4

Construction of Orthogonal Arrays of Index Unity Using Logarithm Tables for Galois Fields

Jose Torres-Jimenez[1], Himer Avila-George[2],
Nelson Rangel-Valdez[3] and Loreto Gonzalez-Hernandez[1]
[1]*CINVESTAV-Tamaulipas, Information Technology Laboratory*
[2]*Instituto de Instrumentación para Imagen Molecular (I3M). Centro mixto CSIC - Universitat Politécnica de Valéncia - CIEMAT, Valencia*
[3]*Universidad Politécnica de Victoria*
[1,3]*México*
[2]*Spain*

1. Introduction

A wide variety of problems found in computer science deals with combinatorial objects. Combinatorics is the branch of mathematics that deals with finite countable objects called combinatorial structures. These structures find many applications in different areas such as hardware and software testing, cryptography, pattern recognition, computer vision, among others.

Of particular interest in this chapter are the combinatorial objects called Orthogonal Arrays (OAs). These objects have been studied given of their wide range of applications in the industry, Gopalakrishnan & Stinson (2008) present their applications in computer science; among them are in the generation of error correcting codes presented by (Hedayat et al., 1999; Stinson, 2004), or in the design of experiments for software testing as shown by Taguchi (1994).

To motivate the study of the OAs, it is pointed out their importance in the development of algorithms for the cryptography area. There, OAs have been used for the generation of authentication codes, error correcting codes, and in the construction of universal hash functions (Gopalakrishnan & Stinson, 2008).

This chapter proposes an efficient implementation for the Bush's construction (Bush, 1952) of OAs of index unity, based on the use of logarithm tables for Galois Fields. This is an application of the algorithm of Torres-Jimenez et al. (2011). The motivation of this research work born from the applications of OAs in cryptography as shown by Hedayat et al. (1999). Also, it is discussed an alternative use of the logarithm table algorithm for the construction of cyclotomic matrices to construct CAs (Colbourn, 2010).

The remaining of the chapter is organized as follows. Section 2 presents a formal definition of OAs and the basic notation to be used through this chapter. Section 3 shows the relevance of OAs for cryptography by showing three of their applications, one in the authentication without secrecy, other in the generation of universal hash functions, and a last one in the construction of difference schemes. Section 4 shows the construction methods, reported in

the literature, for the construction of OAs. Section 5 presents the algorithm described in Torres-Jimenez et al. (2011) for the construction of the logarithm table of a Galois Field, this algorithm served as basis for a more efficient construction of OAs using the Bush's construction. Section 6 contains the efficient implementation, proposed in this chapter, for the Bush's construction of OAs, based on discrete logarithms. Section 7 presents an extension of the use of the algorithm presented by Torres-Jimenez et al. (2011), in the construction of cyclotomic matrices for CAs. Section 8 shows as results from the proposed approach, a set of bounds obtained for CAs using the constructions of cyclotomic matrices aided by the algorithm described in this chapter. Finally, Section 9 presents the main conclusions derived from the research proposed in this chapter.

2. Orthogonal arrays

The Orthogonal Arrays (OAs) were introduced by Rao (1946; 1947) under the name of *hypercubes* and for use in factorial designs. Figure 1 shows an example of an Orthogonal Array $OA_3(12;2,11,2)$. The definition of an OA involves that any pair of columns of this

$$\begin{pmatrix}
0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&1&1&0&1&0&0&0&0\\
0&1&1&1&0&1&1&0&1&0&0&0\\
0&0&1&1&1&0&1&1&0&1&0&0\\
0&0&0&1&1&1&0&1&1&0&1&0\\
1&0&0&0&1&1&1&0&1&1&0&0\\
0&1&0&0&0&1&1&1&0&1&1&0\\
1&0&1&0&0&0&1&1&1&0&1&0\\
1&1&0&1&0&0&0&1&1&1&0&0\\
0&1&1&0&1&0&0&0&1&1&1&0\\
1&0&1&1&0&1&0&0&0&1&1&0\\
1&1&0&1&1&0&1&0&0&0&1&0
\end{pmatrix}$$

Fig. 1. Example of an $OA_3(12;2,11,2)$. The interaction, or strength, is 2; also, it has 11 parameters and 12 runs (or test cases) and the combinations $\{(0,0),(0,1),(1,0),(1,1)\}$ in each pair of columns extracted from it.

matrix should contain the symbol combinations shown in Figure 2.

$$\begin{pmatrix}
0&0\\
0&1\\
1&0\\
1&1
\end{pmatrix}$$

Fig. 2. Symbol combinations expected in any pair of columns in an OA of strength 2 and alphabet 2.

Formally, an orthogonal array (OA), denoted by $OA_\lambda(N;t,k,v)$, can be defined as follows:

Definition 1. *An OA, denoted by $OA(N;t,k,v)$, is an $N \times k$ array on v symbols such that every $N \times t$ sub-array contains all the ordered subsets of size t from v symbols exactly λ times. Orthogonal arrays have the property that $\lambda = \frac{N}{v^t}$. When $\lambda = 1$ it can be omitted from the notation and the OA is optimal.*

Figure 3 shows another example of an $OA(9;2,4,3)$; note that this time the alphabet is $v = 3$ and the combination of symbols $\{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\}$ appears only once in each pair of columns of the OA.

$$\begin{pmatrix} 0&0&0&0 \\ 1&1&1&0 \\ 2&2&2&0 \\ 0&1&2&1 \\ 1&2&0&1 \\ 2&0&1&1 \\ 0&2&1&2 \\ 1&0&2&2 \\ 2&1&0&2 \end{pmatrix}$$

Fig. 3. Example of an $OA(9;2,4,3)$.

The OAs have some interesting properties, among them are the following ones:

1. The parameters of the OA satisfy $\lambda = N/v^t$;
2. An OA of strength t is also an OA of strength t', where $1 \leq t' \leq t$. The index λ' of an OA of strength t' is $\lambda' = \lambda \cdot v^{t-t'}$;
3. Let $A_i = \{0,1,...,r\}$ be a set of $OA(N_i;t_i,k,v)$, the juxtaposed array $A = \begin{bmatrix} A_0 \\ ... \\ A_r \end{bmatrix}$ is an $OA(N;t,k,v)$ where $N = N_1 + N_2 + ... + N_r$ and $t \geq \min\{t_0,t_1,...,t_r\}$;
4. Any permutation of rows or columns in an OA, results in another OA with the same parameters;
5. Any subarray of size $N \times k'$ of an $OA(N;t,k,v)$, is an $OA(N;t',k',v)$ of strength $t' = \min\{k',t\}$;
6. Select the rows of an $OA(N;t,k,v)$ that starts with the symbol 0, and eliminate the first column; the resulting matrix is an $OA(N/v;t-1,k-1,v)$.

The following section presents some applications of OAs in the area of cryptography. These applications are related with the construction of difference schemes, universal hash functions, and in the authentication without secrecy.

3. Relevance of orthogonal arrays in cryptography

The purpose of this section is to present three applications that motivate the study of OAs in the area of cryptography. These applications have been described in (Gopalakrishnan & Stinson, 2008; Stinson, 1992a).

3.1 Authentication without secrecy

The use of authentication codes dates back to 1974, the time when they were invented by Gilbert et al. (1974). Most of the time, the transmission of information between two parts that are interested on keeping the integration of their information, is done through the use of *secrecy*, i.e. the practice of hiding information from certain group of individuals. However, sometimes it is important to transmit the information in areas that are insecure and where it is

not necessary the secrecy. This part corresponds to the area of *Authentication Without Secrecy* (or AWS). An authentication code without secrecy is a code where an observed message can correspond to a unique source state.

Jones & Seberry (1986) described a situation in which two countries want to set transmission devices to monitor the activities of the other, such that possible compliance can be avoided.

The general model to define the use of the AWS can be described with three participants: a transmitter, a receiver, and an opponent. Let's call these participants Alice, Bob and Gabriel, respectively. Suppose that Alice wants to transmit a message to Bob in a public communication channel; however, they expect that the message must be transmitted integrally, i.e. without any changes in its composition. To do so, Alice encrypted the message and sent it through the channel. An encoding rule (based on a key scheme) ciphers the message; each encoding rule will be a one-to-one function from the source space to the message space. The key used to cipher the message has been sent to Bob (the receiver) through a secure channel, before the message has been encoded. Now, the third party member, Gabriel, has malicious intention of deforming the message. What is the chance of Gabriel to access the message of Alice and Bob and modify it conveniently to affect the final result?

Let's consider the following protocol of communication between Alice and Bob: a) Firstly, Alice and Bob choose the encoding code previously; b) Alice encode the message with a previously chosen key K; c) the message $m = (s, a)$ is sent over the communication channel; d) when Bob receives the message he verifies that $a = e_K(s)$ so that he ensures that it comes from Alice.

Let S be a set of k source states; let $\mathcal{M}$ be a set of v messages; and let $\mathcal{E}$ be a set of b encoding rules. Since each encoding rule is a one-to-one function from S to $\mathcal{M}$, the code can be represented by a $b \times k$ matrix, where the rows are indexed by encoding rules, the columns are indexed by source states, and the entry in row e and column s is $e(s)$. This matrix is called the encoding matrix. For any encoding rule $e \in \mathcal{E}$, define $M(e) = \{e(s) : s \in S\}$, i.e. the set of valid messages under encoding rule e. For an encoding rule e, and a message $m \in M(e)$, define $e^{-1}(m) = s$ if $e(s) = m$.

The types of damage that Gabriel can do to the message of Alice and Bob are impersonation, i.e. sending a message to one of them without the message even existed; and substitution, i.e. changing a message sent.

The application of OAs in *authentication without secrecy* is described by the following theorem:

Theorem 1. *Suppose that there is an authentication code without secrecy for k source states and having l authenticators, in which $P_{d_0} = P_{d_1} = 1/l$. Then*

1. *$|\mathcal{E}| \geq l^2$, and equality occurs if and only if the authentication matrix is an $OA(2, k, l)$ (with $\lambda = 1$) and the authentication rules are used with equal probability;*
2. *$|\mathcal{E}| \geq k(l-1)+1$, and equality occurs if and only if the authentication matrix is an $OA_\lambda(2, k, l)$ where*
$$\lambda = \frac{k(l-1)+1}{l^2}, \tag{1}$$
and the authentication rules are used with equal probability.

This theorem has been proven by Stinson (1992a). It also show that this is the minimum probability expected for this case.

3.2 Universal hash function

Assume it is wanted to map keys from some universe U into m bins (labeled). The algorithm will have to handle some data set of $|S| = n$ keys, which is not known in advance. Usually, the goal of hashing is to obtain a low number of collisions (keys from S that land in the same bin). A deterministic hash function cannot offer any guarantee in an adversarial setting if the size of U is greater than m^2, since the adversary may choose S to be precisely the preimage of a bin. This means that all data keys land in the same bin, making hashing useless. Furthermore, a deterministic hash function does not allow for rehashing: sometimes the input data turns out to be bad for the hash function (e.g. there are too many collisions), so one would like to change the hash function.

The solution to these problems is to pick a function randomly from a family of hash functions. A universal hash function is a family of functions indexed by a parameter called the key with the following property: for all distinct inputs, the probability over all keys that they collide is small.

A family of functions $H = \{h : U \to [m]\}$ is called a universal family if Equation 2 holds.

$$\forall x, y \in U, x \neq y : Pr[h(x) = h(y)] \leq \frac{1}{m} \tag{2}$$

Any two keys of the universe collide with probability at most $\frac{1}{m}$ when the hash function h is drawn randomly from H. This is exactly the probability of collision we would expect if the hash function assigned truly random hash codes to every key. Sometimes, the definition is relaxed to allow collision probability $O(1/m)$. This concept was introduced by (Carter & Wegman, 1979; Wegman & Carter, 1981), and has found numerous applications in computer science.

A finite set H of hash functions is $strongly - universal_2$ (or SU_2) if Equation 3 holds.

$$\{h \in H : h(x_1) = y_1, h(x_2) = y_2\}| = |H|/|B|^2, \forall x_1, x_2 \in A(x_1 \neq x_2), y_1, y_2 \in B \tag{3}$$

For practical applications, it is also important that $|H|$ is small. This is because $log_2|H|$ bits are needed to specify a hash function from the family. It is fairly straightforward to show that strongly universal hash functions are equivalent to orthogonal arrays. The following theorem can be found in (Stinson, 1994).

Theorem 2. *If there exists an $OA_\lambda(2, k, n)$, then there exists an SU_2 class H of hash functions from A to B, where $|A| = k, |B| = n$ and $|H| = \lambda n^2$. Conversely, if there exists an SU_2 class H of hash functions from A to B, where $a = |A|$ and $b = |B|$, then there exists an $OA_\lambda(2, k, n)$, where $n = b$,$k = a$ and $\lambda = |H|/n^2$.*

This theorem helps in establishing lower bounds on the number of hash functions and in constructing classes of hash functions which meet these bounds. It is straightforward to extend the definition and the theorem to SU_t class of universal hash functions.

3.3 Thresholds schemes

In a bank, there is a vault which must be opened every day. The bank employs three senior tellers; but it is not desirable to entrust the combination to any one person. Hence, we want

to design a system whereby any two of the three senior tellers can gain access to the vault but no individual can do so. This problem can be solved by means of a threshold scheme.

Threshold schemes are actually a special case of secret sharing schemes. Stinson (1992b) presents a survey in this topic. Informally a (t, w)-threshold scheme is a method of sharing a secret key K among a finite set $\mathcal{P}$ of w participants, in such a way that any t participants can compute the value of K, but no group of $t-1$ (or fewer) participants can do so. The value of K is chosen by a special participant called the *dealer*. The dealer is denoted by D and we assume $D \notin \mathcal{P}$. When D wants to share the key K among the participants in $\mathcal{P}$, he gives each participant some partial information called a *share*. The shares should be distributed secretly, so no participant knows the share given to another participant.

At a later time, a subset of participants $B \subseteq \mathcal{P}$ will pool their shares in an attempt to compute the secret key K. If $|B| \geq t$, then they should be able to compute the value of K as a function of the shares they collectively hold; if $|B| < t$, then they should not be able to compute K. In the example described above, we desire a $(2, 3)$-threshold scheme.

Often, we desire not only that an unauthorized subset of participants should be unable to compute the value of K by pooling their shares, but also they should be unable to determine anything about the value of K. Such a threshold scheme is called a *perfect threshold scheme*. Here, we will be concerned only about perfect threshold schemes.

We will use the following notation. Let $\mathcal{P} = \{P_i : 1 \leq i \leq w\}$ be the set of participants. $\mathcal{K}$ is the *key set* (i.e., the set of all possible keys); and S is the *share threshold schemes*.

Orthogonal arrays come into picture once again by means of the following theorem due to Dawson & Mahmoodian (1993).

Theorem 3. *An ideal (t, w) threshold scheme with $|\mathcal{K}| = v$ exists if and only if an $OA(t, w+1, v)$ exists.*

The construction of the threshold scheme starting from the orthogonal array proceeds as follows. The first column of the OA corresponds to the dealer and the remaining w columns correspond to the w participants. To distribute a specific key K, the dealer selects a random row of the OA such that K appears in the first column and gives out the remaining w elements of the row as the shares. When t participants later pool their shares, the collective information will determine a unique row of the OA (as $\lambda = 1$) and hence they can compute K as the value of the first element in the row.

Can a group of $t-1$ participants compute K? Any possible value of the secret along with the actual shares of these $t-1$ participants determine a unique row of the OA. Hence, no value of the secret can be ruled out. Moreover, it is clear that the $t-1$ participants can obtain no information about the secret.

4. Algorithms to construct OAs

This section presents some of the state-of-art algorithms for the construction of OAs. Special reference is done to the Bush's construction, which is benefited from the approach presented in this chapter because the efficient way of constructing the OAs using logarithm tables.

4.1 Rao-Hamming construction

The Rao-Hamming construction derived from the geniality of two scientists who independently elaborate procedures for the construction of OAs Hedayat et al. (1999). The following theorem describes the purpose of this construction.

Theorem 4. *If there is a prime power then an* $OA(s^n, (s^n-1)/(s-1), 2)$ *exists whenever* $n \geq 2$.

A simple way to obtain an orthogonal array with these parameters is the following. This construction always produces linear arrays. Form an $s^n \times n$ array whose rows are all possible n-tuples from $GF(s)$. Let $C_1, ..., C_n$ denote the columns of this array. The columns of the full orthogonal array then consist of all columns of the form shown in Equation 4.

$$z_1C_1 + ... + z_nC_n = [C_1, ..., C_n]z \tag{4}$$

where $z = (z_1, ..., z_n)^T$ is an n-tuple from $GF(s)$, not all the z_i are zero, and the first nonzero z_i is 1. There are $(s^n-1)/(s-1)$ such columns, as required.

An alternative way to construct an OA using the Rao-Hamming Construction is by forming an $n \times (s^n-1)/(s-1)$ matrix whose columns are all nonzero n-tuples $(z_1, ..., z_n)^T$ from $GF(s)$ in which the first nonzero z_i is 1. The OA is then formed by taking all the linear combinations of the rows of this generator matrix.

An example of the construction of an OA, taken from Hedayat et al. (1999), is shown in Figure 4.

(a)

$$\begin{pmatrix} 1&0&0&1&1&0&1 \\ 0&1&0&1&0&1&1 \\ 0&0&1&0&1&1&1 \end{pmatrix}$$

(b)

$$\begin{pmatrix} 0&0&0&0&0&0&0 \\ 1&0&0&1&1&0&1 \\ 0&1&0&1&0&1&1 \\ 0&0&1&0&1&1&1 \\ 1&1&0&0&1&1&0 \\ 1&0&1&1&0&1&0 \\ 0&1&1&1&1&0&0 \\ 1&1&1&0&0&0&1 \end{pmatrix}$$

Fig. 4. Example of the construction of an $OA(8; 2, 7, 2)$ using the Rao-Hamming construction. Figure 4(a) contains the generator matrix. Figure 4(b) shows the OA constructed from it.

4.2 Difference scheme algorithm

Difference schemes (DS), denoted by $D(r, c, s)$ are tables of r rows and c columns with s symbols such that the difference between each pair of columns yields all the symbols $\{0, 1, 2, ..., s-1\}$.

If you have a difference scheme, you easily generate an orthogonal array by simply replicating the difference scheme s times and adding to each replication all symbols in turn modulo (s): if the sum exceeds s, you divide by s and keep the remainder.

So the problem becomes finding difference schemes. For instance, the multiplicative group of a Galois field is a difference scheme.

$$\begin{pmatrix} 0&0&0&0&0&0 \\ 0&1&2&1&2&0 \\ 0&2&1&1&0&2 \\ 0&2&2&0&1&1 \\ 0&0&1&2&2&1 \\ 0&1&0&2&1&2 \end{pmatrix}$$

Fig. 5. Example of a difference scheme $D(6,6,3)$.

An example is shown in Figure 5, as the multiplication table of $GF(2^2)$.

Given that the DS $D(r,c,s)$ is an array of size $r \times c$ based on the s elements of a group G so that for any two columns the element-wise differences contain every element of G equally often; clearly $r = \lambda s$ for some λ called the index.

If $D = D(r,c,s)$, then $\begin{bmatrix} D+0 \\ D+1 \\ ... \\ D+(s-1) \end{bmatrix}$ is an $OA(rs;2,c,s)$. Figure 6 shows the construction of the $OA(16;2,4,4)$ from a $D(4,4,4)$.

(a)

$$\begin{pmatrix} 0&0&0&0 \\ 0&1&2&3 \\ 0&2&3&1 \\ 0&3&1&2 \end{pmatrix}$$

(b)

$$\begin{pmatrix} 0&0&0&0 \\ 0&1&2&3 \\ 0&2&3&1 \\ 0&3&1&2 \\ 1&1&1&1 \\ 1&2&3&0 \\ 1&3&0&2 \\ 1&0&2&3 \\ 2&2&2&2 \\ 2&3&0&1 \\ 2&0&1&3 \\ 2&1&3&0 \\ 3&3&3&3 \\ 3&0&1&2 \\ 3&1&2&0 \\ 3&2&0&1 \end{pmatrix}$$

Fig. 6. Generated orthogonal array $OA(16;2,4,4)$ using the $D(4,4,4)$. Figure 6(a) presents the different scheme $D = (4,4,4)$. Figure 6(b) the OA constructed.

4.3 Hadamard matrix algorithms

Hadamard matrix is a DS with only two symbols: $\{-1,+1\}$. The interest in Hadamard matrices lies in the Hadamard conjecture which states that all multiples of 4 have a corresponding Hadamard matrix. Hadamard matrices are square matrices with a fixed column of just 1's. The smallest one is shown in Figure 7(a).

(a) (b)

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

Fig. 7. Example of two Hadamard matrices H_2, H_4 of orders 2 and 4, respectively.

The Hadamard matrix H_4, that is shown in Figure 7(b), does not differ from the Rao-Hamming $OA(4;2,3,2)$.

Figure 8 shows another example of a Hadamard matrix. This time it is shown its corresponding OA resulting after the removal of the first column and a symbol recoding.

(a) (b)

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Fig. 8. Figure 8(a) shows a Hadamard matrix of order 8; Figure 8(b) presents its equivalent $OA_2(2,7,2)$.

Not all Hadamard matrices can be generated by the Rao Hamming algorithm just by the addition of a column of 1's. Rao Hamming works if the number of levels is a power of a prime number. And this happens in a Hadamard matrix, where the number of levels is 2 (prime number). But not all Rao Hamming arrays are square after the addition of a single column of 1's. Moreover, the number of rows in a Rao Hamming OA is a power of the number of levels.

Remember the general form $OA(sn;2,(sn-1)/(s-1),s)$, Hadamard matrices are square and the number of rows in the array need only to be a multiple of 4. For instance, 12 is a multiple of 4, it is not a prime power being the product 3. No Rao Hamming construction would yield a H_{12} matrix.

4.4 The Bush's construction

The Bush's construction is used to construct $OA(v^t;t,v+1,v)$, where $v = p^n$ is a prime power. This construction considers all the elements of the Galois Field $GF(v)$, and all the polynomials $y_j(x) = a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \ldots + a_1x + a_0$, where $a_i \in GF(v)$. The number of polynomials $y_j(x)$ are v^t, due to the fact that there are v different coefficients per each of the t terms.

Let's denote each element of $GF(v)$ as e_i, for $0 \leq i \leq v-1$. The construction of an OA following the Bush's construction is done as follow:

1. Generate a matrix $\mathcal{M}$ formed by v^t rows and $v+1$ columns;
2. Label the first v columns of $\mathcal{M}$ with an element $e_i \in GF(v)$;

3. Label each row of $\mathcal{M}$ with a polynomial $y_j(x)$;
4. For each cell $m_{j,i} \in \mathcal{M}$, $0 \leq j \leq v^t - 1, 0 \leq i \leq v - 1$, assign the value u whenever $y_j(e_i) = e_u$ (i.e. evaluates the polynomial $y_j(x)$ with $x = e_i$ and determines the result in the domain of $GF(v)$); and
5. Assign value u in cell $m_{j,i}$, for $0 \leq j \leq v^t - 1, i = v$, if e_u is the leading coefficient of $y_j(x)$, i.e. $e_u = a_{t-1}$ in the term $a_{t-1}x^{t-1}$ of the polynomial $y_j(x)$.

The constructed matrix $\mathcal{M}$ following the previous steps is an OA. We point out in this moment that the construction requires the evaluation of the polynomials $y_j(x)$ to construct the OA. The following subsection describes the general idea of the algorithm that does this construction with an efficient evaluation of these polynomials.

This section presented a survey of some construction reported in the scientific literature that are used to generate OAs. The following section will present an algorithm for the generation of logarithm tables of finite fields.

5. Algorithm for the construction of logarithm tables of Galois fields

In Barker (1986) a more efficient method to multiply two polynomials in $GF(p^n)$ is presented. The method is based on the definition of logarithms and antilogarithms in $GF(p^n)$. According with Niederreiter (1990), given a primitive element ρ of a finite field $GF(p^n)$, the discrete logarithm of a nonzero element $u \in GF(p^n)$ is that integer k, $1 \leq k \leq p^n - 1$, for which $u = \rho^k$. The antilogarithm for an integer k given a primitive element ρ in $GF(p^n)$ is the element $u \in GF(p^n)$ such that $u = \rho^k$. Table 1 shows the table of logarithms and antilogarithms for the elements $u \in GF(3^2)$ using the primitive element $x^2 = 2x + 1$; column 1 shows the elements in $GF(3^2)$ (the antilogarithm) and column 2 the logarithm.

Using the definition of logarithms and antilogarithms in $GF(p^n)$, the multiplication between two polynomials $\mathcal{P}_1(x)\mathcal{P}_2(x) \in GF(p^n)$ can be done using their logarithms $l_1 = log(\mathcal{P}_1(x)), l_2 = log(\mathcal{P}_2(x))$. First, the addition of logarithms $l_1 + l_2$ is done and then the antilogarithm of the result is computed.

Element $u \in GF(p^n)$	$\log_{2x+1}(u)$
1	0
x	1
$2x + 1$	2
$2x + 2$	3
2	4
$2x$	5
$x + 2$	6
$x + 1$	7

Table 1. Logarithm table of $GF(3^2)$ using the primitive element $2x + 1$.

Torres-Jimenez et al. (2011) proposed an algorithm for the construction of logarithm tables for Galois Fields $GF(p^n)$. The pseudocode is shown in Algorithm 5.1. The algorithm simultaneously finds a primitive element and constructs the logarithm table for a given $GF(p^n)$.

Algorithm 5.1: BUILDLOGARITHMTABLE(p, n)

for each $\rho \in GF(p^n) - 0$
do {
  $\mathcal{L} \leftarrow \varnothing$
  $\mathcal{P}(x) \leftarrow 1$
  $k \leftarrow 0$
  while $(\mathcal{P}(x), k) \notin \mathcal{L}$ **and** $k < p^n - 1$
  do {
    $\mathcal{L} \leftarrow \mathcal{L} \bigcup (\mathcal{P}(x), k)$
    $k \leftarrow k + 1$
    $\mathcal{P}(x) \leftarrow p * \mathcal{P}(x)$
  }
  if $k = p^n - 1$
  then {**return** (ρ)
}
return $(\mathcal{L})$

Now, it follows the presentation of the core of this chapter, the efficient implementation of the Bush construction for OAs, based on a modification of the algorithm presented in this section.

6. Efficient construction of OAs

The idea that leads to an efficient construction of OAs through the Bush's construction relies on the algorithm proposed in (Torres-Jimenez et al., 2011). This algorithm computes the logarithm tables and the primitive element of a given Galois Field $GF(v)$. In this chapter, it is proposed an extension of this algorithm such that it can be used in combination with the Bush's construction to efficiently construct OAs of index unity. The result is an algorithm that uses only additions and modulus operations to evaluate the polynomials $y_j(x)$.

Let's show an example of this contribution. Suppose that it is wanted to construct the $OA(4^3; 3, 5, 4)$. This array has an alphabet $v = p^n = 2^2 = 4$ and size 64×5. To construct it, it is required the polynomial $x + 1$ as the primitive element of $GF(2^2)$, and the logarithm table shown in Table 2(a) (both computed using the algorithm in (Torres-Jimenez et al., 2011)). Table 2(b) is a modified version of the logarithm table that contains all the elements $e_i \in GF(2^2)$ (this includes e_0, the only one which can not be generated by powers of the primitive element).

(a)

Power	**Polynomial in GF(2^2)**
0	1
1	x
2	$x+1$

(b)

Element $e_i \in$ GF(2^2)	**Polynomial in GF(2^2)**
e_0	0
e_1	1
e_2	x
e_3	$x+1$

Table 2. Logarithm table for $GF(2^2)$, with primitive element $x + 1$.

The following step in the construction of the OA is the construction of the matrix $\mathcal{M}$. For this purpose, firstly it is labeled its first v columns with the elements $e_i \in GF(2^2)$; after that, the rows are labeled with all the polynomials of maximum degree 2 and coefficients $e_j \in GF(2^2)$. Next, it is defined the integer value u for each cell $m_{j,i} \in \mathcal{M}$, where $0 \leq j \leq v^t - 1$ and

$0 \leq i \leq v-1$, as the one satisfying $y_j(e_i) = e_u$. Finally, it is generated the values of cell $m_{j,i}$, where the column $i = v$, using the value of the leading coefficient of the polynomial $y_j(x)$, for each $0 \leq j \leq v^t - 1$. Table 3 shows part of the construction of the $OA(4^3; 3, 5, 4)$ through this method.

$\mathcal{M}$		**Elements of** $GF(2^2)$				
		e_0	e_1	e_2	e_3	
	$y_j(x)$ **Polynomial**	**0**	**1**	**x**	**x + 1**	
0	e_0	$\{u\|y_0(e_0) = e_u\}$	$\{u\|y_0(e_1) = e_u\}$	$\{u\|y_0(e_2) = e_u\}$	$\{u\|y_0(e_3) = e_u\}$	e_0
1	e_1	$\{u\|y_1(e_0) = e_u\}$	$\{u\|y_1(e_1) = e_u\}$	$\{u\|y_1(e_2) = e_u\}$	$\{u\|y_1(e_3) = e_u\}$	e_0
2	e_2	$\{u\|y_2(e_0) = e_u\}$	$\{u\|y_2(e_1) = e_u\}$	$\{u\|y_2(e_2) = e_u\}$	$\{u\|y_2(e_3) = e_u\}$	e_0
3	e_3	$\{u\|y_3(e_0) = e_u\}$	$\{u\|y_3(e_1) = e_u\}$	$\{u\|y_3(e_2) = e_u\}$	$\{u\|y_3(e_3) = e_u\}$	e_0
4	e_1x	$\{u\|y_4(e_0) = e_u\}$	$\{u\|y_4(e_1) = e_u\}$	$\{u\|y_4(e_2) = e_u\}$	$\{u\|y_4(e_3) = e_u\}$	e_0
5	$e_1x + e_1$	$\{u\|y_5(e_0) = e_u\}$	$\{u\|y_5(e_1) = e_u\}$	$\{u\|y_5(e_2) = e_u\}$	$\{u\|y_5(e_3) = e_u\}$	e_0
6	$e_1x + e_2$	$\{u\|y_6(e_0) = e_u\}$	$\{u\|y_6(e_1) = e_u\}$	$\{u\|y_6(e_2) = e_u\}$	$\{u\|y_6(e_3) = e_u\}$	e_0
7	$e_1x + e_3$	$\{u\|y_7(e_0) = e_u\}$	$\{u\|y_7(e_1) = e_u\}$	$\{u\|y_7(e_2) = e_u\}$	$\{u\|y_7(e_3) = e_u\}$	e_0
8	e_2x	$\{u\|y_8(e_0) = e_u\}$	$\{u\|y_8(e_1) = e_u\}$	$\{u\|y_8(e_2) = e_u\}$	$\{u\|y_8(e_3) = e_u\}$	e_0
9	$e_2x + e_1$	$\{u\|y_9(e_0) = e_u\}$	$\{u\|y_9(e_1) = e_u\}$	$\{u\|y_9(e_2) = e_u\}$	$\{u\|y_9(e_3) = e_u\}$	e_0
10	$e_2x + e_2$	$\{u\|y_{10}(e_0) = e_u\}$	$\{u\|y_{10}(e_1) = e_u\}$	$\{u\|y_{10}(e_2) = e_u\}$	$\{u\|y_{10}(e_3) = e_u\}$	e_0
11	$e_2x + e_3$	$\{u\|y_{11}(e_0) = e_u\}$	$\{u\|y_{11}(e_1) = e_u\}$	$\{u\|y_{11}(e_2) = e_u\}$	$\{u\|y_{11}(e_3) = e_u\}$	e_0
12	e_3x	$\{u\|y_{12}(e_0) = e_u\}$	$\{u\|y_{12}(e_1) = e_u\}$	$\{u\|y_{12}(e_2) = e_u\}$	$\{u\|y_{12}(e_3) = e_u\}$	e_0
13	$e_3x + e_1$	$\{u\|y_{13}(e_0) = e_u\}$	$\{u\|y_{13}(e_1) = e_u\}$	$\{u\|y_{13}(e_2) = e_u\}$	$\{u\|y_{13}(e_3) = e_u\}$	e_0
14	$e_3x + e_2$	$\{u\|y_{14}(e_0) = e_u\}$	$\{u\|y_{14}(e_1) = e_u\}$	$\{u\|y_{14}(e_2) = e_u\}$	$\{u\|y_{14}(e_3) = e_u\}$	e_0
15	$e_3x + e_3$	$\{u\|y_{15}(e_0) = e_u\}$	$\{u\|y_{15}(e_1) = e_u\}$	$\{u\|y_{15}(e_2) = e_u\}$	$\{u\|y_{15}(e_3) = e_u\}$	e_0
16	e_1x^2	$\{u\|y_{16}(e_0) = e_u\}$	$\{u\|y_{16}(e_1) = e_u\}$	$\{u\|y_{16}(e_2) = e_u\}$	$\{u\|y_{16}(e_3) = e_u\}$	e_1
17	$e_1x^2 + e_1$	$\{u\|y_{17}(e_0) = e_u\}$	$\{u\|y_{17}(e_1) = e_u\}$	$\{u\|y_{17}(e_2) = e_u\}$	$\{u\|y_{17}(e_3) = e_u\}$	e_1
18	$e_1x^2 + e_2$	$\{u\|y_{18}(e_0) = e_u\}$	$\{u\|y_{18}(e_1) = e_u\}$	$\{u\|y_{18}(e_2) = e_u\}$	$\{u\|y_{18}(e_3) = e_u\}$	e_1
19	$e_1x^2 + e_3$	$\{u\|y_{19}(e_0) = e_u\}$	$\{u\|y_{19}(e_1) = e_u\}$	$\{u\|y_{19}(e_2) = e_u\}$	$\{u\|y_{19}(e_3) = e_u\}$	e_1
20	$e_1x^2 + e_1x$	$\{u\|y_{20}(e_0) = e_u\}$	$\{u\|y_{20}(e_1) = e_u\}$	$\{u\|y_{20}(e_2) = e_u\}$	$\{u\|y_{20}(e_3) = e_u\}$	e_1
21	$e_1x^2 + e_1x + e_1$	$\{u\|y_{21}(e_0) = e_u\}$	$\{u\|y_{21}(e_1) = e_u\}$	$\{u\|y_{21}(e_2) = e_u\}$	$\{u\|y_{21}(e_3) = e_u\}$	e_1
22	$e_1x^2 + e_1x + e_2$	$\{u\|y_{22}(e_0) = e_u\}$	$\{u\|y_{22}(e_1) = e_u\}$	$\{u\|y_{22}(e_2) = e_u\}$	$\{u\|y_{22}(e_3) = e_u\}$	e_1
23	$e_1x^2 + e_1x + e_3$	$\{u\|y_{23}(e_0) = e_u\}$	$\{u\|y_{23}(e_1) = e_u\}$	$\{u\|y_{23}(e_2) = e_u\}$	$\{u\|y_{23}(e_3) = e_u\}$	e_1
24	$e_1x^2 + e_2x$	$\{u\|y_{24}(e_0) = e_u\}$	$\{u\|y_{24}(e_1) = e_u\}$	$\{u\|y_{24}(e_2) = e_u\}$	$\{u\|y_{24}(e_3) = e_u\}$	e_1
25	$e_1x^2 + e_2x + e_1$	$\{u\|y_{25}(e_0) = e_u\}$	$\{u\|y_{25}(e_1) = e_u\}$	$\{u\|y_{25}(e_2) = e_u\}$	$\{u\|y_{25}(e_3) = e_u\}$	e_1
26	$e_1x^2 + e_2x + e_2$	$\{u\|y_{26}(e_0) = e_u\}$	$\{u\|y_{26}(e_1) = e_u\}$	$\{u\|y_{26}(e_2) = e_u\}$	$\{u\|y_{26}(e_3) = e_u\}$	e_1
27	$e_1x^2 + e_2x + e_3$	$\{u\|y_{27}(e_0) = e_u\}$	$\{u\|y_{27}(e_1) = e_u\}$	$\{u\|y_{27}(e_2) = e_u\}$	$\{u\|y_{27}(e_3) = e_u\}$	e_1
28	$e_1x^2 + e_3x$	$\{u\|y_{28}(e_0) = e_u\}$	$\{u\|y_{28}(e_1) = e_u\}$	$\{u\|y_{28}(e_2) = e_u\}$	$\{u\|y_{28}(e_3) = e_u\}$	e_1
29	$e_1x^2 + e_3x + e_1$	$\{u\|y_{29}(e_0) = e_u\}$	$\{u\|y_{29}(e_1) = e_u\}$	$\{u\|y_{29}(e_2) = e_u\}$	$\{u\|y_{29}(e_3) = e_u\}$	e_1
30	$e_1x^2 + e_3x + e_2$	$\{u\|y_{30}(e_0) = e_u\}$	$\{u\|y_{30}(e_1) = e_u\}$	$\{u\|y_{30}(e_2) = e_u\}$	$\{u\|y_{30}(e_3) = e_u\}$	e_1
31	$e_1x^2 + e_3x + e_3$	$\{u\|y_{31}(e_0) = e_u\}$	$\{u\|y_{31}(e_1) = e_u\}$	$\{u\|y_{31}(e_2) = e_u\}$	$\{u\|y_{31}(e_3) = e_u\}$	e_1
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Table 3. Example of a partial construction of the $OA(4^3; 3, 4, 5)$, using the Bush's construction.

During the definition of values e_u, the polynomials $y_j(e_i)$ must be evaluated. For example, the evaluation of the polynomial $y_{14} = e_3x + e_1$ at value $x = e_2$ yields $y_{14}(e_2) = e_3x + e_1 = e_3 \cdot e_2 + e_1 = e_0$. To obtain the result e_0 it is necessary to multiply the polynomials e_3 and e_2, and to add the result to e_1. Here is where lies the main contribution shown in this chapter, it is proposed to use the primitive element and the logarithm table constructed by the algorithm in (Torres-Jimenez et al., 2011) to do the multiplication through additions. To do that they are used equivalent powers of the primitive element of the elements $e_i \in GF(2^2)$ involved in the operation, e.g. instead of multiplying $(x+1) \cdot (x)$ we multiply $x^2 \cdot x^1$. Then, the sum of indices does the multiplication, and the antilogarithm obtains the correct result in $GF(2^2)$. For the case of $x^2 \cdot x^1$ the result is $x^3 = x^0 = e_1$. Finally, we add this result to e_1 to complete the operation (this yield the expected value e_0). Note that whenever and operation yields a result outside of the field, a modulus operations is required.

The pseudocode for the construction of OAs using the Bush's construction and the logarithm tables is shown in Algorithm 6.1. The logarithm and antilogarithm table $\mathcal{L}_{i,j}$ is obtained through the algorithm reported by Torres-Jimenez et al. (2011). After that, each element e_i and each polynomial $y_j(x)$ in $GF(p^n)$ are considered as the columns and rows of $\mathcal{M}$, the OA that is being constructed. Given that the value of each cell $m_{i,j} \in \mathcal{M}$ is the index u of the element $e_u \in GF(p^n)$ such that $y_j(e_i) = e_u$, the following step in the pseudocode is the evaluation of the polynomial $y_j(x)$. This evaluation is done by determining the coefficient of each term $a_k \in y_j(x)$ and its index, i.e. the value of the element $e_l \in GF(p^n)$ that is the coefficient of a_k, and then adding it to $i \cdot d$ (the index of e_i raised to the degree of the term a_k). A modulus operation is applied to the result to obtained v, and then the antilogarithm is used over v such that the index it is able to get the value u of the element e_u. Remember that the algorithm `BuildLogarithmTable` simultaneously find the primitive element and computes the logarithm and antilogarithm tables.

Algorithm 6.1: BUILDORTHOGONALARRAY(p, n)

$\mathcal{L} \leftarrow$ `BuildLogarithmTable`(p, n)
$\mathcal{M} \leftarrow \varnothing$
for each `element` $e_i \in GF(p^n)$
do
- $c \leftarrow i$
- **for each** `polynomial` $y_j(x) \in GF(p^n)$
  - **do**
    - $r \leftarrow j$
    - **for each** `term` $a_k \in y_j(x)$
      - **do**
        - $d \leftarrow$ `GetDegree`(a_k)
        - $l \leftarrow$ `GetIndexCoefficient`(a_k)
        - $v \leftarrow (i \cdot d + l)$`mod`$(p^n - 1)$
        - $s \leftarrow \mathcal{L}_{v,1}$
    - $m_{r,c} \leftarrow s$

return $(\mathcal{M})$

Note that in the pseudocode the more complex operation is the module between integers, which can be reduced to shifts when $GF(p^n)$ involves powers of two. This fact makes the algorithm easy and efficient for the construction of OAs, requiring only additions to operate, and modulus operations when the field is over powers of primes different of two. After the

construction of the OA, the number of operations required by the algorithm are bounded by $O(N \cdot t^2)$, due to it requires t operations for the construction of an OA matrix of size $N \times (t + 1)$.

7. Efficient constructions of CAs

This section analyzes the case when Covering Arrays can be constructed from cyclotomy by rotating a vector created from an OA(Colbourn, 2010). It is another process that can be benefited from the previously constructed logarithm tables. The cyclotomy process requires the test of different cyclotomic vectors for the construction of CAs. This vectors can be constructed using the logarithm table. The rest of the section details a bit more about CAs and this process of construction.

Definition 2 (Covering Array)**.** *Let N, t, k, v be positive integers with $t \leq N$. A covering array $CA(N; t, k, v)$, with strength t and alphabet size v is an $N \times k$ array with entries from $\{0, 1, ..., k-1\}$ and the property that any $N \times t$ sub-array has all v^t possible t-tuples occurring at least once.*

Figure 9 shows the corresponding $CA(9; 2, 4, 3)$. The strength of this CA is $t = 2$ and the alphabet is $v = 3$, hence the combinations $\{0,0\}$, $\{0,1\}$, $\{0,2\}$, $\{1,0\}$, $\{1,1\}$, $\{1,2\}$, $\{2,0\}$, $\{2,1\}$, $\{2,2\}$ appear at least once in each subset of size $N \times 2$ of the CA. The CAs are commonly used instead of full experimental designs (FED) when constructing test sets, it is so because the relaxation produced by the use of a small interaction in a CA $t = 2$ (pair-wise) significantly reduce the number of test cases in a test set, implying in some cases savings of more than 90 percent in costs (time or other resources); the confidence level of the testing using combinatorial objects as CA increases with the interaction level involved (Kuhn et al., 2008).

When a CA contains the minimum possible number of rows, it is optimal and its size is called the *Covering Array Number* (CAN). The CAN is defined according to Equation 5.

$$CAN(t, k, v) = \min_{N \in \mathbb{N}} \{N : \exists\, CA(N; t, k, v)\}. \tag{5}$$

The trivial mathematical *lower bound* for a covering array is $v^t \leq CAN(t, k, v)$, however, this number is rarely achieved. Therefore determining achievable lower bounds is one of the main research lines for CAs; this problem has been overcome with the reduction of the known upper bounds. The construction of cyclotomic matrices can help to accomplish this purpose.

According to Colbourn (2010), a cyclotomic matrix (CM) is an array $\mathcal{O}$ of size $k \times k$ that is formed by k rotations of a vector of size k (called starter vector). Table 4 gives an example of a CM.

0	0	0	1	0	1	1
0	0	1	0	1	1	0
0	1	0	1	1	0	0
1	0	1	1	0	0	0
0	1	1	0	0	0	1
1	1	0	0	0	1	0
1	0	0	0	1	0	1

Table 4. CM of size 7×7 formed from the starter vector $\{0, 0, 0, 1, 0, 1, 1\}$. This matrix is a $CA(7; 2, 7, 2)$.

The strategy to construct a cyclotomic matrix involves the identification of a good vector starter. This task can be facilitated using the logarithm table derived from a Galois field. The construction is simple. The first step is the generation of the logarithm table for a certain $GF(p^n)$. After that, the table is transposed in order to transform it into a vector starter v. Then, by using all the possible rotations of it, the cyclotomic matrix is constructed. Finally, the validation of the matrix is done such that a CA can be identified.

$$\begin{pmatrix} 0&0&0&0 \\ 0&1&1&1 \\ 0&2&2&2 \\ 1&0&1&2 \\ 1&1&2&0 \\ 1&2&0&1 \\ 2&0&2&1 \\ 2&1&0&2 \\ 2&2&1&0 \end{pmatrix}$$

Fig. 9. Covering array where $N = 9$, $t = 2$, $k = 4$ and $v = 3$.

Figure 10 shows an example of a cyclotomic matrix.

(a) Vector Starter

0	0	1	1	0	0	0	0	0	1	1	0

(b) Cyclotomic matrix

$$\begin{pmatrix} 0&0&1&1&0&0&0&0&0&0&1&1&0 \\ 0&1&1&0&0&0&0&0&0&1&1&0&0 \\ 1&1&0&0&0&0&0&0&1&1&0&0&0 \\ 1&0&0&0&0&0&0&1&1&0&0&0&1 \\ 0&0&0&0&0&0&1&1&0&0&0&1&1 \\ 0&0&0&0&0&1&1&0&0&0&1&1&0 \\ 0&0&0&0&1&1&0&0&0&1&1&0&0 \\ 0&0&0&1&1&0&0&0&1&1&0&0&0 \\ 0&0&1&1&0&0&0&1&1&0&0&0&0 \\ 0&1&1&0&0&0&1&1&0&0&0&0&0 \\ 1&1&0&0&0&1&1&0&0&0&0&0&0 \\ 1&0&0&0&1&1&0&0&0&0&0&0&1 \\ 0&0&0&1&1&0&0&0&0&0&0&1&1 \end{pmatrix}$$

Fig. 10. Example of a cyclotomic vector V, or a vector starter, and the cyclotomic matrix formed with it. The matrix constitutes a $CA(13; 2, 13, 2)$.

The pseudocode to generate the cyclotomic vector and construct the CA is presented in Algorithm 7.1. There, the algorithm `BuildLogarithmTable(p,n)` is used to construct the table of logarithm and antilogarithms $\mathcal{L}$, where the i^{th} row indicate the element $e_i \in GF(p^n)$, and the column 0 its logarithm, and the column 1 its antilogarithm. The first step is the construction of the vector starter $\mathcal{V}$, which is done by transposing the logarithm table $\mathcal{L}_{*,0}$, i.e. the first column of $\mathcal{L}$. After that, the cyclotomic matrix $\mathcal{M}$ is constructed by rotating the vector starter p^n times, each time the vector rotated will constituted a row of $\mathcal{M}$. Finally, the cyclotomic matrix $\mathcal{M}$ must be validated as a CA to finally return it; one strategy to do so is the parallel algorithm reported by Avila-George et al. (2010).

Algorithm 7.1: BUILDCOVERINGARRAY(p, n)

$\mathcal{L} \leftarrow$ `BuildLogarithmTable`(p, n)
for each $e_i \in GF(p^n)$
  do $\{\mathcal{V}_i \leftarrow \mathcal{L}_{i,0}$
for each $e_i \in GF(p^n)$
  do $\begin{cases} \textbf{for each } e_j \in GF(p^n) \\ \quad \textbf{do} \begin{cases} k \leftarrow (i+j) \texttt{mod} (p^n) \\ m_{i,j} \leftarrow \mathcal{V}_k \end{cases} \end{cases}$
if `IsACoveringArray`$(\mathcal{M})$
  then $\{return \mathcal{M}$
  else $\{return \varnothing$

The following section presents some results derived from the research presented so far in this chapter.

8. Results

An example of one of the best known upper bounds for CAs constructed through the use of cyclotomic matrices is shown in Figure 11; the construction of such table was done with aid of the implementation proposed in this chapter.

The results from the experiment are found in the repository of CAs of Torres-Jimenez [1]. Some of the CAs matrices presented there are derived from the use of cyclotomic vectors constructed through the process described in the previous section, benefiting from the construction of the logarithm tables. Table 5 shows new upper bounds derived from this process.

[1] `http://www.tamps.cinvestav.mx/~jtj/CA.php`

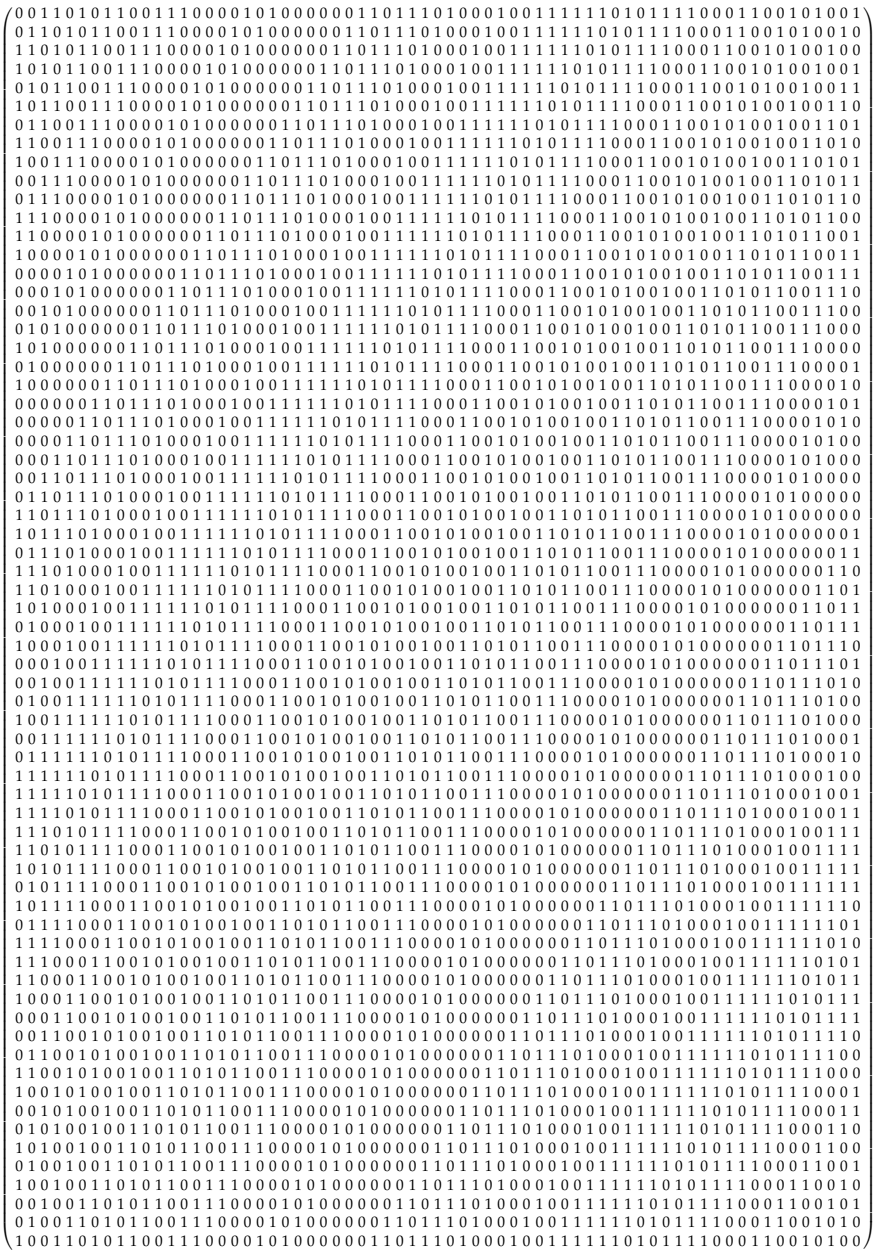


Fig. 11. $CA(67;4,67,2)$ generated through a cyclotomic matrix. This CA is the best known upper bound so far.

k	N	Algorithm
1231	1231	Cyclotomy
1283	1283	Cyclotomy
1319	1319	Cyclotomy
1361	1361	Cyclotomy
1367	1367	Cyclotomy
1373	1373	Cyclotomy
1381	1381	Cyclotomy
1423	1423	Cyclotomy
1427	1427	Cyclotomy
1429	1429	Cyclotomy
1439	1439	Cyclotomy
1447	1447	Cyclotomy
1459	1459	Cyclotomy
1483	1483	Cyclotomy
1487	1487	Cyclotomy
1493	1493	Cyclotomy
1499	1499	Cyclotomy
1511	1511	Cyclotomy
1523	1523	Cyclotomy
1549	1549	Cyclotomy
1559	1559	Cyclotomy
1567	1567	Cyclotomy
1571	1571	Cyclotomy
1579	1579	Cyclotomy
1583	1583	Cyclotomy
1597	1597	Cyclotomy
1601	1601	Cyclotomy
1607	1607	Cyclotomy
1609	1609	Cyclotomy
1613	1613	Cyclotomy
1619	1619	Cyclotomy
1621	1621	Cyclotomy
1627	1627	Cyclotomy
1997	1997	Cyclotomy
1999	1999	Cyclotomy
2003	2003	Cyclotomy
2503	2503	Cyclotomy

Table 5. New upper bounds for CAs obtained through cyclotomic matrices.

9. Conclusions

The main objective of this chapter was the presentation of a efficient implementation of the Bush's construction for Orthogonal Arrays (OAs). Also, it was presented a brief summary of the applications of OAs in cryptography, which could be benefited from the implementation. In addition, the algorithm was also applied for the construction of cyclotomy matrices that yielded new upper bounds of CAs.

Hence, the main contribution of this chapter consisted precisely in an algorithm that requires only additions and modulus operations over finite fields for the construction of OAs. To do so, it relies on a logarithm table constructed through a simple method reported in the literature. It is also presented the details for this construction through the code required to be implemented.

Additionally, the algorithm to construct logarithm table was also slightly modified to construct cyclotomy matrices for the construction of CAs. Here, it is presented the matrix of the $CA(67;4,67,2)$ constructed from a cyclotomic matrix; it represents the best upper bound known so far for these parameters of the CA. Also, it is reported a set of 37 upper bounds of CAs obtained by the construction of the cyclotomy matrices constructed with

support of the algorithm reported here. These matrices are available on request in `http://www.tamps.cinvestav.mx/~jtj/CA.php`.

In addition to the efficient implementation of the Bush's construction through logarithm tables of finite fields, this chapter also presents a brief summary of the combinatorial structures called Orthogonal Arrays. The summary included formal definition, and basic notation used in the scientific literature. Additionally, several applications of OAs in cryptography were presented; and also, different methodologies to construct the combinatorial objects were described; among them was the Bush's construction.

10. Acknowledgments

The authors thankfully acknowledge the computer resources and assistance provided by Spanish Supercomputing Network (TIRANT-UV). This research work was partially funded by the following projects: CONACyT 58554, Calculo de Covering Arrays; 51623 Fondo Mixto CONACyT y Gobierno del Estado de Tamaulipas.

11. References

Avila-George, H., Torres-Jimenez, J., Hernández, V. & Rangel-Valdez, N. (2010). Verification of general and cyclic covering arrays using grid computing, *Data Management in Grid and Peer-to-Peer Systmes, Third International Conference, Globe 2010, Bilbao, Spain*, Vol. 6265 of *Lecture Notes in Computer Science*, Springer, pp. 112–123.

Barker, H. A. (1986). Sum and product tables for galois fields, *International Journal of Mathematical Education in Science and Technology* 17: 473 – 485. `http://dx.doi.org/10.1080/0020739860170409`.

Bush, K. (1952). Orthogonal arrays of index unity, *Annals of Mathematical Statistics* 23(3): 426–434.
URL: *http://www.jstor.org/pss/2236685*

Carter, J. & Wegman, M. (1979). Universal classes of hash functions, *Journal of Computer and System Sciences* 18: 143–154. `http://dx.doi.org/10.1016/0022-0000(79)90044-8`.

Colbourn, C. J. (2010). Covering arrays from cyclotomy, *Designs, Codes and Cryptography* 55: 201–219. `http://dx.doi.org/10.1007/s10623-009-9333-8`.

Dawson, E. & Mahmoodian, E. (1993). Orthogonal arrays and ordered threshold schemes, *Australasian Journal of Combinatorics* 8: 27–44.
URL: *http://ajc.maths.uq.edu.au/pdf/8/ocr-ajc-v8-p27.pdf*

Gilbert, E., MacWilliams, F. & Sloane, N. (1974). Codes which detect deception, *The Bell System Technical Journal* 53: 405–424.
URL: *http://www2.research.att.com/ njas/doc/detection.pdf*

Gopalakrishnan, K. & Stinson, D. R. (2008). Applications of orthogonal arrays to computer science, *Ramanujan Mathematical Society, Lecture Notes Series in Mathematics* 7: 149–164.

Hedayat, A., Sloane, N. & Stufken, J. (1999). *Orthogonal Arrays: Theory and Applications*, Springer-Verlag, New York.

Jones, T. & Seberry, J. (1986). Authentication without secrecy, *ARS Combinatoria* 21-A: 115–121.

Kuhn, R., Lei, Y. & Kacker, R. (2008). Practical Combinatorial Testing: Beyond Pairwise, *IT Professional* 10(3): 19–23. `http://doi.ieeecomputersociety.org/10.1109/MITP.2008.54`.

Niederreiter, H. (1990). A short proof for explicit formulas for discrete logarithms in finite fields, *Applicable Algebra in Engineering, Communication and Computing* 1(1): 55–57. http://dx.doi.org/10.1007/BF01810847.

Rao, C. (1946). Hypercube of strength 'd' leading to confounded designs in factorial experiments, *Bulletin of the Calcutta Mathematical Society* 38: 67–78.

Rao, C. (1947). Factorial experiments derivable from combinatorial arrangements of arrays, *Journal of the Royal Statistical Society* 9: 128–139.
URL: *http://www.jstor.org/pss/2983576*

Stinson, D. (1992a). Combinatorial characterizations of authentication codes, *Designs, Codes and Cryptography* 2: 175–187. http://dx.doi.org/10.1007/BF00124896.

Stinson, D. (1992b). An explication of secret sharing schemes, *Designs, Codes and Cryptography* 2: 357–390. http://dx.doi.org/10.1007/BF00125203.

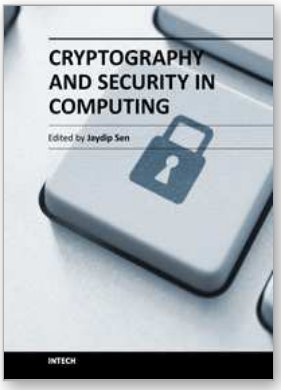
Stinson, D. (1994). Combinatorial techniques for universal hashing, *Journal of Computer and System Sciences* 48: 337–346. http://dx.doi.org/10.1016/S0022-0000(05)80007-8.

Stinson, D. R. (2004). Orthogonal arrays and codes, *Combinatorial Designs*, Springer-Verlag, New York, chapter 10, pp. 225–255.

Taguchi, G. (1994). *Taguchi Methods: Design of Experiments*, American Supplier Institute.

Torres-Jimenez, J., Rangel-Valdez, N., Gonzalez-Hernandez, A. & Avila-George, H. (2011). Construction of logarithm tables for galois fields, *International Journal of Mathematical Education in Science and Technology* 42(1): 91–102. http://dx.doi.org/10.1080/0020739X.2010.510215.

Wegman, M. & Carter, J. (1981). New hash functions and their use in authentication and set equality, *Journal of Computer and System Sciences* 22: 265–279. http://dx.doi.org/10.1016/0022-0000(81)90033-7.



Cryptography and Security in Computing

Edited by Dr. Jaydip Sen

ISBN 978-953-51-0179-6
Hard cover, 242 pages
Publisher InTech
Published online 07, March, 2012
Published in print edition March, 2012

The purpose of this book is to present some of the critical security challenges in today's computing world and to discuss mechanisms for defending against those attacks by using classical and modern approaches of cryptography and other defence mechanisms. It contains eleven chapters which are divided into two parts. The chapters in Part 1 of the book mostly deal with theoretical and fundamental aspects of cryptography. The chapters in Part 2, on the other hand, discuss various applications of cryptographic protocols and techniques in designing computing and network security solutions. The book will be useful for researchers, engineers, graduate and doctoral students working in cryptography and security related areas. It will also be useful for faculty members of graduate schools and universities.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821