathbf{S}(:, :, PC_i \cdot it + m) = \mathbf{S}(:, :, m)$
    {Initialize STBC}
    $[\sim, \mathbf{v}_{kpi}] = sort(\mathbf{A}_{syms}(m, :))$
    {$\mathbf{v}_{kpi}$ holds indices of sorted symbol permutation}
    $\mathbf{v}_{kp} = \mathbf{K}_{perms}(\mathbf{v}_{asp}(it + 1), \mathbf{v}_{kpi})$
    {Symbol permutation vector}
    if $\mathbf{v}_{kp} == [1 \ \ldots \ k]$ **then**
      $\mathbf{A}_{syms}(PC_i \cdot it + m, :) = [1 \ \ldots \ k]$
      {Update symbols record array}
      $\mathbf{v}_{kp} = \mathbf{v}_{kpi}$
      {Reassign symbol permutation vector}
    else
      $\mathbf{A}_{syms}(PC_i \cdot it + m, :) = \mathbf{K}_{perms}(\mathbf{v}_{asp}(it + 1), :)$
      {Update symbols record array}
    end if
    for $ri = 1$ to r **do**
      for $ci = 1$ to c **do**
        if $\mathbf{S}(ri, ci, m) \neq 0$ **then**
          $j =$ symbol index of $\mathbf{S}(ri, ci, m)$
          {e.g. $j = 2$ for $\mathbf{S}(ri, ci, m) == s_2, s_2^*$, etc.}
          if $\mathbf{v}_{kp}(j) \neq j$ **then**
            if $\mathbf{S}(ri, ci, m)$ is negated **then**
              $n = -1$
              {Permuted symbol must be negated}
            else
              $n = 1$
            end if
            if $\mathbf{S}(ri, ci, m)$ is conjugated **then**
              $g = 1$
              {Permuted symbol must be conjugated}
            else
              $g = 0$
            end if
            $\mathbf{S}(ri, ci, PC_i \cdot it + m) = n \cdot \mathbf{s}_{ext}(\mathbf{v}_{kp}(j) + g \cdot k)$
          end if
        end if
      end for
    end for
  end for
end for
return $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

Due to this fact, it is important to keep track of the symbols to be negated if symbol permutation is performed prior to symbol negation. In this case, the permutation applied within the symbol permutation algorithm is used to permute the chosen symbol combination for use with the symbol negation algorithm.

Example 4: A group of STBCs is built by performing symbol permutation prior to symbol negation. Within the symbol permutation algorithm, all original s_2 symbols are replaced with s_3, all original s_3 symbols with s_4, and all original s_4 symbols with s_2. Thus, if the chosen combination of symbols for use with the symbol negation algorithm is s_1 and s_2, i.e., $\mathbf{v}_{asn} = [1 \ 2]$, then the permuted combination of symbols becomes s_1 and s_3, i.e., $\mathbf{v}_{psn} = [1 \ 3]$.

C. DETAILED SET BUILDING ALGORITHMS

In order to build the set, $\mathbf{S}$, of unique orthogonal STBCs from the MPO468 base code, the initialization algorithm is performed first followed by the remaining algorithms in Appendix A and this subsection in any order. The cardinality of $\mathbf{S}$, denoted as $|\mathbf{S}|$, including the base code is

$$\left|\mathbf{S}\right| = 2^{r+c+k-1} \cdot r! \cdot c! \cdot |\mathbf{v}_{asp}| \cdot 2^{|\mathbf{v}_{asn}|}, \tag{7}$$

which equals 7, 610, 145, 177, 600 for $c = 4$, $k = 6$, $r = 8$, $|\mathbf{v}_{asp}| = 15$ is the length of $\mathbf{v}_{asp}$, and $|\mathbf{v}_{asn}| = 2$ is the length of $\mathbf{v}_{asn}$. This cardinality is obtained by the product of the operation limits for all algorithms in Appendix A and this subsection. The operation limit for each algorithm is the multiplication factor applied within to obtain the number of populated codes in the output set from that of the input. The two far-right terms in (7) are contributed by addition of the two new algorithms from this work; thus, it can be seen that these new algorithms increase the set cardinality by a factor of 60.

The values of c, k, and r are known to all algorithms in this article. All text within curly brackets are comments to aid the reader. A slightly modified version of the $perms(n)$ MATLAB function is used to create an $n!$-by-n array containing all permutations of the integers from 1 to n in lexicographic order, where each row contains a different permutation [25]. The $[\mathbf{B}, \mathbf{I}] = sort(\mathbf{A})$ MATLAB function is used to sort the elements of vector $\mathbf{A}$ in ascending order, where $\mathbf{B}$ holds the sorted elements, and $\mathbf{I}$ holds the original indices of the elements of $\mathbf{A}$ as they appear within $\mathbf{B}$ such that $\mathbf{B} = \mathbf{A}(\mathbf{I})$ [25]. The $\sim$ symbol in place of $\mathbf{B}$ indicates that $\mathbf{B}$ is not used. The $de2bi(i, n)$

Algorithm 7 Symbol Negations

Inputs: $\mathbf{S}$, PC_i, $\mathbf{A}_{syms}$, $\mathbf{v}_{asn}$ {Allowed symbol negations vector}
Outputs: $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

$PC_o = PC_i \cdot 2^{|\mathbf{v}_{asn}|}$ {Operation limit = $2^{|\mathbf{v}_{asn}|}$}
for $it = 1$ to $2^{|\mathbf{v}_{asn}|} - 1$ **do**
  $\mathbf{v}_{sym} = de2bi(it, |\mathbf{v}_{asn}|)$ {Symbol negation vector}
  for $m = 1$ to PC_i **do**
    $\mathbf{A}_{syms}(PC_i \cdot it + m, :) = \mathbf{A}_{syms}(m, :)$ {Update symbols record array}
    $\mathbf{v}_{psn} = \mathbf{A}_{syms}(m, \mathbf{v}_{asn})$ {Permuted allowed symbol negations vector}
    $\mathbf{v}_{csn} = \mathbf{v}_{sym} \circ \mathbf{v}_{psn}$ {Current iteration symbol negation vector set}
    $\mathbf{T}$ = 2-dimensional array of 1's of size r-by-c
    for $ri = 1$ to r **do**
      for $ci = 1$ to c **do**
        if $\mathbf{S}(ri, ci, m) \neq 0$ **then**
          $j =$ symbol index of $\mathbf{S}(ri, ci, m)$ {e.g., $j = 3$ for $\mathbf{S}(ri, ci, m) == s_3, s_3^*$, etc.}
          if $j \in \mathbf{v}_{csn}$ **then**
            $\mathbf{T}(ri, ci) = -1$
          end if
        end if
      end for
    end for
    $\mathbf{S}(:, :, PC_i \cdot it + m) = \mathbf{S}(:, :, m) \circ \mathbf{T}$
  end for
end for
return $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

MATLAB function is used to create an n-bit, little-endian (*aka* right most significant bit) binary row vector of the decimal value i [25]. Lastly, $\times$ is used to indicate matrix multiplication whereas $\circ$ indicates element-wise multiplication, and == indicates logically equal whereas = indicates assignment. For completeness and to facilitate the discussion for the reader, the updated algorithms from [17] are included in Appendix A with their original numbering, and the two new algorithms are presented in this subsection as Algorithms 6 and 7.

D. TRACKING SYMBOL PERMUTATIONS

The symbols record array, denoted as $\mathbf{A}_{syms}$, is used for tracking symbol permutations throughout performance of each algorithm. When the $\mathbf{G}$ input to Algorithm 0 is a non-symbol permuted version of the original base code for which $\mathbf{v}_{asp}$ is empirically derived, then $\mathbf{K}_{perm} = [1 \ \ldots \ k]$ is input to Algorithm 0 as well. For this case, symbol permutation tracking is only necessary when Algorithm 6 is performed prior to Algorithm 7. This serves to permute $\mathbf{v}_{asn}$ as discussed in Example 4 and performed in Algorithm 7. On the contrary, when a symbol permuted version of the base code is used as the $\mathbf{G}$ input to Algorithm 0, the $\mathbf{K}_{perm}$ input to Algorithm 0 is vital to properly initialize $\mathbf{A}_{syms}$ which is then used within Algorithm 6 to determine the correct symbol permutation vector, $\mathbf{v}_{kp}$, to prevent duplicate codes from being built.

IV. INDIVIDUAL STBC BUILDING

In schemes that perform beamforming, precoding, or encoding by selecting a code for use from a known codebook (*aka* code set), it is common to store the entire codebook for use as a lookup table [26], [27]; however, when the size of the codebook is very large, this may not be practical, especially for low-storage devices. With this in mind, it would be desirable to build specific selected STBCs from the codebook for use as needed without storing the entire codebook. In order to do this, three pieces of information are needed:

1) the base code and
2) order of operations/algorithms used to build the intended codebook as well as
3) the index value of the desired code.

The base code must be known as it is the first indexed code in the set. The order of operations conducted must also be known as building the set with different order of operations effectively reindexes the entire set with the exception of the first code. Naturally, the set is indexed from 1 to $|\mathbf{S}|$. The index value must be known as it specifies the iteration of each operation to be performed in order to build the selected code, where the iteration of an operation refers to the value of the it variable in each one of the algorithms provided. A fourth piece of information, namely the operation limits discussed in Section III-C, must be known as well but can generally be considered as common knowledge for a given code set.

With these pieces of information, a specific STBC can be built by starting with the base code, determining the specified iteration of each operation to perform based upon the given index value, and performing each operation in the given order. The process for determining the specified iteration of each operation starts by defining a 1-by-l divisor vector, $\mathbf{v}_{div}$, with elements

$$\mathbf{v}_{div}(i) = \begin{cases} 1 & i = 1 \\ \prod_{j=1}^{i-1} \mathbf{v}_{ol}(\mathbf{v}_{oo}(j)) & 1 < i \leq l, \end{cases} \tag{8}$$

where $\mathbf{v}_{ol}$ is a row vector containing the operation limits for Algorithms 1 - 7 in that order, $\mathbf{v}_{oo}$ is an order of operations row vector containing the number of each algorithm in the order in which they are performed to build a code, and l is the length of each row vector, which equals seven when all of the provided algorithms are performed. Upon definition of $\mathbf{v}_{div}$, the iteration of each operation can be contained within a 1-by-l iterations vector, $\mathbf{v}_{it}$, with elements

$$\mathbf{v}_{it}(i) = \begin{cases} \left\lfloor \frac{IV-1}{\mathbf{v}_{div}(i)} \right\rfloor \bmod \mathbf{v}_{ol}(\mathbf{v}_{oo}(i)) & 1 \leq i < l \\ \left\lfloor \frac{IV-1}{\mathbf{v}_{div}(i-1)} \right\rfloor / \mathbf{v}_{ol}(\mathbf{v}_{oo}(i-1)) \rfloor & i = l, \end{cases} \tag{9}$$

where IV represents the index value of the desired code, mod represents the modulo operator, and $\lfloor \rfloor$ represents the

floor function. With these vectors defined, it can be seen that

$$IV = \mathbf{v}_{it} \times \mathbf{v}_{div}^{T} + 1, \tag{10}$$

where $+1$ is required due to set indexing starting at 1.

Once the iterations have been determined, the desired code is built by performing Algorithm 0 first with the $|\mathbf{S}|$ input argument set to 1 since only 1 code will be built. Algorithms 1 - 7 are performed next in the order specified by $\mathbf{v}_{oo}$. To modify each of these algorithms for individual STBC building instead of set building, any reference to $\mathbf{S}$ is replaced with $\mathbf{G}$, the values of PC_i and PC_o are set to 1, the *it* variable is set equal to the value contained in $\mathbf{v}_{it}$ for that algorithm, and all index values of $PC_i \cdot it + m$ are replaced with 1. For any element of $\mathbf{v}_{it}$ equal to zero, the corresponding algorithm is not performed.

Example 5: Using the MPO468 base code with $\mathbf{v}_{oo} = [3\ 6\ 2\ 1\ 7\ 4\ 5]$, $IV = 3,824,537,371,943$, and operation limits as specified in Algorithms 1–7, i.e.,

$$\begin{aligned}\mathbf{v}_{ol} &= \left[2^{r} \quad 2^{c-1} \quad r! \quad c! \quad 2^{k} \quad |\mathbf{v}_{asp}| \quad 2^{|\mathbf{v}_{asn}|}\right] \\ &= \left[256 \quad 8 \quad 40320 \quad 24 \quad 64 \quad 15 \quad 4\right],\end{aligned} \tag{11}$$

$$\mathbf{v}_{div} = \begin{bmatrix} 1 \\ 40320 \\ 604800 \\ 4838400 \\ 1238630400 \\ 4954521600 \\ 118908518400 \end{bmatrix}^{T}, \tag{12}$$

$$\mathbf{v}_{it} = \left[20902 \quad 12 \quad 7 \quad 182 \quad 3 \quad 3 \quad 32\right], \tag{13}$$

and the built STBC is

$$\mathbf{G} = \begin{bmatrix} -s_2 & -s_6^* & 0 & -s_5 \\ s_3^* & 0 & -s_6^* & -s_1^* \\ -s_1 & s_4 & 0 & -s_3 \\ s_4^* & s_1^* & -s_5 & 0 \\ 0 & s_5^* & s_1 & -s_6 \\ -s_6 & s_2^* & -s_3 & 0 \\ 0 & s_3^* & s_2 & s_4^* \\ -s_5^* & 0 & -s_4 & s_2^* \end{bmatrix}. \tag{14}$$

It should be noted that the order of operations conducted to build this STBC directed Algorithm 6 be performed prior to Algorithm 7; therefore, as explained in Section III, Example 4, $\mathbf{A}_{syms} = [1\ 3\ 4\ 2\ 5\ 6]$ for this STBC was used to permute $\mathbf{v}_{asn} = [1\ 2]$ to $\mathbf{v}_{psn} = [1\ 3]$.

V. PLS TECHNIQUE: ALTERNATING STBC SCHEME

Employing a static alternative STBC from the code set other than the original common STBC provides a certain amount of security; however, switching to a different STBC in the set after each codeword in a pseudorandom fashion undoubtedly provides additional security. Analysis of these two options is discussed in Section VIII-A. This section presents the details of such a scheme for pseudorandomly alternating between STBCs in the set. The scheme involves three phases: steady state, initialization, and data transmission. The basic elements of each phase are illustrated in Fig. 1. As this scheme is predominantly located at the physical layer, details regarding medium access, flow, and error control are not discussed and are left as a topic of future research. Downlink transmission (e.g., base station (BS) to user equipment (UE)) is the sole focus of the scheme presented in this article; however, it could readily be extended to include uplink transmission. For simplicity, the communications link described here is MISO, but this may easily be adapted for a MIMO link.

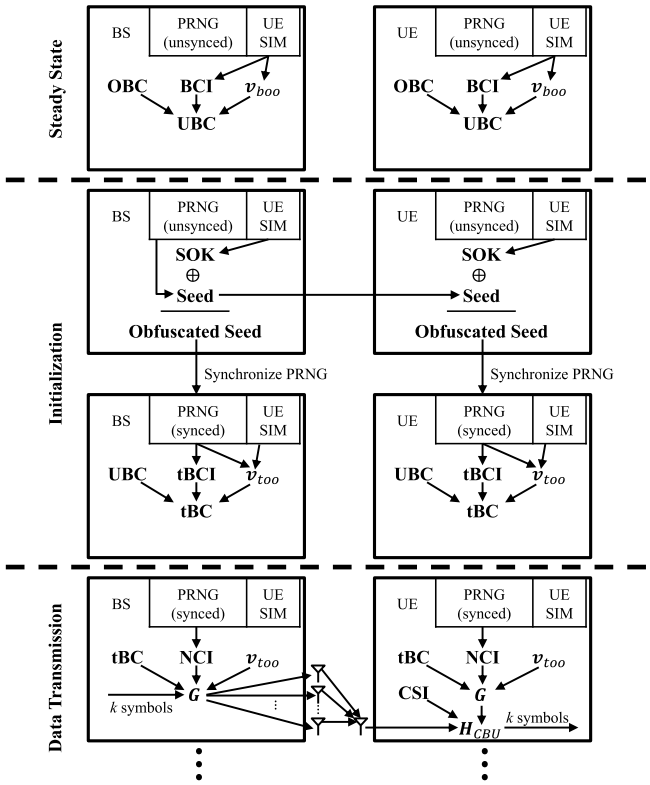

FIGURE 1. Basic elements for each phase of the alternating PLS scheme.

A. STEADY STATE

During this phase, communication has not yet begun between the BS and UE; however, it is obviously assumed that each entity has a priori knowledge of the original base code (OBC) shown in (2), the operation limits discussed in Sections III-C and IV, and the unique contents contained on the UE's subscriber identity module (SIM). These unique contents include a base code index (BCI) and a base order of operations vector ($\mathbf{v}_{boo}$). With this a priori information, both entities are able to build a unique base code (UBC) using the process detailed in Section IV.

From the set size given in (7) for MPO468 and the 7! possible $\mathbf{v}_{boo}$ vectors alone, this scheme allows for $\approx \mathbf{3.84 \cdot 10^{16}}$ unique SIMs. The SIM also includes a seed obfuscation key (SOK) used during the next phase to obfuscate the received seed to minimize any impact of the seed being intercepted. If the size of the employed SOK is 64 bits, then the number of unique SIMs may be increased by a factor of 2^{64}.

row indices							
1	3	6	2	1	7	4	5
2	3	6	2	1	7	5	4
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1638	2	6	4	3	5	7	1
1639	2	6	4	1	3	7	5
1640	2	6	4	1	3	5	7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
5039	5	4	7	1	2	3	6
5040	5	4	7	1	2	6	3

FIGURE 2. Illustration of indexing into permutations array containing elements of $\mathbf{v}_{boo}$ in lexicographic order with row index = 1639.

B. INITIALIZATION

During this phase, it is desired for communication to begin; thus, a random seed is transmitted from BS to UE using a secret sharing method such as the post-quantum cross-layer key agreement scheme presented in [2]. Each entity performs modulo-2 addition, denoted by the $\oplus$ symbol in Fig. 1, between the bits of this random seed and those of the SOK to obtain an obfuscated seed. Each entity then uses this seed to synchronize PRNGs. Next, pseudorandom numbers (PRNs) are generated to create a temporary BCI (tBCI) and temporary order of operations vector ($\mathbf{v}_{too}$) for use during this session. The value of tBCI is set as

$$\text{tBCI} = (\text{PRN} \bmod |\mathbf{S}|) + 1, \tag{15}$$

and $\mathbf{v}_{too}$ is set by indexing a row from a permutations array containing the elements from $\mathbf{v}_{boo}$ in lexicographic order, where the row index is set as

$$\text{row index} = (\text{PRN} \bmod 7!) + 1. \tag{16}$$

Example 6: For a $\mathbf{v}_{boo} = [3\ 6\ 2\ 1\ 7\ 4\ 5]$ and PRN = 278, 570, 369, 041, 398, the row index = 1639 using (16); therefore, $\mathbf{v}_{too} = [2\ 6\ 4\ 1\ 3\ 7\ 5]$ as shown in Fig. 2.

With these two session variables along with the UBC from the steady state phase and operation limits, both entities build a temporary base code (tBC) using the process detailed in Section IV. Upon completion of this step, the BS and UE are ready to begin data transmission.

C. DATA TRANSMISSION

During this phase, the synchronized PRNGs from the initialization phase are used to generate PRNs in order to determine the next STBC to be used during transmission from BS to UE. The next code index (NCI) is set as

$$\text{NCI} = (\text{PRN} \bmod |\mathbf{S}|) + 1. \tag{17}$$

Using this index value, $\mathbf{v}_{too}$, tBC, and operation limits, both entities build the next STBC using the Section IV process.

As MPO468 consists of six data symbols, the steps described here for determining the NCI and building the next STBC are repeated for each set of six data symbols to be transmitted. We now discuss a potential communications link employing this PLS scheme including the steps required by UE for reception.

D. COMMUNICATIONS LINK & RECEPTION

A potential MISO communications link between BS and UE begins with a common digital modulation scheme such as quadrature phase-shift keying (QPSK) or M-ary quadrature amplitude modulation (MQAM) to convert a binary source to data symbols. These symbols are inserted into the next pseudorandomly built STBC and transmitted across the wireless interface using c TX antennas over r symbol time periods. The r-by-1 sample vector, $\mathbf{Z}$, received by UE using a single receive (RX) antenna is

$$\mathbf{Z} = \mathbf{G} \times \mathbf{h}_{BU} + \mathbf{n}, \tag{18}$$

where $\mathbf{G}$ is the next STBC, $\mathbf{h}_{BU}$ is the c-by-1 channel tap vector between BS and UE, and $\mathbf{n}$ is an r-by-1 additive white Gaussian noise vector. UE is assumed to have perfect CSI and knowledge of the STBC used to encode each codeword. UE performs MRC to equalize and decode the data symbols and demodulates the symbols to binary data using the known digital modulation scheme.

The general MRC process applicable to all STBCs discussed in this work can be described by two steps performed in sequence to obtain the estimated data symbol vector, $\hat{\mathbf{s}}$. These steps can be seen as calculating

$$\hat{\mathbf{s}}_{int} = \mathbf{H}_{CBU}^{\dagger} \times \mathbf{Z} \tag{19}$$

followed by

$$\hat{\mathbf{s}}(x) = \frac{\hat{\mathbf{s}}_{int}(x) + \hat{\mathbf{s}}_{int}^{*}(x+k)}{\|\mathbf{h}_{BU}\|^2} \tag{20}$$

for $x \in \{1, \ldots, k\}$, where $\|\mathbf{h}_{BU}\|^2$ is the square of the Euclidean norm of $\mathbf{h}_{BU}$, $\hat{\mathbf{s}}_{int}$ is a $2k$-by-1 intermediate vector, and $\mathbf{H}_{CBU}$ is an r-by-$2k$ combining matrix corresponding to $\mathbf{G}$ such that

$$\mathbf{H}_{CBU} \times \mathbf{s}_{ext} = \mathbf{G} \times \mathbf{h}_{BU}. \tag{21}$$

The intermediate vector represents a weighted version of $\mathbf{s}_{ext}$ with noise plus interference due to spreading of data symbols; however, completion of (20) cancels all interfering terms to produce the estimated data symbol vector. This can be shown to be true according to the sufficient statistics

$$\hat{\mathbf{H}} = \|\mathbf{h}_{BU}\|^2 \cdot \mathbf{I}_k, \tag{22}$$

where

$$\begin{aligned}\hat{\mathbf{H}}(x, y) = {} & \mathbf{H}_{int}(x, y) + \mathbf{H}_{int}^{*}(x+k, y+k) \\ & + \mathbf{H}_{int}(x, y+k) + \mathbf{H}_{int}^{*}(x+k, y)\end{aligned} \tag{23}$$

for $x, y \in \{1, \ldots, k\}$, and

$$\mathbf{H}_{int} = \mathbf{H}_{CBU}^{\dagger} \times \mathbf{H}_{CBU}. \tag{24}$$

These sufficient statistics represent a systematic approach to finding k vectors to solve for each of the k data symbols

contained within any complex orthogonal STBC using MRC. This approach meets the sufficient statistics given in [28, (10.272)]. For the purpose of brevity and without loss of generality, the Alamouti code is used to provide proof for this MRC sufficient statistic and two-step sequence in Appendix B.

Without loss of generality, the process for determining the corresponding $\mathbf{H}_{CBU}$ matrix for any given complex orthogonal STBC matrix, $\mathbf{G}$, and channel tap vector, $\mathbf{h}_{BU}$, can be seen as solving the system of equations given in (21) for the terms of $\mathbf{H}_{CBU}$ with appropriate constraints. In general, the terms of $\mathbf{H}_{CBU}$ are constrained to zeros, weighted terms from $\mathbf{h}_{BU}$, and sums of these weighted terms. The use of sums of weighted terms is provided to account for this approach to be applied with the sporadic 3/4 rate STBCs given in [29] designed using the theory of amicable designs from [30]. For all STBCs discussed in this work, sums of terms are unnecessary, and the only required weights are $\in \{-1, 1\}$. As such, an algorithmic approach for construction of these MRC matrices given the channel tap vector and any STBC matrix discussed in this work is provided in Algorithm 8.

Example 7: The $\mathbf{H}_{CBU}$ matrix for the base code in (2) and channel tap vector, $\mathbf{h}_{BU} = [h_1 \;\; h_2 \;\; h_3 \;\; h_4]^T$, is given as

$$\mathbf{H}_{CBU} = \begin{bmatrix} h_1 & 0 & 0 & 0 & 0 & 0 & 0 & h_4 \\ h_2 & 0 & 0 & 0 & 0 & 0 & -h_4 & 0 \\ h_3 & 0 & 0 & 0 & 0 & h_4 & 0 & 0 \\ 0 & 0 & 0 & h_4 & h_1 & 0 & 0 & 0 \\ 0 & 0 & -h_4 & 0 & h_2 & 0 & 0 & 0 \\ 0 & h_4 & 0 & 0 & h_3 & 0 & 0 & 0 \\ 0 & h_2 & h_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & -h_1 & 0 & h_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & -h_1 & -h_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h_2 & h_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & -h_1 & 0 & h_3 \\ 0 & 0 & 0 & 0 & 0 & 0 & -h_1 & -h_2 \end{bmatrix}^T. \quad (25)$$

VI. NOMENCLATURE AND CHARACTERIZATION

Before analyzing the security of the proposed scheme, we first define some common nomenclature for use in characterization of STBC PLS schemes. In [31], definitions were provided for a cryptogram, message residue class, and cryptogram residue class pertaining to secrecy systems in general. This work intends to relate those terms as they pertain to STBC PLS schemes. Additionally, the notion and definitions for a KR and key residue class (KRC) are provided.

A. CRYPTOGRAM DEFINITION

For this work, the cryptogram, $\mathbf{E}$, is defined as the transmitted STBC matrix, $\mathbf{G}$, with embedded data symbol vector, $\mathbf{s}$, containing k specific data symbols of the chosen digital modulation scheme.

Algorithm 8 MRC Matrix Generation

Input: $\mathbf{G}$ {STBC Matrix}, $\mathbf{h}$ {Channel tap vector}
Output: $\mathbf{H}_C$ {MRC Matrix}

$\mathbf{H}_C$ = 2-dimensional array of 0's of size r-by-$2k$
for $ri = 1$ to r **do**
  for $ci = 1$ to c **do**
    if $\mathbf{G}(ri, ci) \neq 0$ **then**
      $j =$ symbol index of $\mathbf{G}(ri, ci)$
      {e.g., $j = 1$ for $\mathbf{G}(ri, ci) == s_1, -s_1, s_1^*, -s_1^*$}
      if $\mathbf{G}(ri, ci)$ is negated **then**
        $n = -1$
      else
        $n = 1$
      end if
      if $\mathbf{G}(ri, ci)$ is conjugated **then**
        $g = 1$
      else
        $g = 0$
      end if
      $\mathbf{H}_C(ri, j + g \cdot k) = n \cdot \mathbf{h}(ci)$
    end if
  end for
end for
return $\mathbf{H}_C$

Example 8: Given the MPO468 base code shown in (2) employing 16-QAM with $\mathbf{s} = [+1 + 1j, \; -3 + 3j, \; -3 - 1j, \; -1 - 1j, \; +3 - 3j, \; +3 + 1j]^T$, the cryptogram is represented as

$$\mathbf{E} = \begin{bmatrix} +1+1j & -3+3j & -3-1j & 0 \\ +3+3j & +1-1j & 0 & +3+1j \\ +3-1j & 0 & +1-1j & -3+3j \\ 0 & +3-1j & -3-3j & -1-1j \\ -1-1j & +3-3j & +3+1j & 0 \\ -3-3j & -1+1j & 0 & -3-1j \\ -3+1j & 0 & -1+1j & +3-3j \\ 0 & -3+1j & +3+3j & +1+1j \end{bmatrix}. \quad (26)$$

B. KEY RESIDUE DEFINITION

The KR is defined as the pattern or set of symbol pair relationships and their matrix locations within a given STBC. Initially discussed in Section III, when variations are constrained to original, negative, conjugate, or negative-conjugate data symbols, symbol pair relationships refer to the conjugate and negative-conjugate relationships that exist between the symbol variations within a symbol pair. For the KR, the particular symbols in each pair are irrelevant. The pattern is defined only by the locations and relationships of the symbol pair elements.

The KR may be shown in tabular form by listing the relationship and variant locations of each data symbol for all symbol pairs within the STBC. As the SPO334 STBC contains only one symbol pair between each combination of

TABLE 1. Tabular form KR showing the relationships and locations of each element for all symbol pairs in the SPO334 base code.

SPI	Relationship	Location of Variant 1	Location of Variant 2
1.1	Conj.	(1, 1)	(2, 2)
1.2	Negative-Conj.	(2, 1)	(1, 2)
2.1	Conj.	(1, 1)	(3, 3)
2.2	Negative-Conj.	(3, 1)	(1, 3)
3.1	Conj.	(1, 2)	(4, 3)
3.2	Negative-Conj.	(4, 2)	(1, 3)

its three columns, there are $\binom{3}{2} \cdot 1 = 3$ symbol pairs that make up its KR.

Example 9: The tabular form KR for the SPO334 base code shown in (1) is provided in Table 1, where the symbol pair index (SPI) is used to abstractly identify the data symbols within each symbol pair. For instance, SPI 1.2 refers to the second data symbol in the first symbol pair.

Likewise, as the MPO468 STBC contains two symbol pairs between each combination of its four columns, there are $\binom{4}{2} \cdot 2 = 12$ symbol pairs that make up its KR. The tabular form KR for the MPO468 base code is given in Appendix C.

A KR may also be represented in matrix form. Given a STBC matrix, this form of the KR may be determined by performing symbol permutations, symbol negations, and symbol conjugations. Each operation is performed as necessary on the given matrix in order to place the unconjugated and unnegated symbols in the data symbol vector, **s**, down the left-hand column in order from top to bottom.

Example 10: The matrix form KR corresponding to the SPO334 base code shown in (1) is determined by negating and conjugating symbols s_2 and s_3 and is represented as

$$\mathbf{G}_{KR} = \begin{bmatrix} s_1 & -s_2^* & -s_3^* \\ s_2 & s_1^* & 0 \\ s_3 & 0 & s_1^* \\ 0 & s_3 & -s_2 \end{bmatrix}. \tag{27}$$

Example 11: The matrix form KR corresponding to the MPO468 base code shown in (2) is determined by negating and conjugating symbols s_2, s_3, s_5, and s_6 and is represented as

$$\mathbf{G}_{KR1} = \begin{bmatrix} s_1 & -s_2^* & -s_3^* & 0 \\ s_2 & s_1^* & 0 & -s_6^* \\ s_3 & 0 & s_1^* & s_5^* \\ 0 & s_3 & -s_2 & s_4 \\ s_4 & -s_5^* & -s_6^* & 0 \\ s_5 & s_4^* & 0 & -s_3^* \\ s_6 & 0 & s_4^* & s_2^* \\ 0 & s_6 & -s_5 & s_1 \end{bmatrix}. \tag{28}$$

Depending upon the specific symbols contained in **s**, the KR may or may not be able to be precisely determined from a given **E**. This determination process can be seen as finding the STBC from the known set that could produce **E**, assuming the non-zero entries of the left-hand column represent the unconjugated and unnegated symbols in the data symbol vector, **s**, in order from top to bottom. It can clearly be seen that the KR in (28) could produce the cryptogram in (26); however, as an example to illustrate the uncertainty that exists when **s** is structured with particular patterns for multi-pair orthogonal STBCs such as MPO468, (26) represents a special case cryptogram that could have also been produced by seven alternative KRs including

TABLE 2. Relationships and locations of each element for symbol pair 7 in (28) and (29).

SPI	Relationship	Location of Variant 1	Location of Variant 2
7.1	Conj.	(1, 2)	(4, 3)
7.2	Negative-Conj.	(4, 2)	(1, 3)

$$\mathbf{G}_{KR2} = \begin{bmatrix} s_1 & s_5^* & -s_3^* & 0 \\ s_2 & -s_4^* & 0 & s_3^* \\ s_3 & 0 & s_1^* & -s_2^* \\ 0 & s_3 & s_5 & s_4 \\ s_4 & s_2^* & -s_6^* & 0 \\ s_5 & -s_1^* & 0 & s_6^* \\ s_6 & 0 & s_4^* & -s_5^* \\ 0 & s_6 & s_2 & s_1 \end{bmatrix}. \tag{29}$$

This uncertainty impacts the key equivocation in later analysis as each KR corresponds to a different KRC.

C. KEY RESIDUE CLASS DEFINITION

In [32], the term KRC was used to refer to keys which act similarly and may be indistinguishable under certain conditions. Whereas this usage of the term is compatible with our work, [32] adds that when the KRC is known, the message is also known. As this latter statement is not true for our purposes, we provide a more complete definition of KRC as it pertains to STBC PLS schemes.

A KRC is defined herein as a subset of the full code set consisting of all "keys", or STBCs, sharing the same symbol pair relationships in the same matrix locations as one another. Collectively, the KRCs form a partition of the full code set. For all STBCs discussed for this work, it can be shown that there are

$$4^k \cdot k! \tag{30}$$

STBCs within each KRC.

By this definition, all STBCs within a KRC are unique but share a common KR. As previously stated, the particular symbols within each symbol pair do not impact the KR so long as the locations and relationships are the same. For instance, the seventh symbol pair in (28) and (29) produces the same entry within their respective tabular form KRs despite the symbol pair being formed by variants of s_2 and s_3 in (28) vice s_5 and s_3 in (29). For reference, this symbol pair entry is shown in Table 2. Whereas the entry for this particular symbol pair is a match between these two KRs, not all symbol pair entries match; thus, they are not in the same KRC. This is expected, as by our definition only one KR can be associated with each KRC.

To further illustrate the concept of the KRC, the set of all STBCs within a single KRC may be built by using the corresponding KR as the base code and performing Algorithms 0, 5, 6, and 7. The input set cardinality for Algorithm 0 is set to $4^k \cdot k!$. Algorithm 6 is extended by setting $\mathbf{v}_{asp}$ to the range from 1 to $k!$ to allow for all symbol permutations to be performed, and Algorithm 7 is extended by setting $\mathbf{v}_{asn} = [1 \ \ldots \ k]$ to provide all $2^k - 1$ binary combinations of symbol negations in the same manner as Algorithm 5 provides all $2^k - 1$ combinations of symbol conjugations.

As all KRCs are of equal size and partition the full set, the number of KRCs can be determined by dividing the full set cardinality by the class size; however, this can also be determined by finding the number of unique KRs as there is a one-to-one correspondence between KRs and KRCs. In order to find the number of unique KRs, we must look at the factors that distinguish them from one another, namely the symbol pair relationships and locations. The symbol pair locations are most obviously affected by the zero placements within the STBC matrix. As there are exactly two zeros per column and one zero per row of $\mathbf{G}$ for MPO468, it can be seen that there are $\binom{8}{2} \cdot \binom{6}{2} \cdot \binom{4}{2} \cdot \binom{2}{2} = 2,520$ different permutations of zero placements. Each permutation constitutes a set of unique KRs. The symbol pair locations can also be changed by performing row permutations. To operate within a single zero placement permutation of the STBC matrix, the only permitted row permutations can be better understood as row swaps. As such, the rows of the STBC matrix having common zero locations may be swapped. For every zero placement permutation, there are $2^{r/2-1}$ permitted row swaps constituting different KRs. Finally, for each permutation of symbol pair locations, the symbol pair relationships may be modified by performing row negations. While holding a single row constant, the remaining rows may undergo any one of the 2^{r-1} binary combinations of row negations creating different KRs. All combined, for MPO468 there are

$$2,520 \cdot 2^{r/2-1} \cdot 2^{r-1} = 2,580,480 \tag{31}$$

unique KRs and by extension, KRCs. It can be seen that multiplying the class size in (30) by the number of KRCs in (31) produces the same full set cardinality shown in (7) as expected. For comparison, the Alamouti STBC and SPO334 have 2 and 192 KRCs, respectively.

D. MESSAGE AND CRYPTOGRAM RESIDUE CLASS DEFINITION

From [31], a message residue class is defined as the set of messages that might have produced a given cryptogram. Likewise from [31], a cryptogram residue class is defined as the set of cryptograms that may be produced from a given message. We supplement these definitions by adding our cryptogram definition, clarifying that the message is the embedded data symbol vector for a STBC, and adding a KRC caveat to the cryptogram residue class definition.

A message residue class is defined herein as the set of data symbol vectors that might have produced a given cryptogram. Similarly, a cryptogram residue class is the set of cryptograms that may be produced from a given data symbol vector by employing only those STBCs from a single KRC. This caveat regarding the KRC is important as each KRC has different cryptogram residue classes matching the KR for that class. The uncertainty of the KR for particular cryptograms discussed previously in Section VI-B for the MPO468 code is due to the fact that these special case cryptograms exist within cryptogram residue classes for each of the eight associated KRCs.

E. CHARACTERIZATION OF RESIDUE CLASSES

When employing only the STBCs from a single KRC of a chosen STBC, it can be shown that the message and cryptogram residue classes meet the properties for those of a pure cipher system in [31, Th. 3]. For clarity of discussion, a summary and adaptation of these properties is provided here:

Property 1) Message residue classes $\mathbf{C}_1$, $\mathbf{C}_2$, etc. collectively form a partition of the data symbol vector space for a given digital modulation scheme. Similarly for the cryptogram residue classes $\mathbf{C}'_1$, $\mathbf{C}'_2$, etc.

Property 2) Embedding any data symbol vector in $\mathbf{C}_i$ within any STBC of the chosen KRC produces a cryptogram in $\mathbf{C}'_i$. Decoding any cryptogram in $\mathbf{C}'_i$ with any STBC of the chosen KRC leads to a data symbol vector in $\mathbf{C}_i$.

Property 3) The number of data symbol vectors in $\mathbf{C}_i$, say ϕ_i, is equal to the number of cryptograms in $\mathbf{C}'_i$ and is a divisor of the number of STBCs in the chosen KRC shown in (30).

Property 4) Each data symbol vector in $\mathbf{C}_i$ can be embedded within exactly f different STBCs of the chosen KRC to produce each cryptogram in $\mathbf{C}'_i$, where f is defined such that

$$\phi_i = \frac{4^k \cdot k!}{f}. \tag{32}$$

This f value is a repetition metric for the data symbol vectors within each message residue class. To allow for further examination of this metric, we first define a unique symbol set. For any chosen digital modulation scheme, a unique symbol set is herein defined as a set of four symbols that are equal to the negative, conjugate, or negative-conjugate of one another; thus, 16-QAM contains four unique symbol sets whereas 64-QAM contains 16 unique symbol sets. For any modulation scheme of order M with constellation exhibiting bilateral symmetry across both the real and imaginary axes, the number of unique symbol sets is equal to $M/4$.

Example 12: The four unique symbol sets contained within 16-QAM are shown in Fig. 3 and distinguished by different colors and shapes.

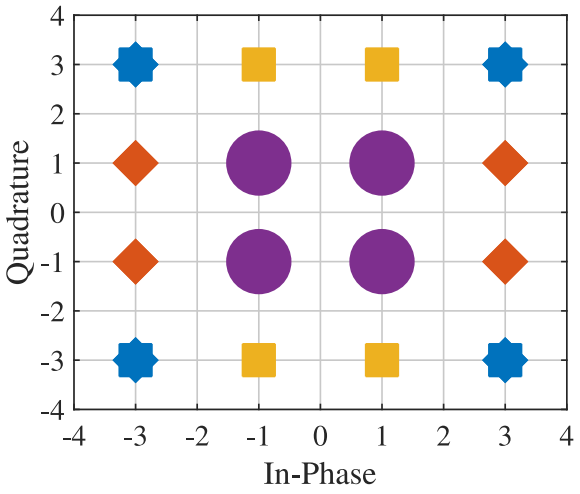

FIGURE 3. The four unique symbol sets for 16-QAM.

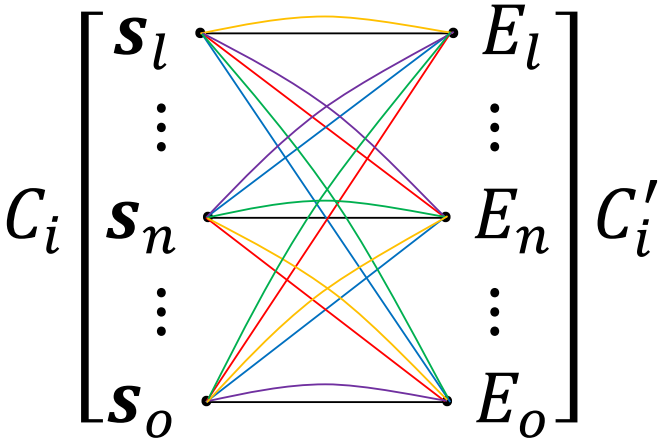

FIGURE 4. After [31], template for Alamouti message and cryptogram residue classes with $f = 2$ employing 16-QAM data symbols and STBCs from a single KRC.

With the unique symbol set definition established, the repetition metric, or f value, for each message and cryptogram residue class can be further defined as

$$f = \prod_{i=1}^{u} n_i!, \tag{33}$$

where u is the number of unique symbol sets represented in each of the data symbol vectors of the message residue class, and $\mathbf{n} = [n_1 \ \ldots \ n_u]^T$ is a u-by-1 column vector containing the number of times a symbol appears in each data symbol vector from each of the unique symbol sets represented.

Example 13: For an MPO468 message residue class containing a 16-QAM data symbol vector represented as

$$\mathbf{s}_i = \begin{bmatrix} +1+1j \\ +3+3j \\ +3-1j \\ -1-1j \\ -3-3j \\ -3+1j \end{bmatrix}, \tag{34}$$

$u = 3$, $\mathbf{n} = [2 \ 2 \ 2]^T$, and $f = 2!{\cdot}2!{\cdot}2! = 8$; thus, substituting this result into (32) gives a message residue class size of

$$\phi_i = \frac{4^k \cdot k!}{f} = \frac{4^6 \cdot 6!}{8} = 368,640. \tag{35}$$

F. RESIDUE CLASSES FOR THE ALAMOUTI STBC

To better understand the message and cryptogram residue classes pertaining to STBC PLS schemes, we now examine these classes for the Alamouti STBC employing 16-QAM data symbols and only the STBCs from one of its two KRCs. For each message and cryptogram residue class, with $k = 2$ and $M = 16$, either $u = 1$ or $u = 2$ leading to $\mathbf{n} = [2]$ or $\mathbf{n} = [1 \ 1]$, respectively; thus, for the Alamouti STBC, the repetition metric for all message and cryptogram residue classes is either $f = 2$ or $f = 1$. Note that $2 = k!$ in this case.

For any chosen base code, the number of data symbol vectors collectively contained within all message residue classes with a repetition metric of $f = k!$ can easily be understood by determining the number of potential data symbol vectors that may be produced by selecting all k data symbols from within a single unique symbol set. This selection process can be further explained as selecting any of the M modulation symbols to be the first data symbol followed by selecting any of the 4 variations (e.g., original, negative, conjugate, or negative-conjugate) of the first modulation symbol for the remaining $k - 1$ data symbols. By this process, there are

$$M \cdot 4^{k-1} \tag{36}$$

data symbol vectors within message residue classes with $f = k!$. By substituting $k!$ for f in (32), it can be seen that there are exactly

$$\phi_i = \frac{4^k \cdot k!}{k!} = 4^k \tag{37}$$

data symbol vectors in each of these classes; therefore, combining (36) and (37), there are

$$\frac{M \cdot 4^{k-1}}{4^k} = \frac{M}{4} \tag{38}$$

message residue classes and associated cryptogram residue classes with a repetition metric of $f = k!$. This is an intuitive solution as each of these classes represents the set of all data symbol vectors that can be produced by selecting all k data symbols from within a single unique symbol set. As there are $M/4$ unique symbol sets, there are $M/4$ message and cryptogram residue classes with $f = k!$.

For the Alamouti STBC employing 16-QAM data symbols, there are $16/4 = 4$ classes with a repetition metric of $f = 2$. Fig. 4 shows a template for these classes with the following variable definitions. Since there are 4 classes based upon this template with $\phi_i = 4^k = 16$ data symbol vectors and cryptograms in each class, $i \in \{1, 2, 3, 4\}$, $l = 16 \cdot (i - 1) + 1$, $o = l + 15$, and $l < n < o$. The multi-colored lines connecting each data symbol vector on the left-hand side to a cryptogram on the right-hand side represent 6 of the 32 STBCs within the chosen KRC. The

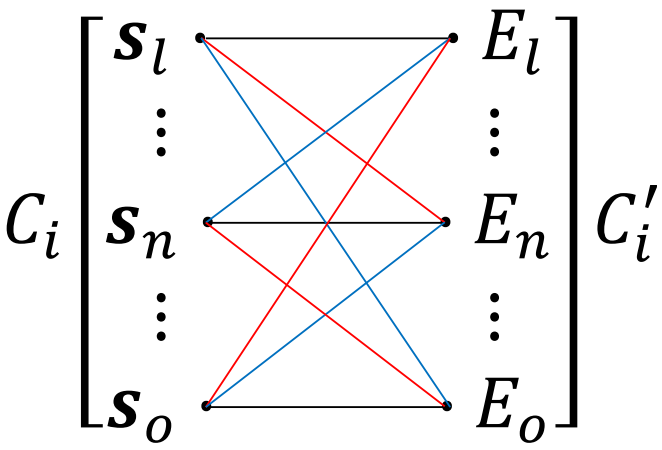

FIGURE 5. After [31], template for Alamouti message and cryptogram residue classes with $f = 1$ employing 16-QAM data symbols and STBCs from a single KRC.

particular colors serve to aid in distinguishing between different STBCs, to illustrate Properties 2 and 4 for the message and cryptogram residue classes of a pure cipher system given in Section VI-E, and to illustrate that a given STBC can not produce a cryptogram in $\mathbf{C}_i'$ from more than one data symbol vector in $\mathbf{C}_i$. Any other pattern or relation that can be found within the chosen colors is unintended and does not represent a known property of these classes.

For any chosen base code, the number of data symbol vectors collectively contained within all message residue classes with a repetition metric of $f = 1$ can be understood as the set of all data symbol vectors produced by selecting all k data symbols from different unique symbol sets. This is only possible when $k < M/4$. Given that this condition is met, there are

$$\prod_{i=0}^{k-1}(M - 4 \cdot i) \tag{39}$$

data symbol vectors within message residue classes with a repetition metric of $f = 1$. By substituting 1 for f in (32), it can be seen that there are exactly

$$\phi_i = \frac{4^k \cdot k!}{1} = 4^k \cdot k! \tag{40}$$

data symbol vectors in each of these classes.

For the Alamouti STBC employing 16-QAM data symbols, combining (39) and (40) provides that there are

$$\frac{M \cdot (M-4)}{4^k \cdot k!} = \frac{16 \cdot 12}{4^2 \cdot 2!} = 6 \tag{41}$$

message residue classes and associated cryptogram residue classes with a repetition metric of $f = 1$. Fig. 5 shows a template for these classes with the following variable definitions. Since there are 6 classes based upon this template with $\phi_i = 4^k \cdot k! = 32$ data symbol vectors and cryptograms in each class, and 4 previously defined classes with $f = 2$ with a total of $4 \cdot 16 = 64$ data symbol vectors and cryptograms combined, $i \in \{5, 6, 7, 8, 9, 10\}$, $l = 32 \cdot (i-5) + 65$, $o = l + 31$, and $l < n < o$. The multi-colored lines serve the same purpose as those in Fig. 4.

It can be seen that Property 1 in Section VI-E is confirmed as these ten message residue classes collectively partition the data symbol vector space for the Alamouti STBC employing 16-QAM with a total of $M^k = 16^2 = 256$ data symbol vectors. Likewise for the cryptogram space matching the KR of the chosen KRC.

VII. INFORMATION-THEORETIC SECURITY AND BIT ERROR RATE ANALYSIS

We now use the established nomenclature to analyze the information-theoretic security of the proposed alternating STBC PLS scheme from the perspectives of message and key equivocation. Theoretical expected BER for a passive eavesdropper is then discussed and plotted along with Monte Carlo simulations for confirmation. First we describe our eavesdropper assumptions and a method Eve may take to obtain information about the embedded data symbol vector and STBC in use for each codeword employing only a single RX antenna. Additional RX antennas and alternative methods may be employed by the eavesdropper; however, this approach shows what may be achieved with minimal additional hardware. Under our worst case analysis, additional antennas prove to be of no added value.

A. EAVESDROPPER ASSUMPTIONS

For the communications link described in Section V-D, it is assumed that Eve:

- has a priori knowledge of the chosen OBC and digital modulation scheme employed,
- may have a priori knowledge of some plaintext data,
- employs 1 RX antenna,
- has perfect CSI of the channel between BS and Eve denoted by the c-by-1 channel tap vector $\mathbf{h}_{BE}$,
- has perfect timing synchronization,
- passively records all received data samples for an entire communications session,
- and has the ability to generate the matrix form KR, $\mathbf{G}_{KR}$, associated with all KRCs for the chosen OBC.

We point out that these are non-trivial assumptions, i.e., that significant knowledge is somehow available a priori to Eve and substantial operations are required for perfect CSI and synchronization.

B. EAVESDROPPER ATTACK METHODOLOGY

With these non-trivial assumptions, Eve may gain information about the embedded data symbol vector and STBC employed for each transmitted cryptogram from the recorded sample vector, $\mathbf{Z}$, by performing the following steps:

1) For each KRC of the OBC, use $\mathbf{G}_{KR}$ along with the channel tap vector, $\mathbf{h}_{BE}$, to determine the corresponding MRC matrix, $\mathbf{H}_{CBE}$.
2) Perform MRC of the received sample vector using the $\mathbf{H}_{CBE}$ matrix associated with each KRC to obtain results, $\hat{\mathbf{s}}_i$ for $i \in \{1, \ldots, l\}$ where $l = 2,580,480$ is the number of KRCs given in (31).

3) Determine the KR of the KRC to which the STBC used for transmission of the cryptogram belongs. This is done by finding the index value, i, of the result, $\hat{\mathbf{s}}_i$, with the minimum Euclidean distance to any one of the potential data symbol vectors, $\mathbf{s} \in \mathbb{M}^k$, where $\mathbb{M}$ is the set of all symbols in the M-ary modulation scheme. This can be represented as

$$i = \underset{\forall i \in \{1,\ldots,l\}}{\operatorname{argmin}} \left(\min_{\forall \mathbf{s} \in \mathbb{M}^k} d\left(\mathbf{s}, \hat{\mathbf{s}}_i\right) \right) \tag{42}$$

where $d(x, y)$ represents the Euclidean distance between x and y.

4) Obtain a coded symbol vector, $\mathbf{s}_c$, by selecting the data symbol vector, $\mathbf{s}$, with the minimum Euclidean distance to the determined result, $\hat{\mathbf{s}}_i$, from Step 3 as represented by

$$\mathbf{s}_c = \underset{\forall \mathbf{s} \in \mathbb{M}^k}{\operatorname{argmin}}\, d\left(\mathbf{s}, \hat{\mathbf{s}}_i\right). \tag{43}$$

For sufficient signal-to-noise ratio (SNR), Step 3 generally provides a single KR determination; however, for the MPO468 STBC and special case cryptograms discussed previously in Section VI-B, eight MRC results should exhibit approximately the same Euclidean distance to a potential data symbol vector. For this case, Eve may use any one of these eight results for $\hat{\mathbf{s}}_i$ in Step 4 as it can be shown that each one yields a coded symbol vector in an equivalent message residue class from a different KRC. By equivalent, we mean that the message residue classes from these different KRCs have the exact same data symbol vectors contained within. It is only the corresponding cryptogram residue classes that are different for each of these KRCs.

From this point forward, the worst case is analyzed by assuming that Eve has been able to successfully obtain a coded symbol vector within the correct message residue class for each codeword of the entire recorded communications session. Likewise, it is assumed that Eve has successfully obtained the correct KR, or set of eight KRs, for each codeword. Although it requires that Eve perform l times the number of MRC steps as the intended receiver as well as a comparison of those l results for each codeword, where l is the number of KRCs, this case is a likely eventuality with the assumptions provided in Section VII-A validated and with sufficient SNR and time.

C. MESSAGE EQUIVOCATION

Having obtained a coded symbol vector within the correct message residue class, the amount of uncertainty remaining about the transmitted data symbol vector within each codeword is referred to as the equivocation of the message [31]. This equivocation is represented as

$$H(\mathbf{s}|\mathbf{s}_c) = \log_2\left(\frac{4^k \cdot k!}{f}\right) \tag{44}$$

where the argument of the logarithm is the size of the message residue class, ϕ_i, to which the coded symbol vector belongs. We can determine ϕ_i using (32) and (33) from Section VI-E.

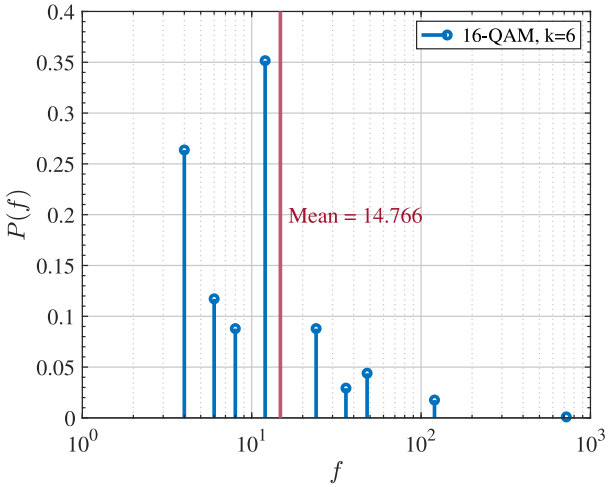

FIGURE 6. Probability mass function of the repetition metric, f, for 16-QAM with $k = 6$.

In order for Eve to obtain an estimate of the data symbol vector, $\hat{\mathbf{s}}$, Eve has two equivalent techniques available to decode $\mathbf{s}_c$. First, Eve could create the message residue class to which the coded symbol vector belongs and randomly select one of the data symbol vectors from the class to be $\hat{\mathbf{s}}$ as they are all equally probable. Alternatively, $\hat{\mathbf{s}}$ may be obtained by performing *symbol assignment* and *variation decision* operations to decode $\mathbf{s}_c$ as discussed in [17]. For clarity of discussion, these operations are repeated here:

1) "Symbol assignment is performed by assigning each of the symbols in $\mathbf{s}_c$ to a given data symbol, e.g., s_1, in the transmitted data symbol vector, $\mathbf{s}$ [17]."
2) "Variation decision is performed by deciding whether the symbol in the coded vector represents the original, negative, conjugate, or negative-conjugate variation of the assigned data symbol, e.g., s_1, $-s_1$, s_1^*, or $-s_1^*$ [17]."

The amount of uncertainty in either decoding process is equivalent and represented in (44).

To determine the mean and upper bounds on message equivocation, the probability mass function of the repetition metric, f, is required for all MQAM schemes evaluated. These mass functions are empirically derived by first calculating the value of f, according to (33), for all M^k possible data symbol vectors for a given M-order modulation scheme and value of k. Assuming equally probable data symbol vectors, the probability of each value of f is calculated by counting the number of vectors resulting in each value of f and dividing that count by the total number of vectors, M^k. These mass functions and mean value of the repetition metric for $k = 6$ are provided in Figs. 6 and 7. For QPSK, it is guaranteed that $f = k!$ as QPSK contains only a single unique symbol set. Message equivocation for various STBCs with k symbols per codeword employing QPSK, 16-QAM, and 64-QAM can be seen in Fig. 8.

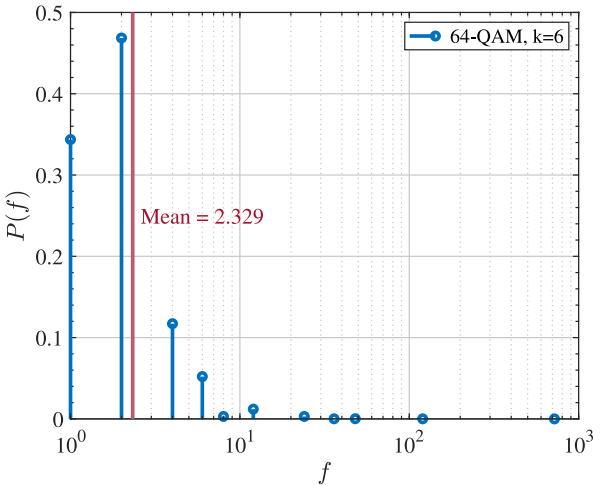

FIGURE 7. Probability mass function of the repetition metric, f, for 64-QAM with $k = 6$.

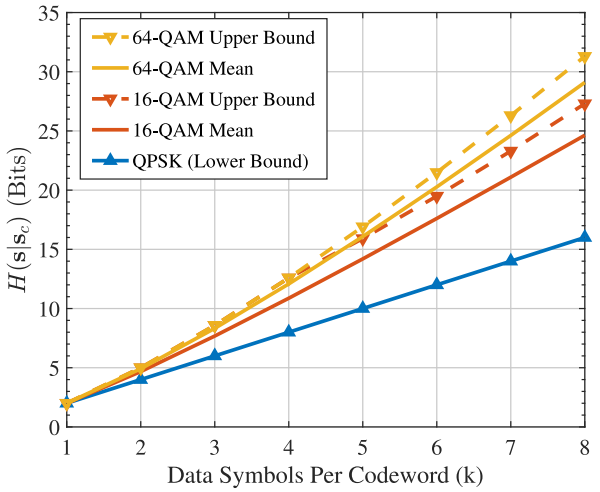

FIGURE 8. Message equivocation given STBCs with various number of data symbols per codeword and modulation schemes.

The amount of information Eve gains about the message by obtaining $\mathbf{s}_c$ is the mutual information between $\mathbf{s}$ and $\mathbf{s}_c$, represented as

$$\begin{aligned} I(\mathbf{s};\mathbf{s}_c) &= H(\mathbf{s}) - H(\mathbf{s}|\mathbf{s}_c) \\ &= k \cdot \log_2(M) - \log_2\left(\frac{4^k \cdot k!}{f}\right), \end{aligned} \tag{45}$$

where $H(\mathbf{s})$ is the entropy in $\mathbf{s}$ before obtaining $\mathbf{s}_c$ which is equal to the number of bits per codeword. When QPSK is used, $M = 4$, $f = k!$, $I(\mathbf{s};\mathbf{s}_c) = 0$, and Eve gains no information about the message by obtaining the coded symbol vector. Mutual information between $\mathbf{s}$ and $\mathbf{s}_c$ for various STBCs with k symbols per codeword employing QPSK, 16-QAM, and 64-QAM can be seen in Fig. 9.

D. KEY EQUIVOCATION

Having determined the correct KR for the transmitted cryptogram from Step 3 in Section VII-B, the amount of

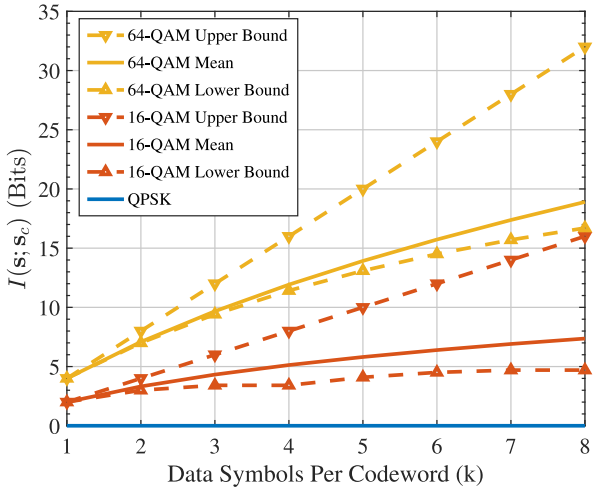

FIGURE 9. Mutual information between s and $\mathbf{s}_c$ given STBCs with various number of data symbols per codeword and modulation schemes.

uncertainty remaining about the STBC used for transmission of each cryptogram is called the equivocation of the key [31]. If Step 3 provided a single KR determination, then only the STBCs from the corresponding KRC may have been used to produce the transmitted cryptogram. For the Alamouti and SPO334 STBCs, this is always true. For this case, the key equivocation is minimized and represented as

$$H(\mathbf{G}|\mathbf{G}_{KR})_{min} = \log_2(4^k \cdot k!), \tag{46}$$

where $4^k \cdot k!$ is the KRC size given in (30).

For the MPO468 STBC, when the symbols in the embedded data symbol vector are patterned such that all rows within the transmitted cryptogram are equal to, or the negative of, another row in the cryptogram, any one of the STBCs within eight KRCs is equally likely to have produced the cryptogram. The STBCs within these eight KRCs are differentiated by row swaps and a particular row negation pattern that corresponds to the exhibited data symbol vector pattern. For these special case cryptograms, Step 3 in Section VII-B determines the eight KRs corresponding to these KRCs. For this case, the key equivocation is maximized and represented as

$$H(\mathbf{G}|\mathbf{G}_{KR})_{max} = \log_2(8) + \log_2\left(4^k \cdot k!\right). \tag{47}$$

For equally likely modulation symbols, this special case can be shown to occur with probability $(2 \cdot M^3)/M^6 = 2 \cdot M^{-3}$ for the MPO468 STBC with $k = 6$. Again, note that these special case cryptograms do not occur when employing the Alamouti or SPO334 STBCs.

For the MPO468 STBC, these two cases combine for an expected key equivocation of

$$H(\mathbf{G}|\mathbf{G}_{KR})_{exp} = \log_2\left(1 + 14 \cdot M^{-3}\right) + \log_2\left(4^k \cdot k!\right) \tag{48}$$

which is approximately equal to $H(\mathbf{G}|\mathbf{G}_{KR})_{min}$ for any value of M of practical concern.

TABLE 3. Entropy in G, key equivocation, and mutual information between G and $\mathbf{G}_{KR}$ for Alamouti, SPO334, and MPO468 STBCs.

STBC	$H(\mathbf{G})$	$H(\mathbf{G}\|\mathbf{G}_{KR})$	$I(\mathbf{G};\mathbf{G}_{KR})$
Alamouti	6 bits	5 bits	1 bit
SPO334	16.17 bits	8.59 bits	7.59 bits
MPO468	42.8 bits	21.5 to 24.5 bits	18.3 to 21.3 bits

The amount of information Eve gains about the key by obtaining the KR is the mutual information between $\mathbf{G}$ and $\mathbf{G}_{KR}$, represented as

$$I(\mathbf{G};\mathbf{G}_{KR}) = H(\mathbf{G}) - H(\mathbf{G}|\mathbf{G}_{KR}), \tag{49}$$

where

$$H(\mathbf{G}) = \log_2(|\mathbf{S}|) \tag{50}$$

is the entropy in $\mathbf{G}$ before obtaining the KR in bits. For each of the three primary STBCs discussed in this work, the values or ranges of values of $H(\mathbf{G})$, $H(\mathbf{G}|\mathbf{G}_{KR})$, and $I(\mathbf{G};\mathbf{G}_{KR})$ are provided in Table 3.

E. EXPECTED EAVESDROPPER BIT ERROR RATE

Although message equivocation gives one perspective of uncertainty, it does not offer a complete understanding of the expected BER of an eavesdropper when decoding the coded symbol vector, $\mathbf{s}_c$. Permuting the symbols within these vectors has a different effect on the expected number of bit errors depending upon the specific symbols in the vector. Every message residue class with $f = k!$ has a common expected BER of $1/b$, where $b = \log_2(M)$ for a typical Gray-coded MQAM scheme of order M. However, there are different expected BERs for other message residue classes depending upon the symbols within each data symbol vector of each class. In [17], it was stated without proof that the expected BER is

$$\text{BER}_{\text{exp}} = \frac{1 + (k-1)\cdot\frac{b}{2}}{k \cdot b}. \tag{51}$$

This represents the average expected BER of all message residue classes. Theoretical proof of (51) is provided in Appendix D. A plot of (51) for various STBCs with k symbols per codeword employing various modulation schemes can be seen in Fig. 10. Additionally, 100,000 Monte Carlo simulations were performed for the three primary STBCs discussed in this work employing QPSK, 16-QAM, 64-QAM, and 256-QAM with energy per bit to noise power spectral density ratio ($\text{E}_\text{b}/\text{N}_0$) values of 5, 10, 15, and 20 dB, respectively. For each simulation, the worst case assumption that Eve has obtained a coded symbol vector within the correct message residue class is enforced. The results of these simulations are included in Fig. 10.

VIII. ATTACK ANALYSIS AND COST-BENEFIT COMPARISON

We now look at the different approaches or models that Eve may take to obtain a correct output data frame given the initial assumptions and approach from Sections VII-A and VII-B. We then discuss the cost of our proposed PLS scheme. Finally, we compare the cost and attack complexity of our scheme to that of two alternative techniques.

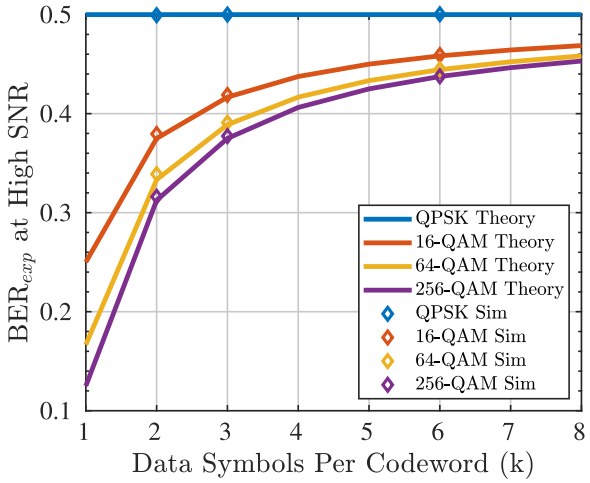

FIGURE 10. Theoretical plot of (51) with various number of data symbols per codeword and modulation schemes including high SNR Monte Carlo simulations for the three primary STBCs discussed herein.

A. STATIC VS. ALTERNATING STBC ATTACK MODELS

After completing the four step approach given in Section VII-B, Eve may use the key and message equivocation for each codeword to search for the correct output data frame.

1) STATIC STBC

If a static STBC from the set is employed throughout the entire recorded communications session, Eve could relatively easily obtain the correct output data frame using the key equivocation. After performing the four step approach for only a few codewords, Eve could determine the KRC of the employed STBC with confidence. Upon determination of the KRC, the correct output data frame is one of $4^k \cdot k!$ possibilities corresponding to the outputs produced by performing the two-step MRC sequence of all received codewords for all STBCs within the determined KRC.

Using this attack model, having known plaintext could further reduce the number of possible output data frames down to the repetition metric, f, as shown in (33). The value of the repetition metric depends upon the data symbol vectors contained within the known plaintext portion of the transmission. This would require solving for the transmitted cryptogram from the received sample vector and determining the f STBCs within the KRC capable of producing the transmitted cryptogram from the known data symbol vector. Solving for the transmitted cryptogram may not be possible with only a single RX antenna depending upon the symbols within the known data symbol vector.

2) ALTERNATING STBC

The complexity of the attack model used for the static STBC increases exponentially by employing the proposed

TABLE 4. Number of codewords and possible output frame combinations for a given frame length when performing a search using the message equivocation of each codeword.

Modulation	f_{mean}	# Codewords	Possible Combinations
QPSK	720	1012	$5.11 \cdot 10^{3655} = 2^{1518 \cdot 8}$
16-QAM	14.766	506	$1.04 \cdot 10^{2682}$
64-QAM	2.329	338	$4.50 \cdot 10^{2062}$

alternating STBC scheme discussed in Section V. Without knowledge of the SIM unique contents and a synchronized PRNG, selection of the STBC used for transmission of each codeword appears to be independent from the perspective of the attacker. Thus, Eve could repeat the four step approach for every codeword of the recorded communications session and perform a search for the correct output data frame using the message equivocation. This search involves all possible combinations of the $4^k \cdot k!/f$ unique data symbol vectors in the determined message residue class of each received codeword. For a frame of n codewords when using this approach, the correct output data frame is one of $(4^k \cdot k!/f_{mean})^n$ possible combinations.

Example 14: For a single data frame of length corresponding to a common Ethernet maximum frame length of $1,518$ bytes [33], the number of codewords and possible output frame combinations when attempting to search for the correct output data frame using the message equivocation of each codeword is shown in Table 4 for various modulations.

Using this attack model, having known plaintext yields little benefit as Eve is unable to determine the NCIs corresponding to the STBCs used to transmit the known data. Without knowledge of the NCIs, tBC, and $\mathbf{v}_{too}$, Eve is unable to improve upon the search complexity for the codewords received either before or after the known plaintext portion of the transmission.

For a transmission longer than ≈ 10 codewords, the least complex attack in the case of alternating STBCs, to our knowledge, is for Eve to perform a brute force search for the tBC, $\mathbf{v}_{too}$, and PRNG state at the beginning of the data transmission phase. For later comparison purposes, this approach is referred to as the exhaustive key search [34]. As stated in Section V-A for the number of unique SIMs when employing the MPO468 STBC and all seven operations, there are $\left|\mathbf{S}\right| \cdot 7! \approx 3.84 \cdot 10^{16}$ unique combinations of the tBC and $\mathbf{v}_{too}$. If the PRNG employed has a state space of 64 bits, then using this attack model, the correct output data frame is one of $2^{64} \cdot \left|\mathbf{S}\right| \cdot 7! \approx 7.08 \cdot 10^{35}$ possibilities corresponding to the outputs produced by performing the reception steps detailed in Sections V-C and V-D for all received codewords for all possible combinations of tBC, $\mathbf{v}_{too}$, and PRNG state at the beginning of the data transmission phase.

B. COMPLEXITY COST OF PROPOSED PLS SCHEME

When considering complexity of the proposed PLS scheme, costs are separated into realtime and non-realtime, where realtime costs refer to those which may not be completed in advance of transmission. Depending upon implementation, STBCs for data transmission may be built in advance and queued for use after the initialization phase has completed. For common time-division duplex (TDD) medium access schemes, there is likely sufficient time to build all STBCs in advance of transmission. When this is the case, all realtime costs are considered negligible. When discussing non-realtime costs, those occurring only once per communications session are considered negligible as well; thus, only recurring non-realtime costs of the data transmission phase are discussed.

Whereas the set building algorithms provided in Section III-C and Appendix A are intended to illustrate a potential method of implementation, they are not optimized for any specific platform. Execution time for these algorithms may differ greatly depending upon the capabilities of the chosen platform and specific implementation; however, we present a simplistic complexity cost associated with the proposed PLS scheme based upon average number of operations required to build each STBC to aid in comparison with related works.

The first recurring cost for this scheme is that of obtaining a PRN and performing the modulo operation given in (17) to obtain the NCI for each STBC. The iterations vector is then calculated as shown in (9). Combined, these costs represent one PRN generation, $l+1$ floor division operations, and l modulo operations, where $l = 7$ when all provided algorithms are performed. If the chosen OBC is the Alamouti or SPO334 STBC, then the two new algorithms presented in Section III may not be performed, and $l = 5$.

Lastly, there is the cost associated with performing each operation included within $\mathbf{v}_{too}$. It is assumed, on average, that each negation and conjugation algorithm is performed on 50% of the rows, columns, and symbols upon which they may operate. Additionally, it is assumed that all symbol-wise operations are performed on the data symbol vector prior to embedding within the chosen STBC to minimize complexity. Row, column, and symbol permutation operations are considered to add negligible complexity as they involve simple reordering operations that are easily optimized when storing the STBC in the queue. Thus, to build each STBC, the average number of operations includes $r/2$ row negations, $(c-1)/2$ column negations, $|\mathbf{v}_{asn}|/2$ symbol negations, and $k/2$ symbol conjugations. Without loss of generality, negation of a single row involves negating c matrix elements, and negation of a single column involves negating r matrix elements. On the contrary, individual symbol negation and conjugation operations involve operations on a single symbol within the data symbol vector. On average, the number of element-wise operations required to build a STBC using this process is

$$\frac{r \cdot (2 \cdot c - 1) + |\mathbf{v}_{asn}| + k}{2}. \tag{52}$$

C. COST-BENEFIT COMPARISON

We now compare the cost and attack complexity of the proposed scheme with two related techniques. Assuming similar applied

TABLE 5. Comparison of costs and exhaustive key search attack complexity for the proposed PLS scheme and two related techniques.

Approach	STBC	Operations Cost Per STBC Due to Alternating	Average Element-Wise Operations Cost Per STBC	Attack Complexity
Alternating STBC	Alamouti	1 PRN, 6 floor division, and 5 modulo	4	$2^{64} \cdot 64 \cdot 5!$
	SPO334	1 PRN, 6 floor division, and 5 modulo	11.5	$2^{64} \cdot 73728 \cdot 5!$
	MPO468	1 PRN, 8 floor division, and 7 modulo	32	$2^{64} \cdot 7610145177600 \cdot 7!$
Phase Rotation [15]	Alamouti	2 PRNs and 2 masking	4	2^{64}
	Octonion	4 PRNs and 4 masking	16	2^{64}
Hidden OSTBC [16]	Rate 1	N/A	64	2^{8}
	Rate 1/8		4096	2^{64}

optimizations, only recurring non-realtime costs are discussed. For ease of comparison, we use the exhaustive key search to analyze attack complexity of each scheme.

1) PHASE ROTATION SCHEME

In [15], RSSI was used as a secret key to seed a PRNG used to generate phase rotations for each TX antenna of a MIMO system after encoding with the Alamouti STBC or the 3/4 rate Octonion orthogonal STBC.

The first recurring cost for this scheme is that of obtaining a PRN for each of the c TX antennas and performing a masking operation to index into a set of L phase rotations. For each antenna, all elements in the corresponding column of the STBC are multiplied by the indexed phase rotation; thus, a combined $r \cdot c$ element-wise operations are required for each STBC using this technique.

To conduct the exhaustive key search attack for this scheme, the search space is that of the state space of the chosen PRNG. It was remarked in [15] that allowed phase rotation angles for MQAM constellations include $\theta = \{0, \pi/2, \pi, 3\pi/2\}$ to avoid increasing the PAPR; thus, with $L = 4$, only 2 bits of each PRN are masked to index the phase rotation for each antenna. With this knowledge, there may be methods to improve the exhaustive key search attack; however, we assume that no improvement exists for ease of comparison.

2) HIDDEN OSTBC SCHEME

In [16], a technique named "hidden OSTBC" was introduced where a pre-shared secret pseudorandom antipodal sequence was applied across the TX antennas of a MIMO system after encoding with a real-valued orthogonal STBC. The primary STBC considered in [16] is a rate 1 real-valued STBC consisting of eight symbols transmitted over eight symbol time periods using eight TX antennas. A few extensions of this STBC were offered up to a rate 1/8 real-valued orthogonal STBC consisting of eight symbols transmitted over 64 symbol time periods using 64 TX antennas. In [16], it was stated that the same pseudorandom sequence could be used for multiple transmissions; thus, this technique is considered to employ a static STBC throughout the entire communications session.

The first and only recurring cost for this scheme is that of multiplying all r elements in the c columns corresponding to the antenna data streams with the appropriate elements from the antipodal sequence; thus, this process involves a combined $r \cdot c$ element-wise operations for each STBC.

To conduct the exhaustive key search attack for this scheme, the search space is the number of possible pseudorandom antipodal sequences. The correct output data frame is one of 2^c possibilities corresponding to the outputs produced when performing the combining rule given in [16] of all received codewords for all 2^c possible pseudorandom sequences.

3) COMPARISON SUMMARY

A summary of the cost and attack complexity comparison is provided in Table 5. Whereas the total operations costs per STBC are roughly comparable across the schemes, the attack complexity of the proposed scheme is at least $64 \cdot 5!$ times greater than that of the next closest technique. By employing the MPO468 STBC, the attack complexity of the proposed scheme is $\approx 3.84 \cdot 10^{16}$ times greater than that of the next closest technique. Thus, far greater security is possible by employing the proposed scheme.

IX. CONCLUSION

In this work, multiple contributions in terms of PLS with the use of STBCs were presented. Algorithms from [17] were adapted and two additional algorithms included to build a much larger code set using the MPO468 base code. An efficient and practical technique to build individual STBCs from the code set using the set index was demonstrated. A PLS scheme was proposed to alternate the STBC in use over a MISO or MIMO communications link throughout a communication session. An algorithm was given for the intended receiver to update the MRC matrix in use as the STBC alternates. Definitions were provided for cryptograms, KRs, KRCs, and message and cryptogram residue classes pertaining to STBC PLS schemes. Information-theoretic security of the proposed PLS scheme including message and key equivocation was analyzed. Theoretical expected BER for a passive eavesdropper was plotted along with Monte Carlo simulations. The difference between the security offered by use of a static versus alternating STBC was discussed showing that far superior security is available by alternating the STBC used for each codeword. Cost and attack

Algorithm 0 Initialization

Inputs: $\mathbf{G}$ {Base STBC}, $|\mathbf{S}|$ {Set cardinality},
$\mathbf{K}_{perm}$ {Symbol permutation of base STBC}
Outputs: $\mathbf{S}$ {Set of STBCs},
PC_o {# of populated codes in output $\mathbf{S}$},
$\mathbf{A}_{syms}$ {Symbols record array}

$PC_o = 1$
$\mathbf{S}$ = 3-dimensional array of 0's of size r-by-c-by-$|\mathbf{S}|$
$\mathbf{S}(:,:,1) = \mathbf{G}$ {Populate set index 1 with Base STBC}
for $m = 1$ to $|\mathbf{S}|$ **do**
  $\mathbf{A}_{syms}(m,:) = \mathbf{K}_{perm}$
  {Initialize each row of $\mathbf{A}_{syms}$ with $\mathbf{K}_{perm}$}
end for
return $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

Algorithm 1 Row Negations

Inputs: $\mathbf{S}$, PC_i {# of populated codes in input $\mathbf{S}$}, $\mathbf{A}_{syms}$
Outputs: $\mathbf{S}$, PC_o {# of populated codes in output $\mathbf{S}$}, $\mathbf{A}_{syms}$

$PC_o = PC_i \cdot 2^r$ {Operation limit = 2^r}
for $it = 1$ to $2^r - 1$ **do**
  $\mathbf{v}_{row} = de2bi(it, r)$ {Row negation vector}
  In $\mathbf{v}_{row}$, replace 1's with -1's and 0's with 1's
  $\mathbf{T}_{row} = diag(\mathbf{v}_{row})$
  for $m = 1$ to PC_i **do**
    $\mathbf{A}_{syms}(PC_i \cdot it + m,:) = \mathbf{A}_{syms}(m,:)$
    $\mathbf{S}(:,:,PC_i \cdot it + m) = \mathbf{T}_{row} \times \mathbf{S}(:,:,m)$
  end for
end for
return $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

Algorithm 2 Column Negations

Inputs: $\mathbf{S}$, PC_i, $\mathbf{A}_{syms}$
Outputs: $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

$PC_o = PC_i \cdot 2^{c-1}$ {Operation limit = 2^{c-1}}
for $it = 1$ to $2^{c-1} - 1$ **do**
  $\mathbf{v}_{col} = de2bi(it, c)$ {Column negation vector}
  In $\mathbf{v}_{col}$, replace 1's with -1's and 0's with 1's
  $\mathbf{T}_{col} = diag(\mathbf{v}_{col})$
  for $m = 1$ to PC_i **do**
    $\mathbf{A}_{syms}(PC_i \cdot it + m,:) = \mathbf{A}_{syms}(m,:)$
    $\mathbf{S}(:,:,PC_i \cdot it + m) = \mathbf{S}(:,:,m) \times \mathbf{T}_{col}$
  end for
end for
return $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

Algorithm 3 Row Permutations

Inputs: $\mathbf{S}$, PC_i, $\mathbf{A}_{syms}$
Outputs: $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

$PC_o = PC_i \cdot r!$ {Operation limit = $r!$}
$\mathbf{R}_{perms} = perms(r)$ {Row permutations array}
for $it = 1$ to $r! - 1$ **do**
  for $m = 1$ to PC_i **do**
    $\mathbf{A}_{syms}(PC_i \cdot it + m,:) = \mathbf{A}_{syms}(m,:)$
    $\mathbf{S}(:,:,PC_i \cdot it + m) = \mathbf{S}(\mathbf{R}_{perms}(it+1,:),:,m)$
  end for
end for
return $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

Algorithm 4 Column Permutations

Inputs: $\mathbf{S}$, PC_i, $\mathbf{A}_{syms}$
Outputs: $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

$PC_o = PC_i \cdot c!$ {Operation limit = $c!$}
$\mathbf{C}_{perms} = perms(c)$ {Column permutations array}
for $it = 1$ to $c! - 1$ **do**
  for $m = 1$ to PC_i **do**
    $\mathbf{A}_{syms}(PC_i \cdot it + m,:) = \mathbf{A}_{syms}(m,:)$
    $\mathbf{S}(:,:,PC_i \cdot it + m) = \mathbf{S}(:,\mathbf{C}_{perms}(it+1,:),m)$
  end for
end for
return $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

complexity were compared between the proposed scheme and two related techniques from the literature showing that far greater security is available by employing the proposed scheme.

APPENDIX A
UPDATED SET BUILDING ALGORITHMS

Originally in [17], Algorithms 0 through 5 have been updated for set building with the two new algorithms presented in Section III and are provided here for completeness. All functions and notation described in Section III-C applies to these algorithms as well. The $diag(\mathbf{v})$ MATLAB function is used to create a square diagonal matrix with the elements of vector $\mathbf{v}$ on the main diagonal [25]. In Algorithm 5, it should be understood that multiplying a STBC element by a conjugate operator, denoted as $(\cdot)^*$, in a transformation matrix, $\mathbf{T}$, results in the conjugate of that STBC element.

APPENDIX B
PROOF OF MRC SEQUENCE

For the purpose of brevity and without loss of generality, we use the Alamouti STBC to prove the two-step MRC sequence given in Section V-D. Starting with a variant of the Alamouti STBC represented as

$$\mathbf{G} = \begin{bmatrix} s_1 & -s_2^* \\ -s_2 & -s_1^* \end{bmatrix} \tag{53}$$

and channel tap vector $\mathbf{h}_{BU} = [h_1 \; h_2]^T$, Algorithm 8 is performed to construct

$$\mathbf{H}_{CBU} = \begin{bmatrix} h_1 & 0 & 0 & -h_2 \\ 0 & -h_1 & -h_2 & 0 \end{bmatrix}. \tag{54}$$

Algorithm 5 Symbol Conjugations

Inputs: $\mathbf{S}$, PC_i, $\mathbf{A}_{syms}$
Outputs: $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

$PC_o = PC_i \cdot 2^k$ {Operation limit = 2^k}
for $it = 1$ to $2^k - 1$ **do**
  $\mathbf{v}_{sym} = de2bi(it, k)$ {Symbol conjugation vector}
  for $m = 1$ to PC_i **do**
    $\mathbf{A}_{syms}(PC_i \cdot it + m, :) = \mathbf{A}_{syms}(m, :)$
    $\mathbf{T}$ = 2-dimensional array of 1's of size r-by-c
    for $ri = 1$ to r **do**
      for $ci = 1$ to c **do**
        if $\mathbf{S}(ri, ci, m) \neq 0$ **then**
          j = symbol index of $\mathbf{S}(ri, ci, m)$
          {e.g., $j = 1$ for $\mathbf{S}(ri, ci, m) == s_1, s_1^*$, etc.}
          if $\mathbf{v}_{sym}(j) == 1$ **then**
            $\mathbf{T}(ri, ci) = (\cdot)^*$
          end if
        end if
      end for
    end for
    $\mathbf{S}(:, :, PC_i \cdot it + m) = \mathbf{S}(:, :, m) \circ \mathbf{T}$
  end for
end for
return $\mathbf{S}$, PC_o, $\mathbf{A}_{syms}$

To confirm the sufficient statistic, (24) is calculated to obtain

$$\begin{aligned}\mathbf{H}_{int} &= \mathbf{H}_{CBU}^{\dagger} \times \mathbf{H}_{CBU} \\ &= \begin{bmatrix} h_1^* & 0 \\ 0 & -h_1^* \\ 0 & -h_2^* \\ -h_2^* & 0 \end{bmatrix} \times \begin{bmatrix} h_1 & 0 & 0 & -h_2 \\ 0 & -h_1 & -h_2 & 0 \end{bmatrix} \\ &= \begin{bmatrix} h_1h_1^* & 0 & 0 & -h_1^*h_2 \\ 0 & h_1h_1^* & h_1^*h_2 & 0 \\ 0 & h_1h_2^* & h_2h_2^* & 0 \\ -h_1h_2^* & 0 & 0 & h_2h_2^* \end{bmatrix}. \end{aligned} \tag{55}$$

Equation (23) is then used iteratively for $x, y \in \{1, \ldots, k\}$ to confirm (22). With $k = 2$, $x = 1$, and $y = 1$,

$$\begin{aligned}\hat{\mathbf{H}}(1,1) &= \mathbf{H}_{int}(1,1) + \mathbf{H}_{int}^*(3,3) + \mathbf{H}_{int}(1,3) + \mathbf{H}_{int}^*(3,1) \\ &= h_1h_1^* + h_2h_2^* + 0 + 0 = \|\mathbf{h}_{BU}\|^2. \end{aligned} \tag{56}$$

With $k = 2$, $x = 2$, and $y = 1$,

$$\begin{aligned}\hat{\mathbf{H}}(2,1) &= \mathbf{H}_{int}(2,1) + \mathbf{H}_{int}^*(4,3) + \mathbf{H}_{int}(2,3) + \mathbf{H}_{int}^*(4,1) \\ &= 0 + 0 + h_1^*h_2 - h_1^*h_2 = 0. \end{aligned} \tag{57}$$

With $k = 2$, $x = 1$, and $y = 2$,

$$\begin{aligned}\hat{\mathbf{H}}(1,2) &= \mathbf{H}_{int}(1,2) + \mathbf{H}_{int}^*(3,4) + \mathbf{H}_{int}(1,4) + \mathbf{H}_{int}^*(3,2) \\ &= 0 + 0 - h_1^*h_2 + h_1^*h_2 = 0. \end{aligned} \tag{58}$$

With $k = 2$, $x = 2$, and $y = 2$,

$$\begin{aligned}\hat{\mathbf{H}}(2,2) &= \mathbf{H}_{int}(2,2) + \mathbf{H}_{int}^*(4,4) + \mathbf{H}_{int}(2,4) + \mathbf{H}_{int}^*(4,2) \\ &= h_1h_1^* + h_2h_2^* + 0 + 0 = \|\mathbf{h}_{BU}\|^2. \end{aligned} \tag{59}$$

Thus, (22) is validated and the sufficient statistic is met for MRC.

For the communications link given in Section V-D, the r-by-1 received sample vector in (18) becomes

$$\begin{aligned}\mathbf{Z} &= \mathbf{G} \times \mathbf{h}_{BU} + \mathbf{n} = \mathbf{H}_{CBU} \times \mathbf{s}_{ext} + \mathbf{n} \\ &= \begin{bmatrix} s_1 & -s_2^* \\ -s_2 & -s_1^* \end{bmatrix} \times \begin{bmatrix} h_1 \\ h_2 \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} \\ &= \begin{bmatrix} h_1 & 0 & 0 & -h_2 \\ 0 & -h_1 & -h_2 & 0 \end{bmatrix} \times \begin{bmatrix} s_1 \\ s_2 \\ s_1^* \\ s_2^* \end{bmatrix} + \begin{bmatrix} n_1 \\ n_2 \end{bmatrix} \\ &= \begin{bmatrix} s_1h_1 - s_2^*h_2 + n_1 \\ -s_2h_1 - s_1^*h_2 + n_2 \end{bmatrix}. \end{aligned} \tag{60}$$

With this received vector, the first step of the MRC sequence given in (19) is calculated as

$$\begin{aligned}\hat{\mathbf{s}}_{int} &= \mathbf{H}_{CBU}^{\dagger} \times \mathbf{Z} \\ &= \begin{bmatrix} h_1^* & 0 \\ 0 & -h_1^* \\ 0 & -h_2^* \\ -h_2^* & 0 \end{bmatrix} \times \begin{bmatrix} s_1h_1 - s_2^*h_2 + n_1 \\ -s_2h_1 - s_1^*h_2 + n_2 \end{bmatrix} \\ &= \begin{bmatrix} s_1h_1h_1^* - s_2^*h_1^*h_2 + n_1h_1^* \\ s_2h_1h_1^* + s_1^*h_1^*h_2 - n_2h_1^* \\ s_1^*h_2h_2^* + s_2h_1h_2^* - n_2h_2^* \\ s_2^*h_2h_2^* - s_1h_1h_2^* - n_1h_2^* \end{bmatrix}. \end{aligned} \tag{61}$$

The second step given in (20) is then used iteratively for $x \in \{1, \ldots, k\}$ to obtain the estimated data symbol vector, $\hat{\mathbf{s}}$. With $k = 2$ and $x = 1$,

$$\begin{aligned}\hat{\mathbf{s}}(1) &= \frac{\hat{\mathbf{s}}_{int}(1) + \hat{\mathbf{s}}_{int}^*(3)}{\|\mathbf{h}_{BU}\|^2} \\ &= \frac{s_1h_1h_1^* - s_2^*h_1^*h_2 + n_1h_1^* + s_1h_2h_2^* + s_2^*h_1^*h_2 - n_2^*h_2}{\|\mathbf{h}_{BU}\|^2} \\ &= s_1 + \frac{n_1h_1^* - n_2^*h_2}{\|\mathbf{h}_{BU}\|^2}. \end{aligned} \tag{62}$$

With $k = 2$ and $x = 2$,

$$\begin{aligned}\hat{\mathbf{s}}(2) &= \frac{\hat{\mathbf{s}}_{int}(2) + \hat{\mathbf{s}}_{int}^*(4)}{\|\mathbf{h}_{BU}\|^2} \\ &= \frac{s_2h_1h_1^* + s_1^*h_1^*h_2 - n_2h_1^* + s_2h_2h_2^* - s_1^*h_1^*h_2 - n_1^*h_2}{\|\mathbf{h}_{BU}\|^2} \\ &= s_2 - \frac{n_2h_1^* + n_1^*h_2}{\|\mathbf{h}_{BU}\|^2}. \end{aligned} \tag{63}$$

APPENDIX C
EXAMPLE TABULAR KR

The KR corresponding to the MPO468 base code given in (2) represented in matrix form is shown in (28) whereas the tabular form is provided here in Table 6.

APPENDIX D
PROOF OF EXPECTED BIT ERROR RATE

From [17], $\hat{\mathbf{s}}$ may be obtained by performing *symbol assignment* and *variation decision* operations to decode $\mathbf{s}_c$. Also

TABLE 6. Tabular form KR showing the relationships and locations of each element for all symbol pairs in the MPO468 base code.

SPI	Relationship	Location of Variant 1	Location of Variant 2
1.1	Conj.	(1, 1)	(2, 2)
1.2	Negative-Conj.	(2, 1)	(1, 2)
2.1	Conj.	(5, 1)	(6, 2)
2.2	Negative-Conj.	(6, 1)	(5, 2)
3.1	Conj.	(1, 1)	(3, 3)
3.2	Negative-Conj.	(3, 1)	(1, 3)
4.1	Conj.	(5, 1)	(7, 3)
4.2	Negative-Conj.	(7, 1)	(5, 3)
5.1	Conj.	(2, 1)	(7, 4)
5.2	Negative-Conj.	(7, 1)	(2, 4)
6.1	Negative-Conj.	(3, 1)	(6, 4)
6.2	Conj.	(6, 1)	(3, 4)
7.1	Conj.	(1, 2)	(4, 3)
7.2	Negative-Conj.	(4, 2)	(1, 3)
8.1	Conj.	(5, 2)	(8, 3)
8.2	Negative-Conj.	(8, 2)	(5, 3)
9.1	Conj.	(2, 2)	(8, 4)
9.2	Negative-Conj.	(8, 2)	(2, 4)
10.1	Negative-Conj.	(4, 2)	(6, 4)
10.2	Conj.	(6, 2)	(4, 4)
11.1	Conj.	(3, 3)	(8, 4)
11.2	Negative-Conj.	(8, 3)	(3, 4)
12.1	Negative-Conj.	(4, 3)	(7, 4)
12.2	Conj.	(7, 3)	(4, 4)

from [17], "when performing symbol assignment, the number of permutations that will assign l symbols correctly and the remaining $k-l$ symbols incorrectly is

$$\text{Perms}(l,k) = \begin{cases} 1 & k = l \quad (64a) \\ \displaystyle\sum_{i=2}^{k-l} \left(\frac{(-1)^i \cdot k!}{i! \cdot l!} \right) & k - l \geq 2" \quad (64b) \\ 0 & \text{otherwise.} \quad (64c) \end{cases}$$

Assuming $\mathbf{s}_c$ is in the correct message residue class, we use this information to calculate expected BER as

$$\text{BER}_{\text{exp}} = \frac{E[l|k] \cdot E[B_{ca}] + (k - E[l|k]) \cdot E[B_{ia}]}{k \cdot b}, \quad (65)$$

where $E[\,]$ is the expectation operator, l is the number of correctly assigned symbols per codeword, k is the number of symbols per codeword, B_{ca} is the number of bit errors per correctly assigned symbol, B_{ia} is the number of bit errors per incorrectly assigned symbol, $b = \log_2(M)$ is the number of bits per symbol, and M is the order of the digital modulation scheme employed. $E[B_{ca}] = 1$ based upon the average of the four possible variation decisions as discussed in [17]. $E[B_{ia}] = \frac{b}{2}$ assuming symbols within the codeword are independent of one another. The expected value of l given k, can be calculated as

$$E[l|k] = \sum_{l=0}^{k} l \cdot P(l|k), \quad (66)$$

where

$$P(l|k) = \frac{\text{Perms}(l,k)}{k!} \quad (67)$$

is the probability of l given k, assuming each permutation is equally probable. Substituting (67) into (66) and removing the $l = 0$ and $l = k-1$ cases as they always evaluate to 0 results in

$$E[l|k] = \begin{cases} 1 & k \leq 2 \quad (68a) \\ \dfrac{k}{k!} + \displaystyle\sum_{l=1}^{k-2} l \cdot \sum_{i=2}^{k-l} \left(\frac{(-1)^i}{i! \cdot l!} \right) & k > 2. \quad (68b) \end{cases}$$

It can be shown that evaluating (68) for any value of k yields $E[l|k] = 1$. Thus, substituting this result along with $E[B_{ca}]$ and $E[B_{ia}]$ into (65) provides

$$\text{BER}_{\text{exp}} = \frac{1 \cdot 1 + (k-1) \cdot \frac{b}{2}}{k \cdot b}. \quad (69)$$

X. ACKNOWLEDGMENT

The authors would like to acknowledge the support of Mr. Adolfo Venegas, NIWC Pacific.

REFERENCES

[1] B. Chen, C. Zhu, W. Li, J. Wei, V. C. M. Leung, and L. T. Yang, "Original symbol phase rotated secure transmission against powerful massive MIMO eavesdropper," *IEEE Access*, vol. 4, pp. 3016–3025, 2016.

[2] Y. Qassim, M. E. Magaña, and A. Yavuz, "Post-quantum hybrid security mechanism for MIMO systems," in *Proc. Int. Conf. Comput. Netw. Commun.*, Santa Clara, CA, USA, Jan. 2017, pp. 684–689.

[3] L. Sun and Q. Du, "Physical layer security with its applications in 5G networks: A review," *China Commun.*, vol. 14, no. 12, pp. 1–14, Dec. 2017.

[4] J. Hua, S. Jiang, W. Lu, Z. Xu, and F. Li, "A novel physical layer encryption algorithm based on statistical characteristics of time-selective channels," *IEEE Access*, vol. 6, pp. 38225–38233, 2018.

[5] H. Noura, M. Mansour, and A. Chehab, "Physical layer security schemes for MIMO systems: An overview," *Wireless Netw.*, vol. 26, no. 3, pp. 2089–2111, 2020. [Online]. Available: http://search.proquest.com/docview/2239962925/

[6] A. D. Wyner, "The wire-tap channel," *Bell Syst. Techn. J.*, vol. 54, no. 8, pp. 1355–1387, 1975.

[7] I. Csiszar and J. Korner, "Broadcast channels with confidential messages," *IEEE Trans. Inf. Theory*, vol. 24, no. 3, pp. 339–348, May 1978.

[8] A. Thangaraj, S. Dihidar, A. R. Calderbank, S. W. McLaughlin, and J. Merolla, "Applications of LDPC codes to the wiretap channel," *IEEE Trans. Inf. Theory*, vol. 53, no. 8, pp. 2933–2945, Aug. 2007.

[9] X. Zhou, L. Song, and Y. Zhang, *Physical Layer Security in Wireless Communications* (Wireless networks and mobile communications 20). Boca Raton, FL, USA: Taylor Francis, 2014.

[10] A. Subramanian, A. Thangaraj, M. Bloch, and S. W. McLaughlin, "Strong secrecy on the binary erasure wiretap channel using large-girth LDPC codes," *IEEE Trans. Inf. Forensics Security*, vol. 6, pp. 585–594, 2011.

[11] S. Goel and R. Negi, "Guaranteeing secrecy using artificial noise," *IEEE Trans. Wireless Commun.*, vol. 7, no. 6, pp. 2180–2189, Jun. 2008.

[12] S. A. A. Fakoorian, H. Jafarkhani, and A. L. Swindlehurst, "Secure space-time block coding via artificial noise alignment," in *Proc. Conf. Rec. 45th Asilomar Conf. Signals Syst. Comput. (ASILOMAR)*, Pacific Grove, CA, USA, 2011, pp. 651–655.

[13] J. M. Hamamreh, E. Guvenkaya, T. Baykas, and H. Arslan, "A practical physical-layer security method for precoded OSTBC-based systems," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Doha, Qatar, 2016, pp. 1–6.

[14] T. Allen, A. Tajer, and N. Al-Dhahir, "Secure alamouti multiple access channel transmissions: Multiuser transmission and multi-antenna eavesdroppers," *IEEE Wireless Commun. Lett.*, vol. 8, no. 5, pp. 1510–1513, Oct. 2019.
[15] T. Allen, J. Cheng, and N. Al-Dhahir, "Secure space-time block coding without transmitter CSI," *IEEE Wireless Commun. Lett.*, vol. 3, no. 6, pp. 573–576, Dec. 2014.
[16] N. Sanandaji and A. Falahati, "A hidden OSTBC scheme to enhance physical layer security by employing a pseudorandom precoder," *IEEE Trans. Commun.*, vol. 68, no. 1, pp. 363–375, Jan. 2020.
[17] M. Cribbs, R. Romero, and T. Ha, "Orthogonal STBC set building and physical layer security application," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Atlanta, GA, USA, 2020, pp. 1–5.
[18] P. Ramabadran *et al.*, "A novel physical layer encryption scheme to counter eavesdroppers in wireless communications," in *Proc. 25th IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, Bordeaux, France, 2018, pp. 69–72.
[19] D. Chen, N. Zhang, N. Cheng, K. Zhang, Z. Qin, and X. Shen, "Physical layer based message authentication with secure channel codes," *IEEE Trans. Depend. Secure Comput.*, vol. 17, no. 5, pp. 1079–1093, Sep./Oct. 2020.
[20] O. Souihli and T. Ohtsuki, "The two-way MIMO wire-tap channel," in *Proc. IEEE Int. Conf. Commun.*, Dresden, Germany, 2009, pp. 1–6.
[21] C. Sahin, B. Katz, and K. R. Dandekar, "Secure and robust symmetric key generation using physical layer techniques under various wireless environments," in *Proc. IEEE Radio Wireless Symp. (RWS)*, Austin, TX, USA, 2016, pp. 211–214.
[22] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 8, pp. 1451–1458, Oct. 1998.
[23] S. N. Diggavi, N. Al-Dhahir, A. Stamoulis, and A. R. Calderbank, "Great expectations: The value of spatial diversity in wireless networks," *Proc. IEEE*, vol. 92, no. 2, pp. 219–270, Feb. 2004.
[24] X.-B. Liang, "Orthogonal designs with maximal rates," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2468–2503, Oct. 2003.
[25] *MATLAB Documentation*, MathWorks, Inc., Natick, MA, USA. Accessed: Feb. 10, 2020. [Online]. Available: https://www.mathworks.com/help/index.html
[26] D. J. Love, R. W. Heath, and T. Strohmer, "Grassmannian beamforming for multiple-input multiple-output wireless systems," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2735–2747, Oct. 2003.
[27] J. Zhu, X. She, and L. Chen, "Efficient CQI update scheme for codebook based MU-MIMO with single CQI feedback in E-UTRA," in *Proc. IEEE 19th Int. Symp. Pers. Indoor Mobile Radio Commun.*, Cannes, France, 2008, pp. 1–6.
[28] T. T. Ha, *Theory and Design of Digital Communication Systems*. New York, NY, USA: Cambridge Univ. Press, 2011.
[29] V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space-time block codes from orthogonal designs," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1456–1467, Jul. 1999.
[30] A. V. Geramita and J. Seberry, *Orthogonal Designs, Quadratic Forms and Hadamard Matrices, Lecture Notes in Pure and Applied Mathematics*. vol. 43. New York, NY, USA: Marcel Dekker, 1979.
[31] C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Techn. J.*, vol. 28, no. 4, pp. 656–715, Oct. 1949.
[32] J. van Tilburg and D. E. Boekee, "Divergence bounds on key equivocation and error probability in cryptanalysis," in *Advances in Cryptology (CRYPTO)*, H. Williams, Ed. Berlin, Germany: Springer-Verlag, 1986, pp. 489–513.
[33] *IEEE Standard for Ethernet*, IEEE Standard 802.3-2015, pp. 1–4017, 2016.
[34] M. J. Wiener, "Exhaustive key search," in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg, Ed. Boston, MA, USA: Springer, 2005, pp. 206–209. [Online]. Available: https://doi.org/10.1007/0-387-23483-7_147

MICHAEL R. CRIBBS (Graduate Student Member, IEEE) received the B.S.E.E. degree from the University of Kansas, Lawrence, KS, USA, in 2009, and the M.S.E.E. degree from the Naval Postgraduate School, Monterey, CA, USA, in 2015, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering.

He enlisted in the United States Navy in 2004 and received his commission in the United States Navy in 2009. He continues to serve on active duty as an Engineering Duty Officer with the Department of Electrical and Computer Engineering, Naval Postgraduate School. His research interests include communications and coding theory.

RIC A. ROMERO (Senior Member, IEEE) received the B.S.E.E. degree from Purdue University, West Lafayette, IN, USA, in 1999, the M.S.E.E. degree from the University of Southern California, Los Angeles, CA, USA, in 2004, and the Ph.D. degree in electrical and computer engineering from the University of Arizona, Tucson, AZ, USA, in 2010.

He was a Senior Multidisciplined Engineer II with Raytheon Missile Systems, Tucson, from 1999 to 2010. He was involved in various communications, radar, and research and development programs. He was also a Graduate Research Assistant with the Laboratory for Sensor and Array Processing, University of Arizona, from 2007 to 2010. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA, USA. His research interests include the general areas of radar, sensor information processing, and communications.

Dr. Romero was awarded the 2004 Corporate Excellence in Technology Award, which is a company-wide technical prize at Raytheon Corporation. He was also granted the Raytheon Advanced Scholarship Program Fellowships, from 2002 to 2004 and from 2005 to 2007.

TRI T. HA (Life Fellow, IEEE) has been a Professor with the Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA, USA, since 1989. Prior to joining NPS, he was with Fairchild Industries and General Telephone and Electronics Corporation and an Associate Professor with the Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, for four years. He has authored three textbooks. His current research interests are in wireless communications and cyber warfare.