

EWSCS'06

Palmse, Estonia
5-10 March 2006

**Lecture 2: Orthogonal Arrays and Error-
Correcting Codes in Cryptography**

James L. Massey

Prof.-em. ETH Zürich, Adjunct Prof., Lund Univ.,
Sweden, and Tech. Univ. of Denmark
Trondhjemsgade 3, 2TH
DK-2100 Copenhagen East

JamesMassey@compuserve.com

Some basic coding definitions and properties

- A q -ary (n, k) linear code is a k -dimensional subspace of the space of n -tuples over $GF(q)$.
- A generator matrix (or encoding matrix) \mathbf{G} is any $k \times n$ matrix whose rows are a basis for the code. (Thus \underline{v} is a codeword if and only if $\underline{v} = \underline{u}\mathbf{G}$ for some k -tuple \underline{u} .)
- A generator matrix is in systematic form if $\mathbf{G} = [\mathbf{I}_k; \mathbf{P}]$ for some $k \times (n-k)$ matrix \mathbf{P} . (After possibly rearranging positions within codewords, every linear code has a generator matrix in systematic form.)
- A parity-check matrix is any matrix \mathbf{H} such that \underline{v} is a codeword if and only if $\underline{v}\mathbf{H}^T = 0$.
- If $\mathbf{G} = [\mathbf{I}_k; \mathbf{P}]$ then $\mathbf{H} = [-\mathbf{P}^T; \mathbf{I}_{n-k}]$ is a parity-check matrix for the code.

- The dual code of an (n,k) linear code is the $(n,n-k)$ linear code whose generator matrices are parity-check matrices of the former code.
- The Hamming distance between two n -tuples is the number of positions in which they differ.
- The minimum distance d_{\min} of a code is the smallest Hamming distance between distinct codewords.

Calculation of Minimum Distance:

The **minimum distance** d_{\min} of an (n,k) linear code with parity-check matrix H **equals the smallest number of distinct columns of H that form a linearly dependent set.**

Example:

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \Rightarrow H = [1 \ 1 \ 1] \Rightarrow d_{\min} = 2.$$

Definition:

A q -ary orthogonal array of power t is a rectangular array of q -ary symbols such that, for some λ , within every choice of t ($t \geq 1$) columns, each of the q^t possible q -ary t -tuples occurs in exactly λ rows. Orthogonal arrays with distinct rows are said to be simple.

Examples:

$$\left. \begin{array}{ccc} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{array} \right\} \begin{array}{l} \text{Binary orthogonal arrays} \\ \text{of power } t = 2 \text{ } (\lambda = 1) \end{array} \left\{ \begin{array}{ccc} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{array} \right.$$

An orthogonal array of power t ($t > 1$) is also an orthogonal array of power $t-1$, etc.

An alternative and more insightful definition:

A q -ary orthogonal array of power t is a rectangular array of q -ary symbols such that, when a row is selected uniformly at random, **the digits in every choice of t ($t \geq 1$) positions are equally likely to be any of the q^t possible q -ary t -tuples.**

Example:

001
010
011
100
101
110

Binary orthogonal
array of power $t = 1$

A simple and useful lemma:

Let G be a generator matrix for a q -ary linear (n, k) code and consider selecting one of the q^k codewords uniformly at random. Then **the digits in τ specified components are equally likely to take on any of the q^τ possible values** if and only if **the corresponding τ columns of G are linearly independent**.

Example: The codewords in the binary linear $(3, 2)$ code with generator matrix

$$G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \text{ are}$$

$000, 101, 011, 110$. When the codewords are equally likely, the digits in any 1 or in any 2 positions are equally likely to take on any of the 4 possible values, but the digits in all 3 positions are not equally likely to take on any of the 8 possible values.

Proof of the lemma:

Let G_τ denote the submatrix of G consisting of the τ columns in the specified positions. The corresponding τ digits of a codeword can be written $\underline{u}G_\tau$ where \underline{u} is the k -tuple of "information digits" and is equally likely to take on any of its q^k possible values. If G_τ has linearly dependent columns, i.e., its rank is less than τ , then there are τ -tuples \underline{v}_τ such that $\underline{u}G_\tau = \underline{v}_\tau$ has no solutions for \underline{u} so the values of \underline{v}_τ cannot be equally likely.

Conversely, if G_τ has full column rank τ , then there are solutions \underline{u} for every \underline{v}_τ . Moreover, the number of these solutions is equal to the number of solutions of the homogeneous equation $\underline{u}G_\tau = 0$. Thus all values of \underline{v}_τ are equally likely.

Error-correcting codes are a rich source of OA's!

Proposition: The **maximum t** , for which a $q^k \times n$ q -ary array, whose rows are the codewords of a q -ary linear (n, k) code with $k < n$ and **dual-code distance $d^\perp > 1$** , is an **orthogonal array** [necessarily simple] of **power t** , is **$t = d^\perp - 1$** . (This holds also for a q -ary nonlinear (n, k) code if d^\perp is taken as the so-called "dual distance".)

Example: The $(3, 2)$ binary linear code with generator matrix $G = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ has $d^\perp = 3$.

The codeword array is: $\left. \begin{array}{ccc} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{array} \right\}$ Binary orthogonal array of power **$t = d^\perp - 1 = 2$**

A less obvious example:

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

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

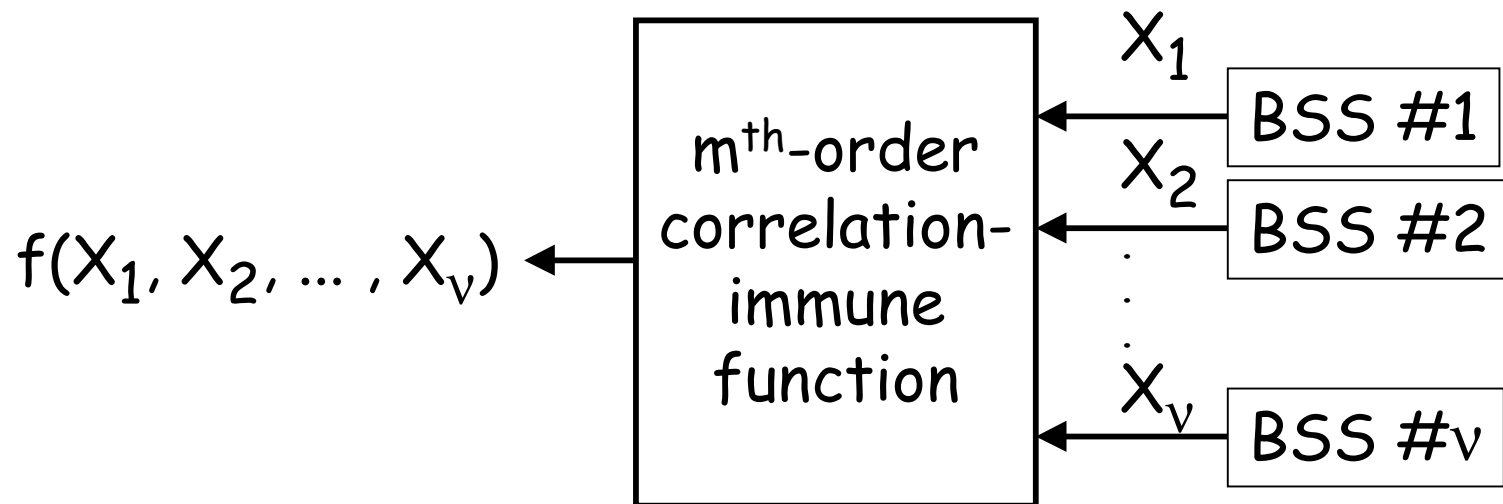
$$\Rightarrow d^\perp = 3$$

Thus, the array of codewords

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

is an orthogonal array
of power $t = d^\perp - 1 = 2$.

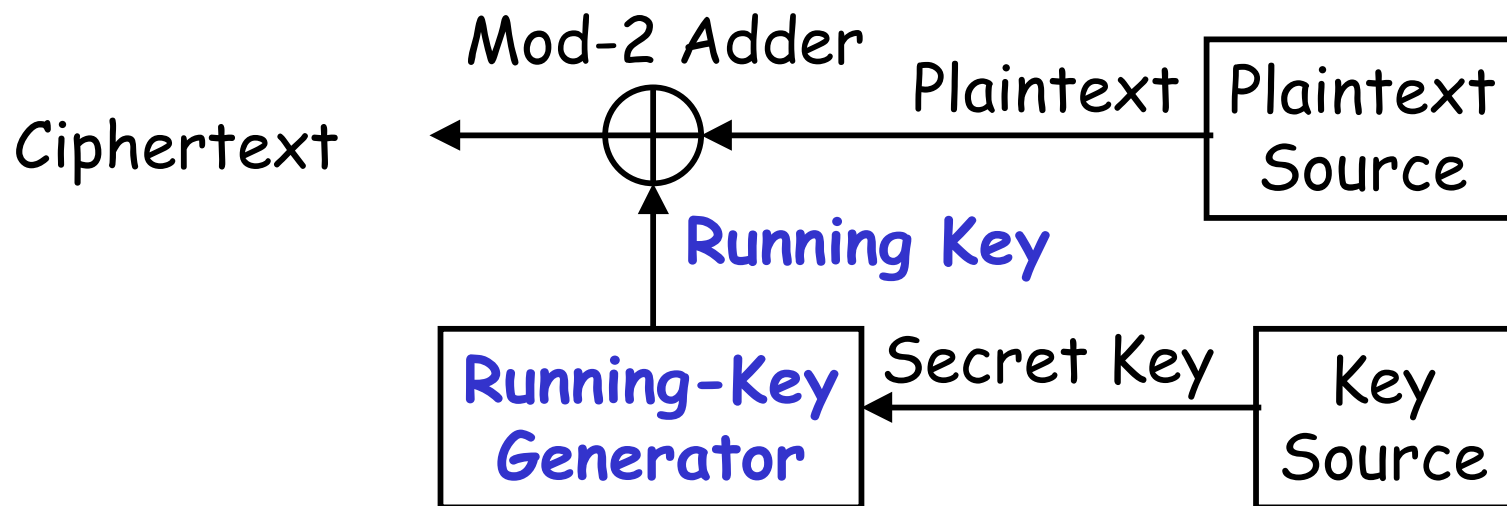
The boolean function $f(x_1, x_2, \dots, x_v)$ is said to be **m^{th} -order correlation immune** if, when the inputs are independent balanced binary random variables, **the output is independent of every set of m or fewer of the inputs.**



BSS = "Binary Symmetric Source"

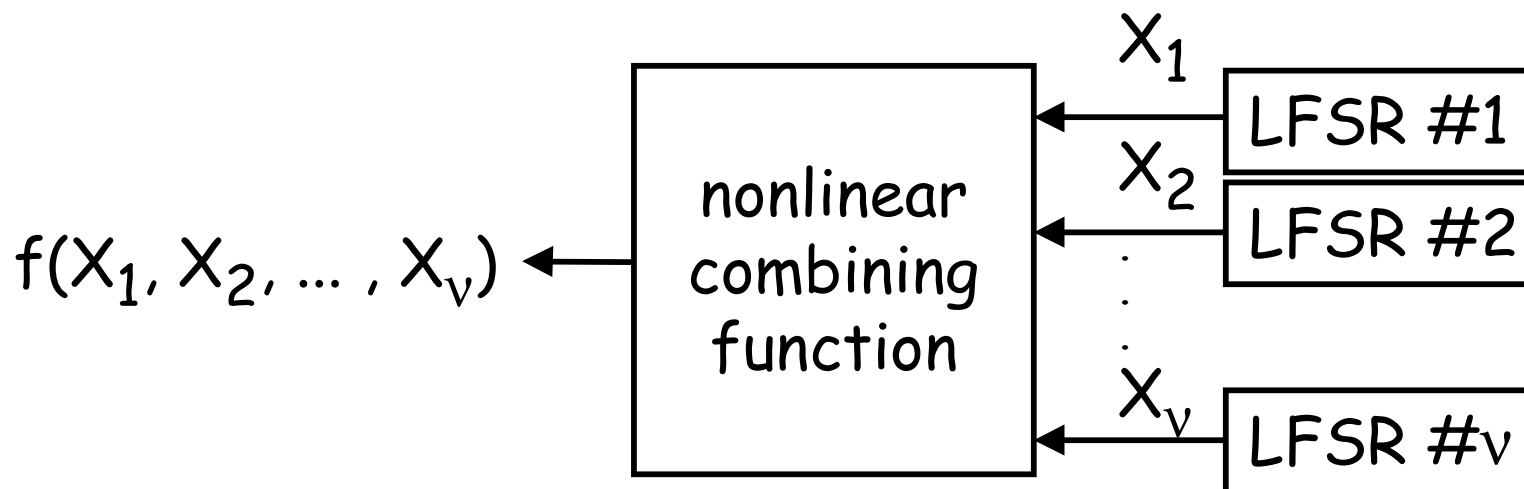
T. Siegenthaler, "Correlation-Immunity of Nonlinear Combining Functions for Cryptographic Applications," *IEEE Trans. Info. Th.*, vol. IT-30, pp. 776-780, Oct. 1984.

An **Additive Stream Cipher** has the Structure:



One tries to design the running-key generator so that the running key cannot be distinguished by an attacker from a BSS output sequence.

Why Siegenthaler introduced correlation-immunity:
He was interested in running-key generators of the following kind for additive stream ciphers:



where the initial states of the linear feedback shift registers (LFSRs) are determined by the secret key. If the nonlinear combining function is m^{th} -order correlation-immune, then the enemy cryptanalyst must attack the LFSRs at least $m+1$ at a time.

An immediately consequence of the definition of correlation-immunity:

Proposition: A boolean function $f(x_1, x_2, \dots, x_v)$ is **m-th order correlation immune** if and only if both

- the set of **arguments** $[x_1, x_2, \dots, x_v]$ for which $f = 1$ form the rows of an **orthogonal array of power m**, and
- the set of **arguments** $[x_1, x_2, \dots, x_v]$ for which $f = 0$ form the rows of an **orthogonal array of power m**.

N.B. These two arrays are **disjoint** and their **union is the set of all 2^v binary v -tuples**.

N.B. The function need not be **balanced** (but it is usually desirable that it be so). If f is balanced, the two arrays will each have **2^{v-1}** rows.

Example of an unbalanced first-order correlation-immune combining function:

x_1 x_2 x_3	$f(x_1 x_2 x_3)$
0 0 1	0
0 1 0	0
0 1 1	0
1 0 0	0
1 0 1	0
1 1 0	0
0 0 0	1
1 1 1	1

Binary orthogonal array of power $t = 1$

Binary orthogonal array of power $t = 1$

How can we use codes to construct correlation-immune functions?

Useful fact #1: If A is a q -ary **orthogonal array of power t** having n columns and \underline{e} is a q -ary **n -tuple**, then the **array $A + \underline{e}$** obtained by adding \underline{e} to each row of A is **also an orthogonal array of power t** .

Example: Adding $\underline{e} = [0 \ 0 \ 1]$ to each row of the array

0 0 0		0 0 1	
1 0 1	gives the array	1 0 0	which is also an
0 1 1		0 1 0	OA of power $t = 2$.
1 1 0		1 1 1	

N.B. If A is the array of codewords in a linear code, then $A + \underline{e}$ is a coset of the code. Cosets are either identical or disjoint.

Useful fact #2: If A_1 and A_2 are OA's of power t having the same number of columns, then **stacking A_1 on top of A_2 gives another orthogonal array of power t .**

Example:

Codewords of a
(6,3) binary linear
code

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

orthogonal array
of power $t = 2$

Coset of the
above code

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

orthogonal array
of power $t = 2$

The array of 16 rows has power $t = 2$.

If A is the array of codewords in an (n,k) **nonlinear** code that is **systematic**, i.e., there is **some set of k coordinates** such that **no two distinct codewords coincide in all these coordinates** and \underline{e} is an n -tuple that is **all-zero in these coordinates**, then $A + \underline{e}$ is a **pseudo-coset** of the code. Distinct pseudo-cosets are disjoint.

Useful fact #3: Pseudo-cosets are orthogonal arrays of the same power as their parent code. Moreover, they partition the set all q -ary n -tuples.

D. R. Stinson and J. L. Massey, "An Infinite Class of Counterexamples to a Conjecture Concerning Non-Linear Resilient Functions," *J. of Cryptology*, Vol 8, No. 3, pp. 167-173, 1995.

How to make a boolean function $f(x_1, x_2, \dots, x_\nu)$ that is correlation-immune of order m and balanced:

- Choose an (ν, k) binary code, **either linear or nonlinear but systematic**, with dual distance $d^\perp = m+1$.
- Form $2^{\nu-k} - 1$ (pseudo-)cosets of this code.
- Choose b as either 0 or 1.
- Assign f the value b for all arguments that are ν -tuples in the code or in the chosen cosets and assign f the value that is the complement of b for all remaining arguments.

If we know the boolean function, how can we find its order of correlation immunity in a practical way?

Let ξ be a **primitive N^{th} root of unity** in a field F , i.e., $\xi^N = 1$ but $\xi^i \neq 1$ for $1 \leq i < N$.

If ξ is a **primitive N^{th} root of unity** in a field F , then the powers ξ^i for $i = 0, 1, \dots, N-1$ are distinct and are all the zeroes of the polynomial $1 - D^N$.

The **Discrete Fourier Transform (DFT)** of length N generated by ξ , a primitive N^{th} root of unity in a field F , is the mapping **$\text{DFT}_\xi(\cdot)$** from F^N to F^N such that **$\underline{B} = \text{DFT}_\xi(\underline{b})$** means

$$B_i = \sum_{n=0}^{N-1} b_n \xi^{in}$$

where **$\underline{b} = [b_0, b_1, \dots, b_{N-1}]$** and **$\underline{B} = [B_0, B_1, \dots, B_{N-1}]$** .

The **inverse DFT** is given by

$$b_n = \frac{1}{N} \sum_{i=0}^{N-1} B_i \xi^{-in}$$

where **N** denotes the sum of **N** 1's in the field F .

Exercise: Show that this expression correctly inverts the DFT in any field F . (Hint substitute the DFT definition into the righthand side of the above expression and then make use of the properties of primitive **N**th roots of unity.)

It will be convenient to rewrite the expressions for the **1-dimensional DFT and inverse DFT** as

$$B(i) = \sum_{n \in \{0,1, \dots, N-1\}} b(n) \xi^{in}$$

$$b(n) = N^{-1} \sum_{i \in \{0,1, \dots, N-1\}} B(i) \xi^{-in}.$$

The **ν -dimensional DFT**, i.e., the DFT when one has ν time axes so that one writes a time "point" as a vector $\underline{n} = [n_1, n_2, \dots, n_\nu]$ and a frequency "point" as a vector $\underline{i} = [i_1, i_2, \dots, i_\nu]$, is obtained by doing a 1-dimensional DFT on each time axis.

Note that $\xi^{i_1 n_1} \xi^{i_2 n_2} \dots \xi^{i_\nu n_\nu} = \xi^{\underline{i} \cdot \underline{n}}$.

The ν -dimensional DFT of length N generated by ξ is the mapping $\text{DFT}(\cdot)$ from $(F^N)^\nu$ to $(F^N)^\nu$ defined by $\underline{\mathbf{B}} = \text{DFT}_\xi(\underline{\mathbf{b}})$ in the manner

$$B(\underline{\mathbf{i}}) = \sum_{\underline{\mathbf{n}} \in \{0,1, \dots, N-1\}^\nu} b(\underline{\mathbf{n}}) \xi^{\underline{\mathbf{i}} \cdot \underline{\mathbf{n}}}.$$

The inverse transform is

$$b(\underline{\mathbf{n}}) = N^{-\nu} \sum_{\underline{\mathbf{i}} \in \{0,1, \dots, N-1\}^\nu} B(\underline{\mathbf{i}}) \xi^{-\underline{\mathbf{i}} \cdot \underline{\mathbf{n}}}$$

where

$$B(\underline{\mathbf{i}}) = B(i_1, i_2, \dots, i_\nu) \text{ and } b(\underline{\mathbf{n}}) = b(n_1, n_2, \dots, n_\nu).$$

When the field is the field of **real numbers** and $N = 2$ (i.e., $\xi = -1$), the ν -dimensional DFT is called the ν -dimensional Walsh-Hadamard Transform.

The Walsh-Hadamard Transform of the boolean function $f(n_1, n_2, \dots, n_\nu) = f(\underline{n})$ (whose values are treated as real numbers) is the function $F(i_1, i_2, \dots, i_\nu) = F(\underline{i})$ given by

$$\begin{aligned} F(\underline{i}) &= \sum_{\underline{n} \in \{0,1\}^\nu} f(\underline{n}) (-1)^{\underline{i} \cdot \underline{n}} \\ &= \sum_{\underline{n} \in \{0,1\}^\nu} f(\underline{n}) (-1)^{\underline{i} \cdot \underline{n} \bmod 2}. \end{aligned}$$

The inverse transform is

$$\begin{aligned} f(\underline{n}) &= 2^{-\nu} \sum_{\underline{i} \in \{0,1\}^\nu} F(\underline{i}) (-1)^{-\underline{i} \cdot \underline{n}} \\ &= 2^{-\nu} \sum_{\underline{i} \in \{0,1\}^\nu} F(\underline{i}) (-1)^{\underline{i} \cdot \underline{n} \bmod 2}. \end{aligned}$$

Note that $F(\underline{0}) = \text{number of } \underline{n} \in \{0,1\}^\nu \text{ such that } f(\underline{n}) = 1.$

Example: $\nu = 1$

$$F(0) = f(0) + f(1)$$

$$f(0) = 1/2 [F(0) + F(1)]$$

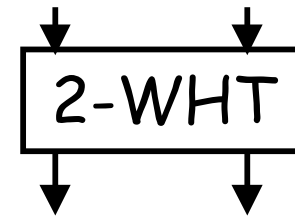
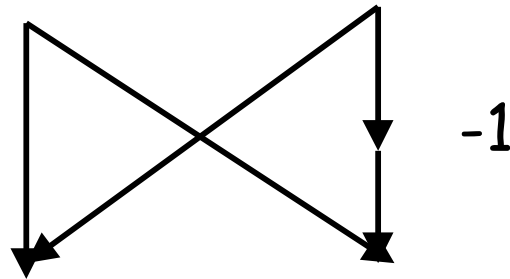
$$F(1) = f(0) - f(1)$$

$$f(1) = 1/2 [F(0) - F(1)]$$

We will call this transform the 2-WHT.

"Butterfly" of the 2-WHT:

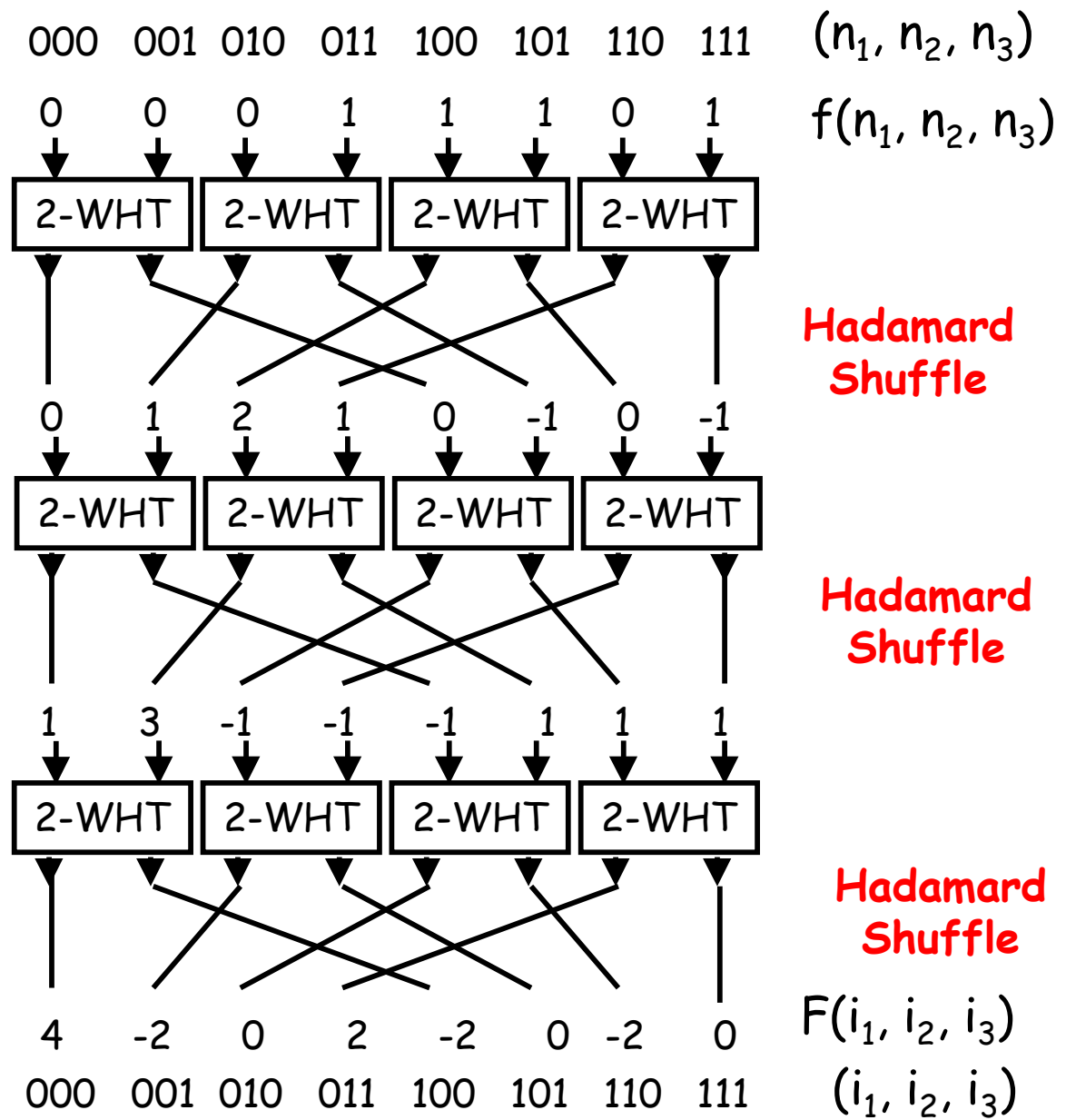
Our notation:



A binary random variable Y is **balanced** if $P[Y = 1] = 1/2$.

If $\underline{X} = [X_1, X_2, \dots, X_\nu]$ is a random vector whose components are independent binary coin-tossing random variables, then $P[f(\underline{X}) = 1] = 2^{-\nu} F(\underline{0})$.
Thus $f(\underline{X})$ is **balanced** if and only if $F(\underline{0}) = 2^{\nu-1}$.

Example
 for $\nu = 3$.
 ("*Geffe's function*")
 $f(n_1, n_2, n_3) = n_1 \bar{n}_2 \oplus n_2 n_3$
 where the overbar denotes binary complementation.)



Lemma: Let $\underline{X} = [X_1, X_2, \dots, X_\nu]$ be a random vector whose components are independent binary coin-tossing random variables (with values treated as the real numbers 0 or 1) and let \underline{i} be non-zero in $\{0,1\}^\nu$. Then $f(\underline{X})$ is independent of $\underline{X} \bullet \underline{i} \bmod 2$ (the "dot product" of \underline{X} and \underline{i} taken modulo 2) if and only if $F(\underline{i}) = 0$.

Proof: Because $\underline{i} \neq \underline{0}$, it follows that $\underline{x} \bullet \underline{i} \bmod 2 = 0$ for exactly half of the 2^ν vectors \underline{x} in $\{0,1\}^\nu$. Thus

$$\begin{aligned}
 & P[f(\underline{X}) = 1 \mid \underline{x} \bullet \underline{i} \bmod 2 = 0] - P[f(\underline{X}) = 1 \mid \underline{x} \bullet \underline{i} \bmod 2 = 1] \\
 &= (1/2^{\nu-1}) [\sum_{\underline{x}: \underline{x} \bullet \underline{i} \bmod 2 = 0} f(\underline{x}) - \sum_{\underline{x}: \underline{x} \bullet \underline{i} \bmod 2 = 1} f(\underline{x})] \\
 &= (1/2^{\nu-1}) \sum_{\underline{x} \in \{0,1\}^\nu} f(\underline{x}) (-1)^{\underline{x} \bullet \underline{i} \bmod 2} \\
 &= (1/2^{\nu-1}) F(\underline{i}).
 \end{aligned}$$

In other words, the WHT of f gives a **complete picture of the dependence** between $f(\underline{X})$ and mod-two sums of the components of \underline{X} .

Example: Geffe's function $f(\underline{X}) = f(X_1, X_2, X_3)$ is **not** independent of $X_1 = \underline{X} \cdot [1, 0, 0] \bmod 2$, **nor** of $X_3 = \underline{X} \cdot [0, 0, 1] \bmod 2$, but it *is* independent of $X_2 = \underline{X} \cdot [0, 1, 0]$. Similarly, $f(\underline{X})$ *is* independent of $X_1 \oplus X_3 = \underline{X} \cdot [1, 0, 1] \bmod 2$, but **not** of $X_1 \oplus X_2 = \underline{X} \cdot [1, 1, 0] \bmod 2$.

Lemma: Let $\underline{Y} = [Y_1, Y_2, \dots, Y_m]$ where Y_1, Y_2, \dots, Y_m are independent coin-tossing binary random variables and let Z be any random variable. Then Z and \underline{Y} are independent if and only if Z is independent of the binary random variable $W_{\underline{i}} = \underline{i} \cdot \underline{Y}$ for every non-zero \underline{i} in $\{0,1\}^m$.

For a simple proof of this lemma using Walsh-Hadamard transforms, see

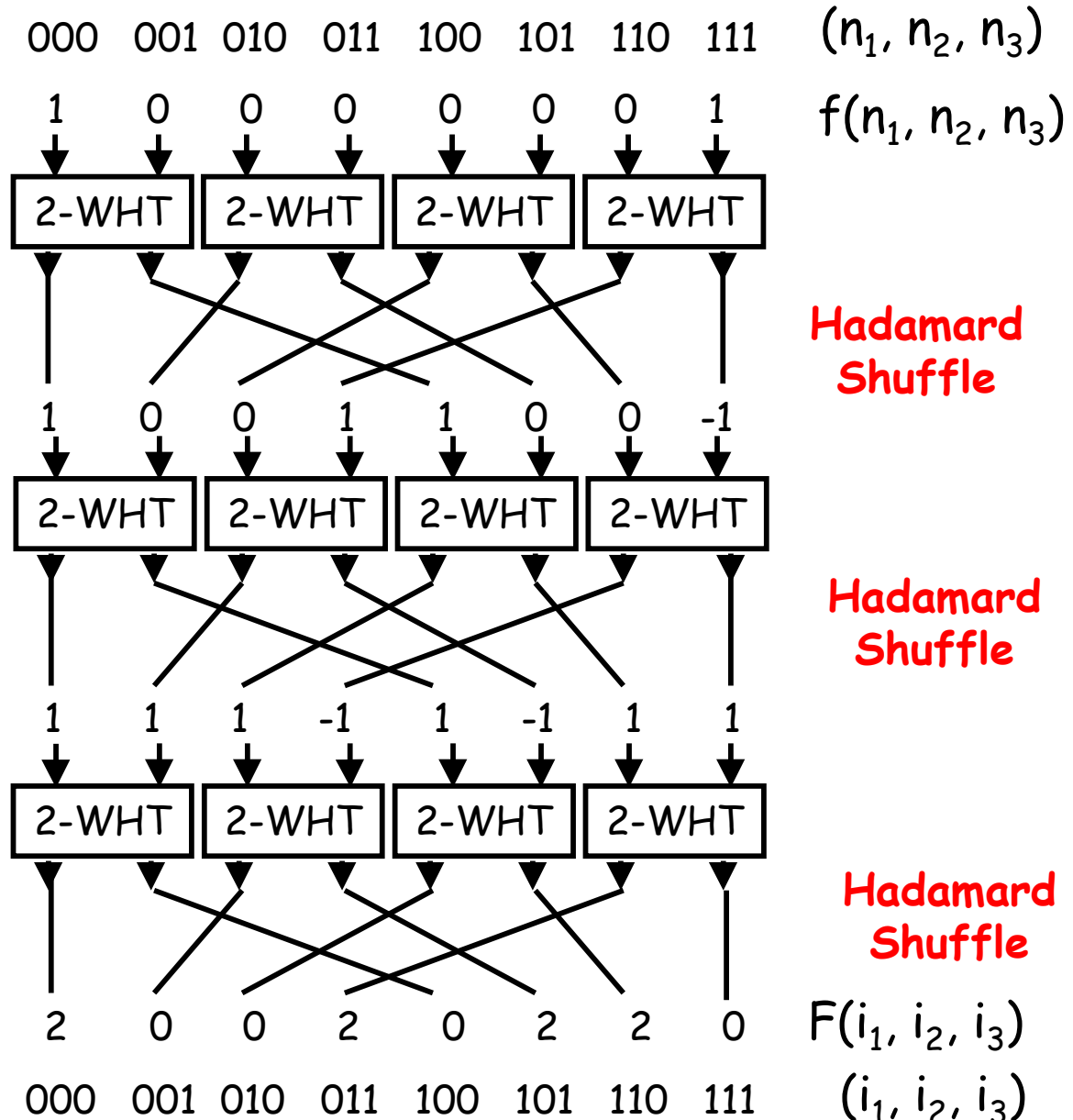
L. Brynielsson, "A Short Proof of the Xiao-Massey Lemma," *IEEE Trans. Info. Th.*, vol. IT-35, p. 1344, Nov. 1989.

Combined with the previous lemma, this lemma now gives the following:

Theorem: $f(n_1, n_2, \dots, n_\nu)$ is m^{th} -order correlation immune if and only if its Walsh-Hadamard transform $F(i_1, i_2, \dots, i_\nu)$ vanishes at all "frequencies" (i_1, i_2, \dots, i_ν) with Hamming weight between 1 and m , inclusive, when the arguments of the function are taken as the "time" indices and the function values treated as the real numbers 0 and 1.

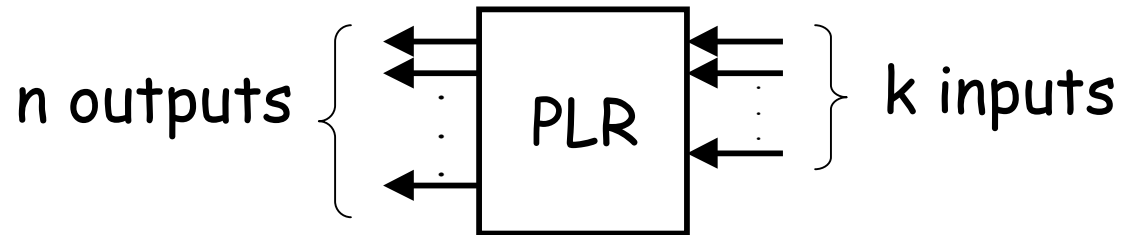
Example: Geffe's function is not even first-order correlation immune.

Another Example for $\nu = 3$.



Unbalanced \Leftarrow
and 1st-order
correlation immune

Perfect Local Randomizers:



An injective (or "one-to-one") function f from k q -ary digits to n q -ary digits ($n > k$) is a perfect local randomizer of order t if **choosing the k input digits uniformly at random** guarantees that the **output digits in every choice of t components** (not necessarily consecutive) of the output n -tuple are **also uniformly random**.

Such a function might be useful in a key-schedule algorithm, or elsewhere within block ciphers.

Maurer, U. M. and Massey, J. L., "Local randomness in pseudorandom sequences," *Journal of Cryptology*, **4**, pp. 135-149, 1991.

Example: A perfect local randomizer of order $\dagger = 2$.

x_1, x_2, x_3	$f(x_1, x_2, x_3)$
0 0 0	0 0 0 0 0 0 0
0 0 1	1 0 0 1 1 1 0
0 1 0	0 1 0 0 1 1 1
0 1 1	0 0 1 1 1 0 1
1 0 0	1 1 0 1 0 0 1
1 0 1	1 0 1 0 0 1 1
1 1 0	0 1 1 1 0 1 0
1 1 1	1 1 1 0 1 0 0

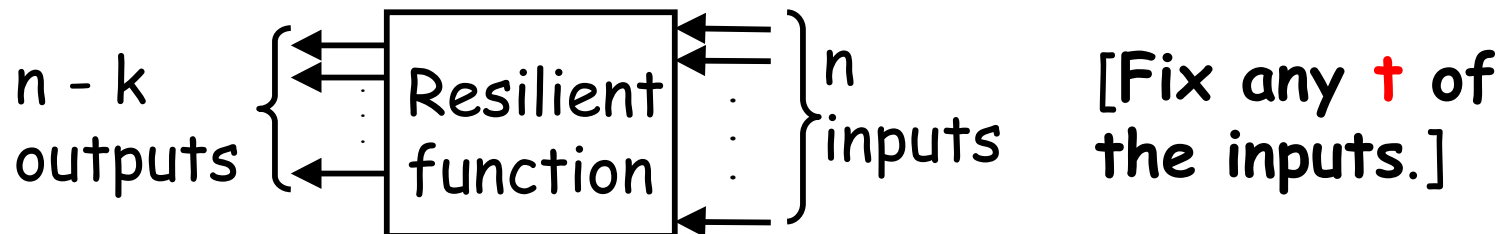
Binary orthogonal array
of power $\dagger = 2$

Proposition: An injective function f from k q -ary digits to n q -ary digits ($n > k$) is a **perfect local randomizer of order t** if and only if **the array having as its rows the range of f is an orthogonal array of power t .**

Thus, we can build perfect local randomizers of order t with the help of error-correcting codes in a manner entirely analogous to that given previously for constructing correlation-immune functions.

Resilient Functions

A function f from n q -ary digits to $n-k$ q -ary digits ($1 < k < n$) is said to be **t -resilient** if, for **every choice of t of the input digits**, when the **values** of these digits are **fixed** and the values of the **other $n-t$ input digits chosen uniformly at random**, **all $n-k$ output digits are uniformly random**.



Resilient functions can be used to thwart divide-and-conquer attacks of various kinds on ciphers.

B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich and R. Smolensky, "The bit extraction problem or t -resilient functions," Proc. 26th IEEE Symp. Foundations Computer Sci. pp. 396-407, 1985.

Example:

The boolean function $y_1 = f(x_1, x_2, x_3)$ with function table:

	x_1	x_2	x_3	y_1
The codewords of (3,2) code with $d^\perp = 3$	0	0	0	1
	1	0	1	1
	0	1	1	1
	1	1	0	1
The only other coset of this code	1	0	0	0
	0	0	1	0
	1	1	1	0
	0	1	0	0

Is a **2-resilient function**. This remains true if the function values are complemented.

Proposition: The function $f(x_1, x_2, \dots, x_n) = [y_1 y_2 \dots y_{n-k}]$ from n q -ary digits to $n - k$ q -ary digits is **t -resilient** if and only if

- f is **balanced**, i.e., the sets $f^{-1}(y_1 y_2 \dots y_{n-k})$ have the same cardinality for all q^{n-k} choices of $y_1 y_2 \dots y_{n-k}$, and
- for each of the q^{n-k} choices of $y_1 y_2 \dots y_{n-k}$, the array whose rows are the set of x_1, x_2, \dots, x_n such that $f(x_1, x_2, \dots, x_n) = [y_1 y_2 \dots y_{n-k}]$, i.e., the set $f^{-1}(y_1 y_2 \dots y_{n-k})$, is an **orthogonal array of power t** .

Thus, we can also build **t -resilient** functions with the help of error-correcting codes in a manner entirely analogous to that given previously for constructing correlation-immune functions.

Exercise: Find the relation between resilient functions from n bits to 1 bit and correlation-immune functions.

Measuring nonlinearity:

The nonlinear order of a boolean function $f(x_1, x_2, \dots, x_n)$ is the maximum degree of its terms when written as a sum of multivariable polynomials

$$f(x_1, x_2, \dots, x_n) = a_0 1 + a_1 x_1 + \dots + a_n x_n + a_{12} x_1 x_2 + a_{13} x_1 x_3 + \dots + a_{n-1,n} x_{n-1} x_n + \dots + a_{12\dots n} x_1 x_2 \dots x_n$$

[which is the so-called algebraic normal form (ANF).]

The nonlinearity of a boolean function is the minimum Hamming distance of its function table from the function table of a linear or affine function, i.e., a function for which only a_0, a_1, \dots, a_n may be non-zero.

Examples:

x_1, x_2, x_3	000	001	010	011	100	101	110	111
$f(x_1, x_2, x_3) = 1$	1	1	1	1	1	1	1	1
$f(x_1, x_2, x_3) = x_1$	0	0	0	0	1	1	1	1
$f(x_1, x_2, x_3) = x_2$	0	0	1	1	0	0	1	1
$f(x_1, x_2, x_3) = x_3$	0	1	0	1	0	1	0	1

The **linear and affine functions** are those whose function tables are linear combinations of those above, i.e., the codewords in the first-order Reed-Muller code of length 2^3 because this code has the generator matrix with these function tables as its rows. These are the functions of **nonlinearity 0**.

Rueppel's observation:

Proposition: The **nonlinearity** of a boolean function of n variables is the **Hamming distance of its function table to the nearest codeword in the first-order Reed-Muller code of length 2^n .**

Examples:

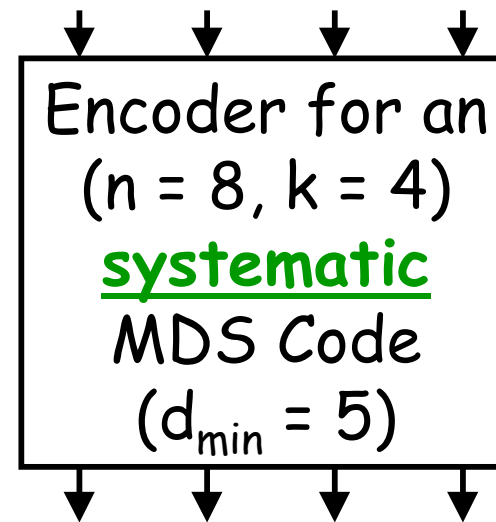
x_1, x_2, x_3	000	001	010	011	100	101	110	111
$f(x_1, x_2, x_3) = x_1x_2$	0	0	0	0	0	0	1	1
$f(x_1, x_2, x_3) = x_1x_3$	0	0	0	0	0	1	0	1
$f(x_1, x_2, x_3) = x_2x_3$	0	0	0	1	0	0	0	1

For instance, the function x_1x_2 has Hamming distance 2 to the nearest linear function and thus **nonlinearity 2**.

Rueppel, R. A., *Analysis and design of stream ciphers*. Heidelberg and New York: Springer 1986.

Diffusion via Linear Codes - example of using a linear code to obtain guaranteed diffusion.

A Maximum-Distance Separable (MDS) code is an (n, k) code with $d_{\min} = n - k + 1$.



By linearity, if the difference of two input 4-tuples is non-zero, then the difference of the two corresponding output 4-tuples will also be non-zero and **the total of non-zero differences among the 8 symbols will be at least $d_{\min} = 5$.**

This method of obtaining diffusion was first suggested by Vaudenay and is used in the AES cipher Rijndael and in several other recent ciphers.

A few more references:

On correlation immunity

G.-Z. Xiao and J.L. Massey, "A Spectral Approach to Correlation-Immune Combining Functions," *IEEE Trans. Info. Th.*, vol. IT-34, pp. 569-571, May 1988.

On uses of coding theory in cryptography

J. L. Massey, "Some Applications of Coding Theory in Cryptography," in *Codes and Cyphers: Cryptography and Coding IV* (Ed. P. G. Farrell). Essex, England: Formara Ltd., pp. 33-47, 1995.

J. L. Massey, "Some Applications of Code Duality in Cryptography," in *Matemática Contemporânea*, vol. 21 (Eds. N. Rocco and S. Sidki). Sociedade Brasileira de Matemática, 2001, pp. 187-209.

If you liked this way of looking at orthogonal arrays in terms of random quantities, see

J. L. Massey, "Randomness, Arrays, Differences and Duality," *IEEE Trans. Info. Theory*, Vol. IT-48, pp. 1698-1703, June 2002