

Applications of Combinatorial Designs to Software Engineering, Cyber Security and Disaster Science

Dr. Dimitris Simos, FTICA

Based on joint works with Charles Colbourn, Bernhard Garn, Ludwig Kampel, Ilias Kotsireas, Temur Kutsia, Manuel Leithner and Michael Wagner.

5th Pythagorean conference, Hotel Horizon Blu, June 6, 2025

Tribute to Design Theorists and some Background of the Talk

- 25 years ago, C. J. Colbourn, J. H. Dinitz and D.R. Stinson, published their seminal survey on applications of combinatorial designs to communications, cryptography, and networking
[Surveys in Combinatorics, 1999. London Mathematical Society Lecture Note Series. Cambridge University Press; pages 37-100]

Γενική Μεταπτυχιακή Εξέταση

Συνδυαστικοί Σχεδιασμοί και εφαρμογές στην Κωδικοποίηση,
την Κρυπτογραφία, και τις Τηλεπικοινωνίες

(Combinatorial Designs and their applications, in Coding Theory,
Cryptography and Communications)

Δημήτριος Ε. Σίμος
Δόκιμος Υποψήφιος Διδάκτορας
Ε.Μ.Π.

- 17 years ago, I presented my PHD proposal based on their work
- 15 years ago, I presented links between designs and telecommunications in PYTHAG4
- I still consider their work, one of the most influential which shaped me as a scientist and all researchers I was privileged to mentor – thank you ☺

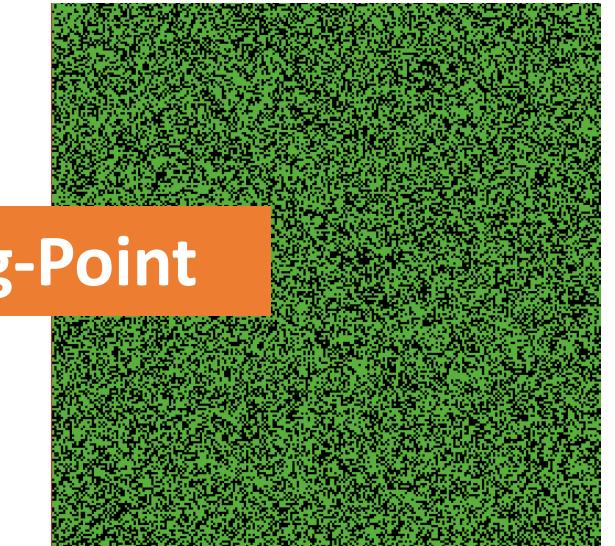
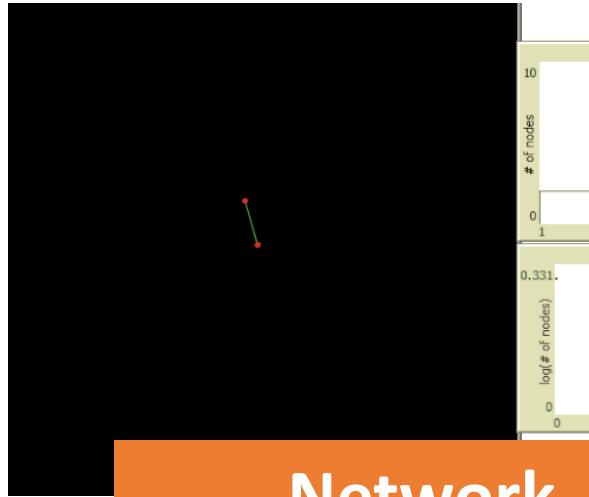
Outline of the Talk

1. Critical Software / Cyber-physical Systems
2. Natural Hazards / Disasters
3. Conclusion & Future Outlook



Introduction

Analysis, Modelling & Simulation of Complex Systems

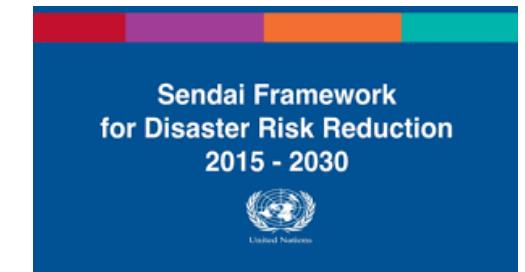


Patterns appear in Every Instance of a Complex System

- **Cyber-physical Systems (Systems Under Test / Attack Paths)**
 - Software/System Failures
 - Security Vulnerabilities
 - Hardware Trojans
- **Disaster Management (Natural / Technological Hazards)**
 - Compound and Cascading Effects of Hazards

Motivation for Combinatorial Designs in Complex Systems

- **Economic Impact:** Software testing may consume up to half of the overall software development cost (“system of systems view”)
 - **Combinatorial explosion:** **Exhaustive search** of input space increases time needed exponentially
 - Added level of complexity for system testing and climate science (modelling real-world environments and vulnerabilities)
- How can we **estimate** the residual risk that **remains** after testing? How can we **guarantee** aspects of test quality (e.g. test coverage, locating faults)?
- **In this Talk:** Formulate **testing, reasoning** and **resilience** problems as **combinatorial problems** and then use efficient **methods** to tackle them



Advancing Testing & Security of Critical Software

Combinatorics beyond Experimental Design

Example: A Large System for Testing

- Suppose we have a system with on-off switches
- 34 switches = $2^{34} = 1.7 \times 10^{10}$ possible settings



- How do we **test** this system?

Example of a Mathematical Structure used in Testing

System Under Test (SUT) with 3 Boolean Input Parameters a, b, c

- Could be function, application, configuration file, etc.
- Exhaustive test set: $2^3 = 8$ tests
- 2-way covering array (test set): 4 tests

| a | b | c | | (a, b) | (b, c) | (a, c) |
|---|---|---|--|--------|--------|--------|
| 0 | 0 | 0 | | (0, 0) | (0, 0) | (0, 0) |
| 0 | 1 | 1 | | (0, 1) | (1, 1) | (0, 1) |
| 1 | 0 | 1 | | (1, 0) | (0, 1) | (1, 1) |
| 1 | 1 | 0 | | (1, 1) | (1, 0) | (1, 0) |

Table 1: 2-way test set (left) covering all pairs of parameters (right)

Covering Arrays $CA(N; t, k, v)$ of Strength t

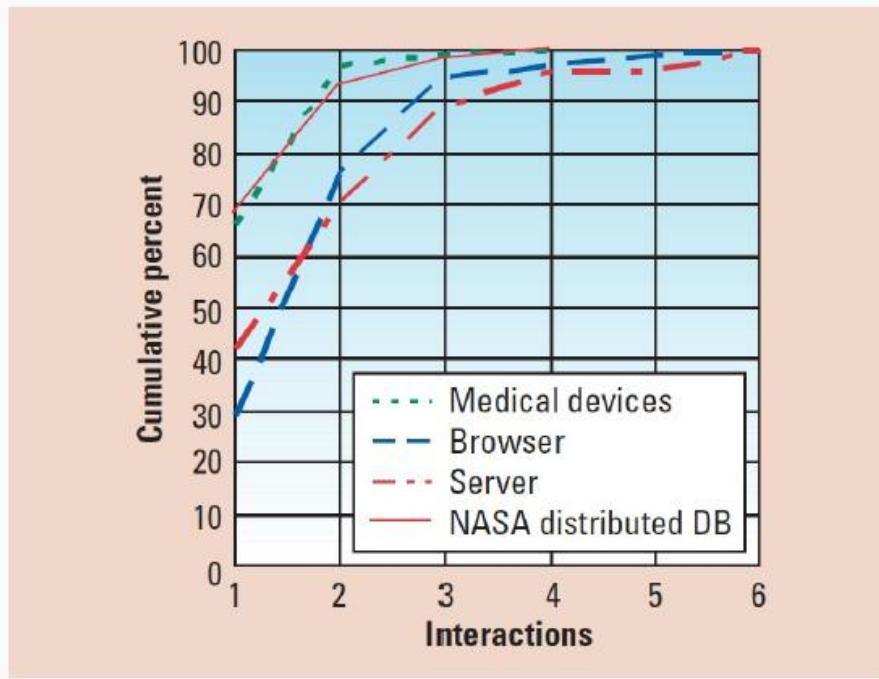
- Cover all t -way combinations of k input parameters at least **once**
- Input parameters have v total values each
- Such a mathematical object has N total rows (tests)

How is this Knowledge Useful?

- Recall the system with on-off switches
- $34 \text{ switches} = 2^{34} = 1.7 \times 10^{10}$ possible settings
- **Assumption: What if we knew no failure involves more than 3 switch settings interacting?**
 - If only **3-way** combinations, need a CA with **only** 33 tests
 - If only **4-way** combinations, need a CA with **only** 85 tests



Empirical Evidence: Fault Coverage vs. Interactions



- The **maximum degree** of interaction observed so far in actual real-world faults is **relatively small** (six)
 - 2-way **interaction**: **age** > 100 and **zip-code** = 5001, DB push **fails**
- Most failures are induced by single factor faults or by the **joint combinatorial effect** (interaction) of two factors, with progressively fewer failures induced by interactions between three or more factors

Combinatorial Testing (CT)

What is Combinatorial Testing?

Combinatorial Strategy for **Higher** Interaction Testing ($t \geq 2$)

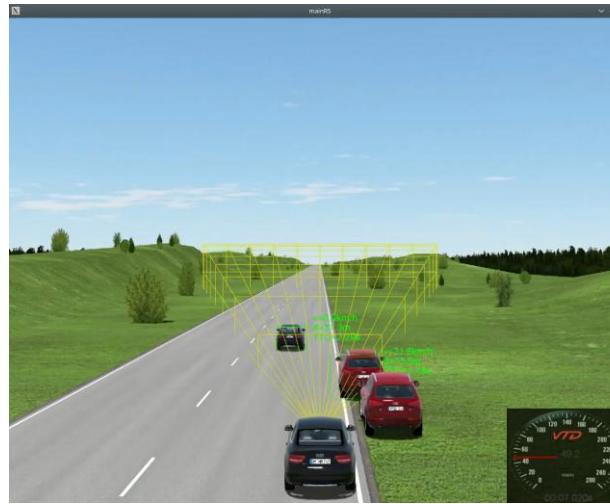
Where it can be Applied?

To **system configurations**, **input data** or both

Key Facts:

- CT utilizes 100% coverage of t -way combinations of k input data or system configuration parameters
- Coverage is provided by **mathematical objects** (covering arrays), that are later transformed to software artifacts
- t -way tests that cover **all** such few parameter (factor) interactions can be very effective and provide **strong assurance**

Virtual Driving Function Testing Problem



Tesla must provide NHTSA with Autopilot recall data by July or face up to \$135 million in fines

PUBLISHED TUE, MAY 7 2024 5:08 PM EDT | UPDATED TUE, MAY 7 2024 7:04 PM EDT

SHARE



List of Tables and Figures

Contents [Edit](#) [View history](#) [Tools](#)

[Article](#) [Talk](#) [Read](#) [Edit](#) [View history](#) [Tools](#)

(Top)

▼ Fatal crashes

- Handan, Hebei, China (January 20, 2016)
- Williston, Florida, USA (May 7, 2016)
- Mountain View, California, USA (March 23, 2016)
- Kenagawa, Japan (April 29, 2016)
- Oroney Beach, Florida, USA (March 1, 2019)
- Key Largo, Florida, USA (April 25, 2019)
- Fremont, California, USA (August 24, 2019)

Tesla Autopilot was released in October 2015, and the first fatal crashes involving the advanced driver assistance system (ADAS) occurred less than one year later. The fatal crashes have attracted attention from news publications and United States government agencies, including the National Transportation Safety Board (NTSB) and National Highway Traffic Safety Administration (NHTSA), which have argued the Tesla Autopilot death rate is higher than the reported estimates.^[1] In addition to the fatal crashes, there have been many nonfatal crashes, the causes have included the ADAS failing to recognize other vehicles, insufficient Autopilot driver engagement, and violating the operational design domain.

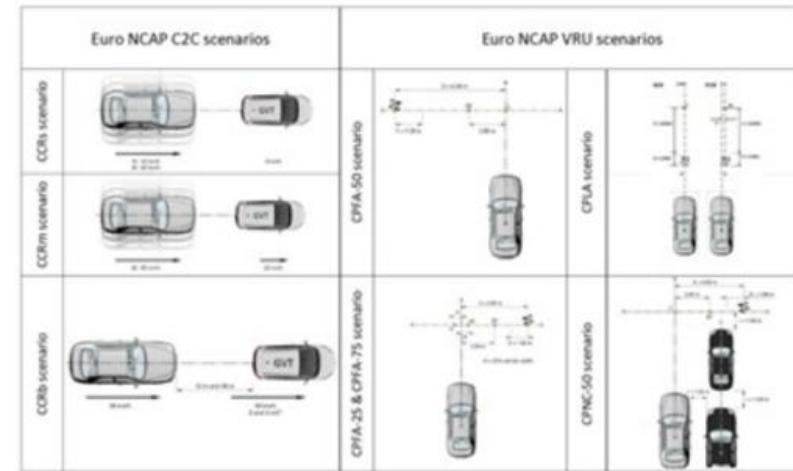
As of June 2024, there have been forty-four verified fatalities involving Autopilot^[2] and hundreds of nonfatal incidents^[3]. Collectively, these have led to a formal investigation by the NHTSA, culminating in a general recall in December 2023 of all vehicles equipped with Autopilot, which was resolved by an over-the-air software update. Immediately after closing its investigation in April 2024, NHTSA opened a recall query to determine the effectiveness of the recall.


The Model S after it was recovered from the crash scene in Williston, Florida

Virtual Driving Scenarios for testing the AEB function

IPM for Driving Scenarios

- Developed and used in previous works (Wotawa et al.^a)
- Description of traffic situation via parameters + values
 - Speed of the car: EgoVehicle1_Start_speed
 - Type of car: EgoVehicle1_VehicleType
 - Position of the car: EgoVehicle1_Offset_s
 - Number of pedestrians: NumberOfPede
- Resulting IPM consists of 39 parameters & 42 constraints:



(3, 3, 31, 3, 5, 1, 6, 4, 12, 12, 12, 10, 14, 12, 12, 12, 10, 14, 31, 4, 3, 20, 9, 3, 3, 31, 4, 3, 20, 9, 3, 3, 31, 4, 3, 20, 9, 3, 3)

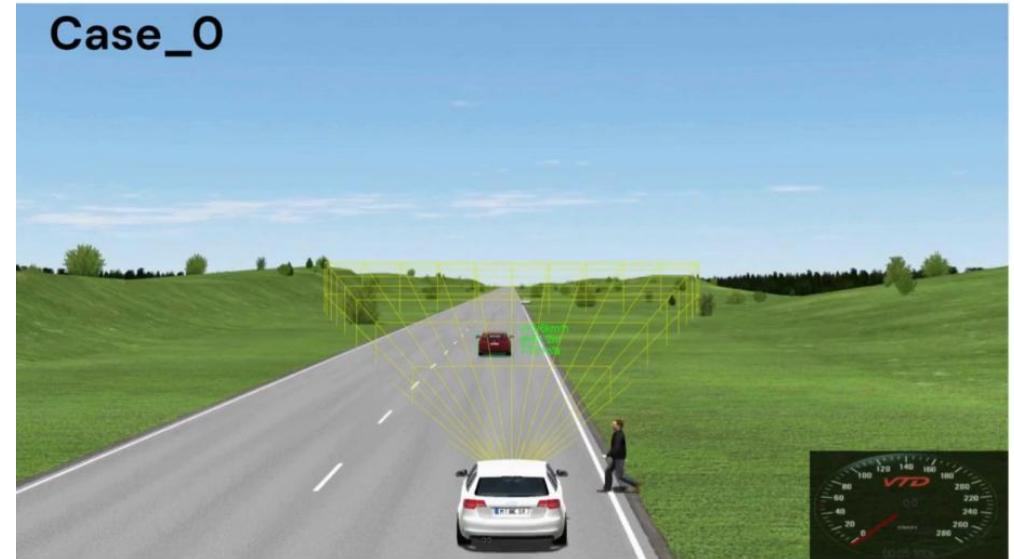
```
1 [Parameter]
2 NumberOfVehiclePlayer (enum) : 1, 2, 3
3 NumberOfPede (enum) : 1, 2, null
4 EgoVehicle1_Start_speed (enum) : 0, 1.388888889, 2.777777778, 4.166666667, 5.555555556, 6.944444444, 8.333333333, 9.722222222, ...
5 EgoVehicle1_VehicleType (enum) : Audi_A3_2009_white, Audi_Q5_2008_red, Audi_S5_2009_black_metallic
6 EgoVehicle1_Offset_s (enum) : -2, -1, 0, 1, 2
7 EgoVehicle1_Offset_t (enum) : 0
8 EgoVehicle1_Rate (enum) : 5, 6, 7, 8, 9, 10
9 EgoVehicle1_Target_Speed (enum) : 14, 28, 42, 55
10 Pedestrians_Objects1_Start_speed (enum) : 0, 0.28, 0.56, 0.83, 1.11, 1.39, 1.6, 1.94, 2.22, 2.5, 2.8, null
11 Pedestrians_Objects1_Offset_s (enum) : 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, null
12 Pedestrians_Objects1_Offset_t (enum) : 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, null
13 Pedestrians_Objects1_Rate (enum) : 0, 1, 2, 3, 4, 5, 6, 7, 8, null
14 Pedestrians_Objects1_Type (enum) : Adult, Child, Wheelchair, Animal, Ballon, Paper, Dog, Stone, Adult_w_Bicycle, Adult_w_child, custom1, custom2, custom3, null
15 Pedestrians_Objects2_Start_speed (enum) : 0, 0.28, 0.56, 0.83, 1.11, 1.39, 1.6, 1.94, 2.22, 2.5, 2.8, null
16 Pedestrians_Objects2_Offset_s (enum) : -9, -10, -11, -12, -13, -14, -15, -16, -17, -18, -19, null
```

Inspection of Individual Crash Scenario

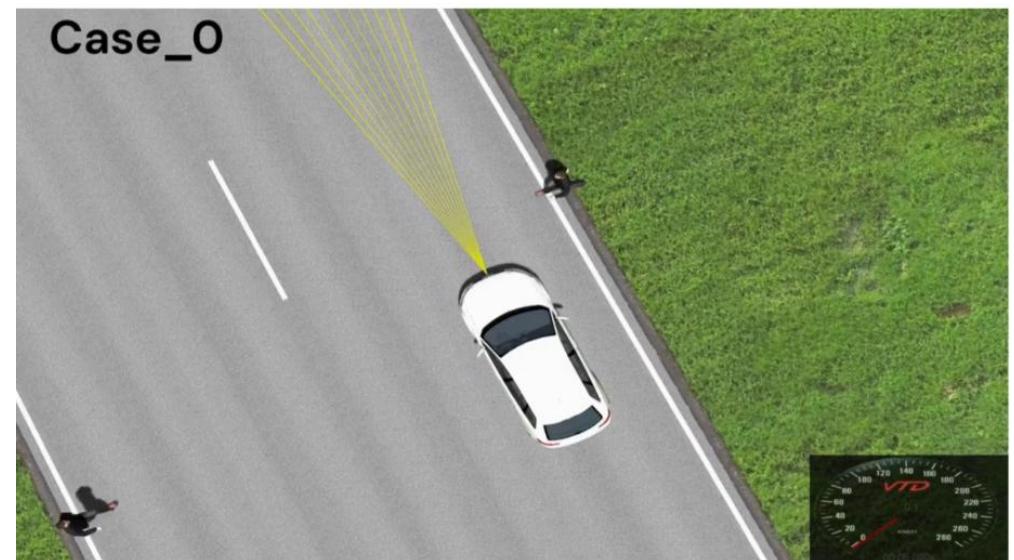
One Factor at a Time (OFAT) Strategy

| EgoVehicle1_Offset_s | Pedestrains_Objects1_Sta | Pedestrains_Objects1_Off | Pedestrains_Objects2_Off | Vehicles_Players2_Offset | Offset | Oracle |
|----------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------|-----------|
| rt_speed | set_s | set_t | set_s | _t | | |
| 1 | 1.39 | 10 | 20 | -18 | 65 | crash |
| 0 | 1.39 | 10 | 20 | -18 | 65 | non-crash |
| 1 | 0 | 10 | 20 | -18 | 65 | non-crash |
| 1 | 1.39 | 3 | 20 | -18 | 65 | non-crash |
| 1 | 1.39 | 10 | 10 | -18 | 65 | non-crash |
| 1 | 1.39 | 10 | 20 | -9 | 65 | non-crash |
| 1 | 1.39 | 10 | 20 | -18 | 15 | non-crash |

Case_0



Case 0



Why Rigorous Security Testing of Software and Systems?

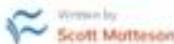


Topic — Software



Report: Software failure caused \$1.7 trillion in financial losses in 2017

Published January 26, 2018



Software testing company Tricentis found that retail and consumer technology were the areas most affected, while software failures in public service and healthcare were down from the previous year.

Lack of IT-Security Specialists

- 4M Cybersecurity specialists missing worldwide [Cyber WorkForce Study 2023]
- In EU IT-Security employees grow by 7.2% and their lack is 9.7%

The Problem of Malicious Hardware Logic Detection

Cryptographic Trojans as Instances of Malicious Hardware

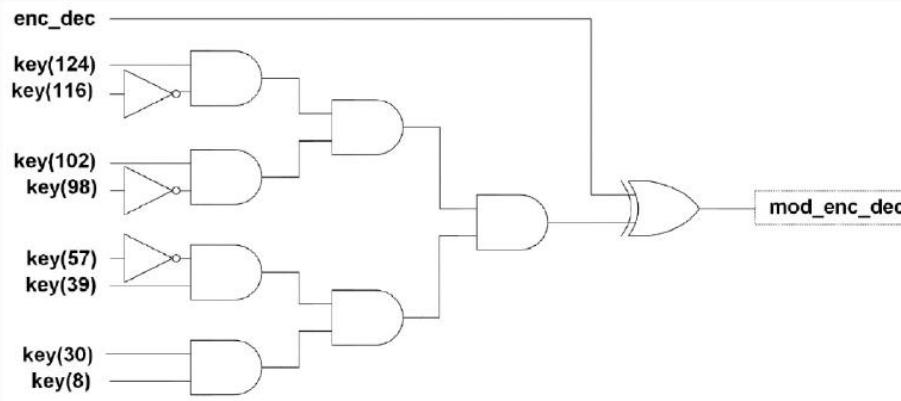
- **Scenario:** Trojans reside inside **cryptographic circuits** that perform encryption and decryption in FPGA technologies
 - **Examples:** Block ciphers (AES), Stream Ciphers (Mosquito)
- **Problem:** Hardware Trojan horse (HTH) **detection**



Combinational Trojans

A Combinational Trojan in AES-128

- Activates when a **specific combination** of key bits appears



- When **all** monitored inputs are "1", the Trojan payload part (just one XOR gate!) is activated
- Trojan reverses the mode of operation (DoS attack)

Trojan Design the Latest Years

Allegedly Reported Cases of Hardware Trojans

- **2007:** Syrian radar failed to warn of an incoming air strike (a backdoor built into the system's chips was rumored to be responsible)
- **2012:** Counterfeit semiconductor chips on the rise (commercial, military grade), rumored to be traced back to China

How Large are Today's Hardware Trojan Horses?

Recent study added fewer than 1,000 transistors to the 1.8 million already on the chip (a small backdoor circuit that gave access to privileged regions of chip memory)

- **Increased Awareness:** DARPA Report, 2011, US House of Representatives, 2012, US DoD Trusted Foundry Program 2012

Exciting (Triggering) Hardware Trojan Horses

Threat Model

- The attacker can control the **key** or the **plaintext** input and can observe the **ciphertext** output
- The attacker combines only a **few** signals for the activation

Input Model for Symmetric Ciphers

- **Activating Sequence:** Trojan **monitors** $k \ll 128$ key bits of AES-128
- **Attack vectors:** Model **activating** sequences of the Trojan (**black-box** testing); 128 **binary** parameters for AES-128
- **Input space:** $2^{128} = 3.4 \times 10^{38}$ for 128 bits key
 - **Exhaustive testing** becomes **intractable**

The Problem of Generating a Test Set

The Problem for Testing of Hardware Trojans

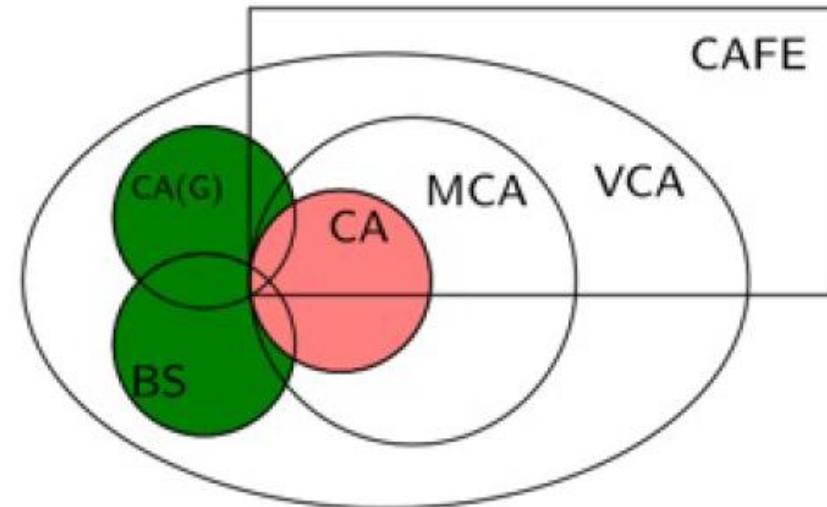
- How to efficiently test **all** possible k -bit input vectors for Trojan activation?

The General (Combinatorial) Test Generation Problem

Let n and $k \ll n$ parameters of a SUT. Construct sets of test vectors of **minimal** size that cover **all** possible k -subspaces

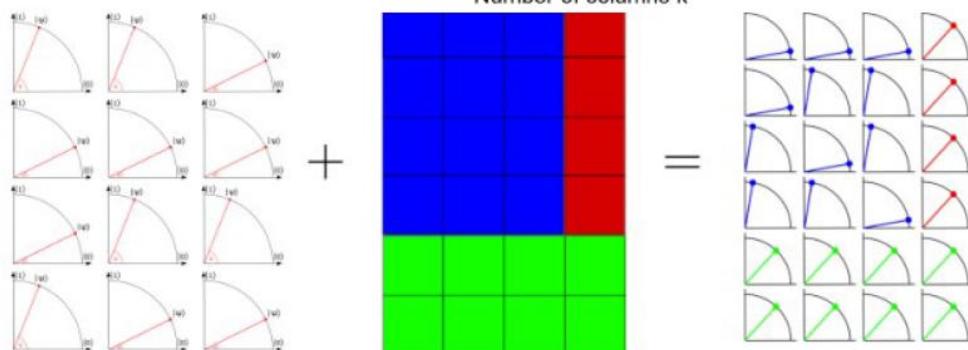
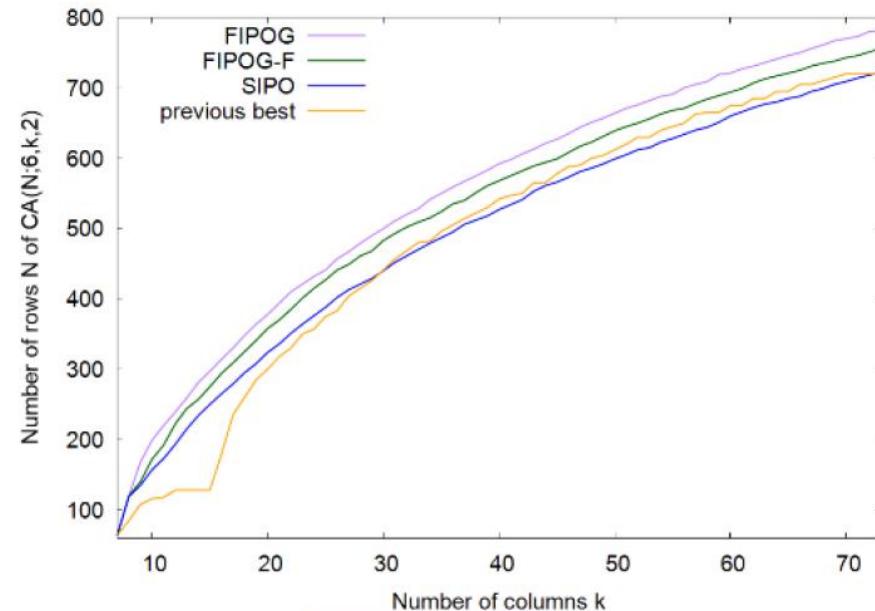
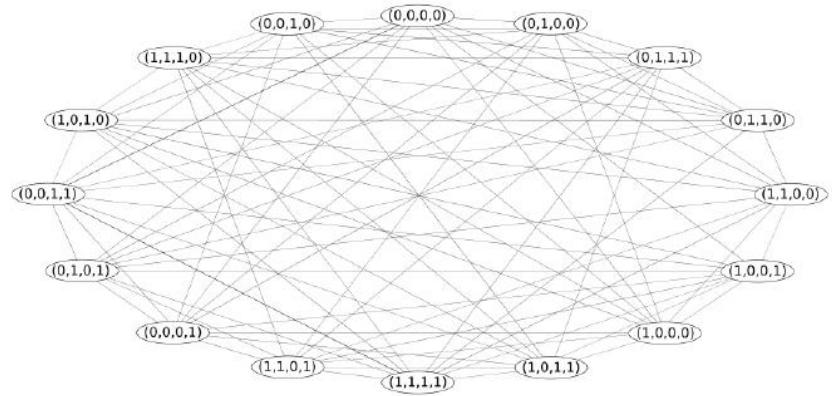
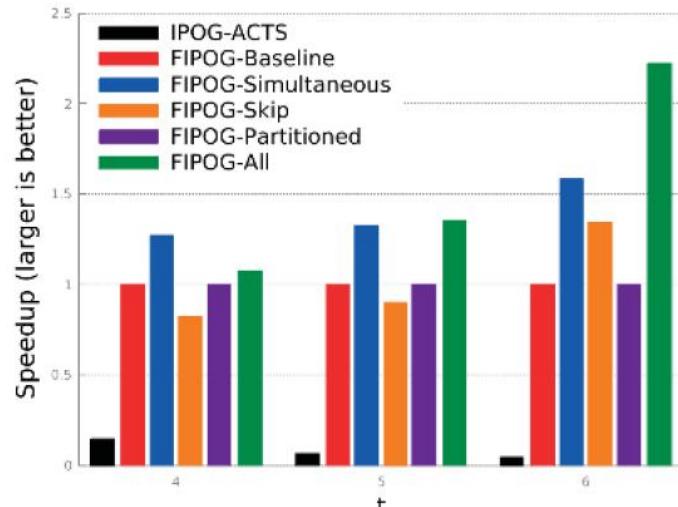
- **Equivalent** to finding a $CA(N; t, k, v)$ with **minimum** number of rows (also called the t -way covering problem)!
- The t -way covering problem is a **hard** **combinatorial optimization** problem studied for centuries

Complexity Classification of Problems of CAs



| Classes of Covering Arrays | Decide Existence | Decide Size | Determine Size | Generation |
|----------------------------|------------------|-------------|----------------|------------|
| optimal CA _{2,2} | P | P | P | P |
| optimal CA _{t,v} | P | NP | ??? | ??? |
| optimal MCA _{t,v} | P | NP | ??? | ??? |
| optimal BS _d | P | NP-complete | NP-hard | NP-hard |
| optimal VCA _{τ,2} | P | NP-complete | NP-hard | NP-hard |
| optimal VCA _{τ,v} | P | NP | ??? | ??? |
| optimal VCA _τ | P | NP-hard | NP-hard | NP-hard |
| optimal CA(G) | P | NP-complete | NP-hard | NP-hard |
| CAFE | NP-complete | NP-hard | NP-hard | NP-hard |

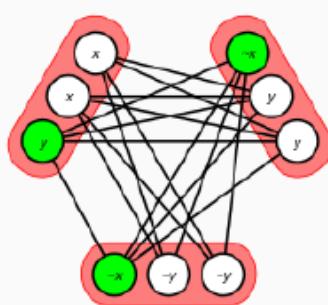
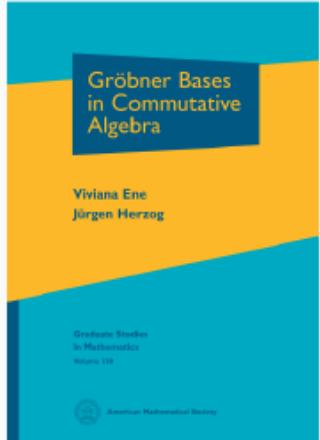
How to Construct CAs: Greedy, Neural, Learning and Quantum Computing Algorithms



From top left clockwise: (i) speedup of efficient IPO algorithm; (ii) new optimized CAs from metaheuristic-enhanced IPO algorithm; (iii) Artificial neural networks for CA generation; (iv) quantum-inspired IPO algorithm.

How to Construct CAs: Algebraic and SAT Methods

$$\begin{pmatrix} 0 & 0 & x_1 \\ 1 & 0 & x_2 \\ 0 & 1 & x_3 \\ 1 & 1 & x_4 \end{pmatrix} \cdot \begin{pmatrix} a \\ 0 \\ b \end{pmatrix} = \begin{pmatrix} bx_1 \\ a + x_2b \\ bx_3 \\ a + bx_4 \end{pmatrix}$$



$$x_1(x_1 - 1) = 0$$

$$x_2(x_2 - 1) = 0$$

$$x_3(x_3 - 1) = 0$$

$$x_4(x_4 - 1) = 0$$

$$(bx_1 - b)(bx_2 + a - b)(bx_3 - b)(bx_4 + a - b) = 0$$

$$(bx_1 - b)(bx_2 - b)(bx_3 + a - b)(bx_4 + a - b) = 0$$

$$(bx_1 - a - b)(bx_2 - b)(bx_3 - a - b)(bx_4 - b) = 0$$

$$(bx_1 - a - b)(bx_2 - a - b)(bx_3 - b)(bx_4 - b) = 0$$

$$b^2 x_1 (bx_2 + a) x_3 (bx_4 + a) = 0$$

$$b^2 x_1 x_2 (bx_3 + a) (bx_4 + a) = 0$$

How to Construct CAs: CPHF to CAs

Definition: Covering Perfect Hash Family

Given a prime power v , a *covering perfect hash family* (CPHF)

$CPHF(n; k, v, t)$ is an $n \times k$ array of *permutation vectors*, where all selections of t columns contain at least one row where the t -tuple of permutation vectors is covering.

110 111 121 100 122 112 101 012 120 011
102 121 111 011 122 101 001 012 010 112

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 2 | 2 | 1 | 2 | 0 | 0 | 1 |
| 0 | 2 | 2 | 0 | 1 | 1 | 2 | 1 | 0 | 1 | 2 |
| 1 | 1 | 2 | 0 | 2 | 1 | 0 | 1 | 0 | 1 | 2 |
| 1 | 2 | 0 | 0 | 1 | 0 | 1 | 2 | 2 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 2 | 2 | 2 | 0 | 2 | 0 |
| 2 | 2 | 1 | 0 | 1 | 2 | 0 | 0 | 2 | 1 | 1 |
| 2 | 0 | 2 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 2 | 1 | 0 | 0 | 2 | 0 | 2 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 2 | 2 | 1 | 0 | 0 | 2 | 2 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 2 | 2 | 0 | 1 | 1 | 2 | 0 |
| 2 | 2 | 0 | 1 | 0 | 2 | 1 | 1 | 1 | 0 | 1 |
| 2 | 0 | 1 | 1 | 2 | 1 | 2 | 0 | 0 | 2 | 0 |
| 2 | 1 | 2 | 1 | 1 | 0 | 0 | 2 | 0 | 0 | 0 |
| 0 | 0 | 2 | 1 | 2 | 0 | 1 | 2 | 2 | 2 | 0 |
| 0 | 1 | 0 | 1 | 1 | 2 | 2 | 1 | 2 | 0 | 0 |
| 0 | 2 | 1 | 1 | 0 | 1 | 0 | 0 | 2 | 1 | 0 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 0 | 2 | 0 | 0 |
| 2 | 0 | 0 | 2 | 1 | 1 | 0 | 2 | 2 | 1 | 2 |
| 2 | 1 | 1 | 2 | 0 | 0 | 1 | 1 | 2 | 2 | 0 |
| 0 | 0 | 1 | 2 | 1 | 0 | 2 | 1 | 1 | 1 | 1 |
| 0 | 1 | 2 | 2 | 0 | 2 | 0 | 0 | 0 | 1 | 2 |
| 0 | 2 | 0 | 2 | 2 | 1 | 1 | 2 | 1 | 0 | 0 |
| 1 | 1 | 0 | 2 | 0 | 1 | 2 | 2 | 0 | 2 | 0 |
| 1 | 2 | 1 | 2 | 2 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 2 | 2 | 1 | 2 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 2 | 0 | 2 | 1 | 2 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 1 | 2 | 1 | 1 | 2 | 0 | 2 | 0 |
| 1 | 2 | 2 | 2 | 1 | 2 | 1 | 2 | 1 | 0 | 1 |
| 0 | 2 | 1 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 1 |
| 2 | 0 | 2 | 1 | 2 | 1 | 0 | 2 | 2 | 1 | 2 |
| 0 | 2 | 0 | 2 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 2 | 1 | 1 | 1 | 0 | 0 | 1 |
| 2 | 2 | 0 | 0 | 1 | 2 | 1 | 0 | 2 | 2 | 1 |
| 0 | 1 | 2 | 2 | 1 | 0 | 1 | 1 | 2 | 1 | 2 |
| 2 | 2 | 0 | 0 | 1 | 2 | 1 | 0 | 2 | 2 | 1 |
| 1 | 0 | 1 | 2 | 2 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 2 | 0 | 2 | 1 | 2 | 0 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 1 | 2 | 2 | 0 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 2 | 0 | 0 |
| 0 | 2 | 0 | 1 | 2 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 2 | 0 | 1 | 2 | 1 | 0 | 0 | 1 | 2 |
| 0 | 2 | 0 | 1 | 1 | 2 | 0 | 1 | 1 | 2 | 0 |
| 1 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 2 | 0 | 0 | 1 | 2 | 2 | 1 | 1 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | 2 | 0 | 2 | 0 | 2 | 0 | 1 | 1 |
| 1 | 1 | 2 | 0 | 2 | 0 | 1 | 1 | 1 | 2 | 0 |
| 2 | 1 | 0 | 1 | 2 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 2 | 1 | 2 | 0 | 0 | 1 | 0 | 1 | 0 | 2 |
| 0 | 0 | 2 | 0 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 2 | 0 | 1 | | | | | | | | |

How to Classify (partially) CAs: Balanced Covering Arrays

(λ, \mathbf{y}) -balanced Covering Arrays

An array is called (λ, \mathbf{y}) -balanced covering arrays for $\lambda = (\lambda_1, \dots, \lambda_t)$ and $\mathbf{y} = (y_1, \dots, y_t)$, iff for all $i = 1, \dots, t$ each i -way interaction appears at least λ_i times and at most y_i times.

Class of all (λ, \mathbf{y}) -balanced CAs with N rows, k columns over v -ary alphabet:

$$\mathbb{CA}_{\lambda}^{\mathbf{y}}(N; t, k, v) = \bigcap_{i \in \{1, \dots, t\}} \mathbb{CA}_{\lambda_i}(N; i, k, v) \cap \mathbb{PA}_{y_i}(N; i, k, v)$$

There exist exactly two non-equivalent CA(15; 3, 12, 2)'s:

$$A_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

A_1 is a $\mathbb{CA}_{(7,3,1)}^{(8,5,4)}(15; 3, 12, 2)$

A_2 is a $\mathbb{CA}_{(7,2,1)}^{(8,6,4)}(15; 3, 12, 2)$

Optimized Test Sets for CAs for Trojan Detection

- Comparison of test set sizes using the **constant weight vectors** (CWV) procedure (Tang and Woo, 1983) and the **CA** generation methods

| n | t | Lesperance et al. (2015) | CWV | ours |
|-----|-----|--------------------------|--------------|---------|
| 128 | 2 | 2^7 | 129 | 11 |
| 128 | 3 | - | 256 | 37 |
| 128 | 4 | 2^{13} | 8, 256 | 112 |
| 128 | 5 | - | 16, 256 | 252 |
| 128 | 6 | - | 349, 504 | 720 |
| 128 | 7 | - | 682, 752 | 2, 462 |
| 128 | 8 | 2^{23} | 11, 009, 376 | 17, 544 |

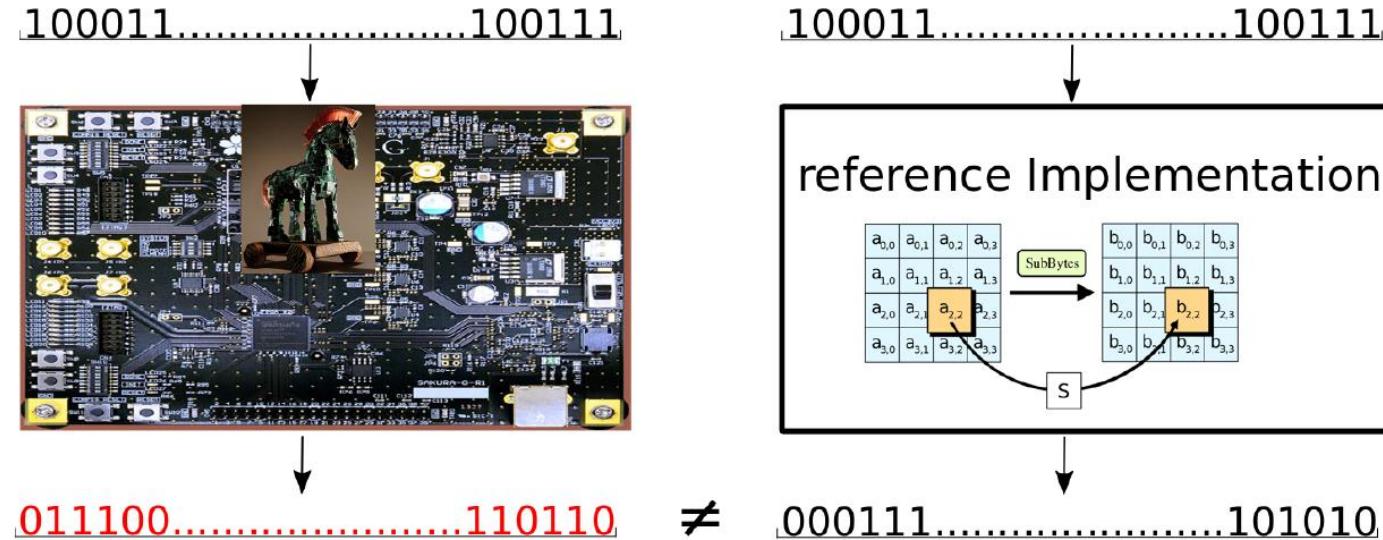
Employed CA Generation Methods:

- Simulated Annealing (SA) algorithms
- CAs from cyclotomy, constructions via Hash families

Case Study for Exciting Hardware Trojan Horses

Test Execution

- Hardware implementation: AES symmetric [encryption](#) algorithm over the Verilog-HDL model with the Sakura-G FPGA board



Oracle

Compare the output with a [Trojan-free](#) design of AES-128 (e.g. software implementation)

Test Results for Detecting Hardware Trojan Horses

- Test suite **strength** (t) vs. Trojan **length** (k)

| t | Suite size | Number of activations | | |
|-----|------------|-----------------------|---------|---------|
| | | $k = 2$ | $k = 4$ | $k = 8$ |
| 2 | 11 | 5 | 3 | 0 |
| 3 | 37 | 12 | 4 | 0 |
| 4 | 112 | 32 | 7 | 1 |
| 5 | 252 | 62 | 14 | 1 |
| 6 | 720 | 307 | 73 | 6 |
| 7 | 2462 | 615 | 153 | 10 |
| 8 | 17544 | 4246 | 1294 | 178 |

Our Evaluation Results at a Glance

- There are about 366 ***trillion*** possible **combinations** for the Trojan activation;
- The whole space is **covered** with less than 18 ***thousands*** vectors
- .. and these vectors **activate** the Trojan ***hundreds*** of times

Combinatorial Security Testing (CST) vulnerabilities found via interaction patterns

Proven-method: Rigorous testing for security

- Complex web applications:
 - XSS, SQL-i vulnerabilities
- Next generation protocol testing:
 - Parsing/input validation errors in TLS/BLE
- Intelligent and autonomous systems
 - Faults in autonomous driving functions
- Hardware Trojan Horse detection
 - Detection of Combinational Trojans

Views: desktop mobile print

STANDARDS PARTICIPATE MEMBERSHIP ABOUT W3C

W3C » Participate » Mail, News, Blogs, Podcasts, and... » W3C Blog

MAIL, NEWS, BLOGS, PODCASTS, AND TUTORIALS

News

Weekly Newsletter

W3C Blog

Mailing Lists

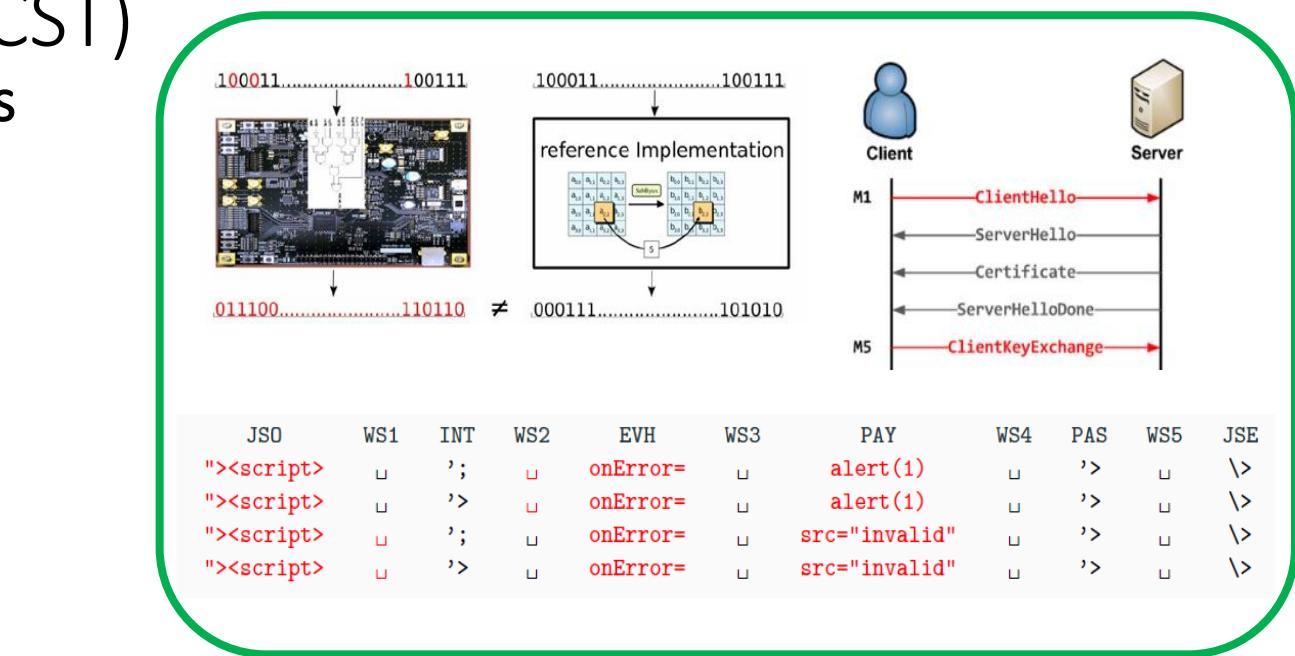
Podcasts and Video

Tutorials and Courses

R XSS SECURITY AUDIT RESULTS

11 December 2014 by Ted Guild | Posted in: Bugs Life, Security, W3C Life

W3C recently submitted to a Web Application Penetration Test. It was conducted by researchers and testers of [SBA Research](#) within the context of Mobsitet research project and specifically targeted Reflected-Cross-Site-Scripting vulnerabilities using combinatorial testing methodologies. SBA Research approached W3C since the size of our website and the nature of our organization made for an interesting test subject. W3C seeks to continually improve its security and has submitted to penetration tests in the past, conducted its own audits and welcomes community reports on its open collaborative infrastructure. A RXSS vulnerability was found in W3C's online tidy service and corrected. Anyone running their own instance of this service is encouraged to upgrade.



CVE-ID

CVE-2015-4631 [Learn more at National Vulnerability Database \(NVD\)](#)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

Multiple cross-site scripting (XSS) vulnerabilities in Koha 3.14.x before 3.14.16, 3.16.x before 3.16.12, 3.18.x before 3.18.08, and authorities/authorities/home in the 'callnumber' parameter to acm/authorities_id, the 'id' authorities_id or 'id' in field in acm/authorities_id.

CVE-ID

CVE-2018-19202 [Learn more at National Vulnerability Database \(NVD\)](#)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

A reflected XSS vulnerability in index.php in MyBB 1.8x through 1.8.19 allows remote attackers to inject JavaScript via the 'upsetting[bburl]' parameter.

References

Note: References

- CONFIRM
- CONFIRM

CVE-ID

CVE-2018-19201 [Learn more at National Vulnerability Database \(NVD\)](#)
• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

A reflected XSS vulnerability in the ModCP Profile Editor in MyBB before 1.8.20 allows remote attackers to inject JavaScript via the 'username' parameter.

References

Note: References are provided for the convenience of the reader to help distinguish between vulnerabilities. The list is not intended to be complete.

- CONFIRM: <https://blog.mybb.com/2018/02/27/mybb-1-8-20-released-security-maintenance-release/>
- CONFIRM: <https://github.com/mybb/mybb/blob/feature/SECURITY/mechanical-details-of-known-issues>

Resilience of Complex Systems

Combinatorics beyond Experimental Design

Prevention and Reduction of Disaster Risk

Several Scientific Issues need to be addressed:

- Extreme weather events
- Impact of cascading effects by natural hazards
- Multi-hazard risk reduction

Existing Technology could and need to be further improved:

- Risk assessment
- Simulation tools
- Precise prediction, forecast and early warning systems

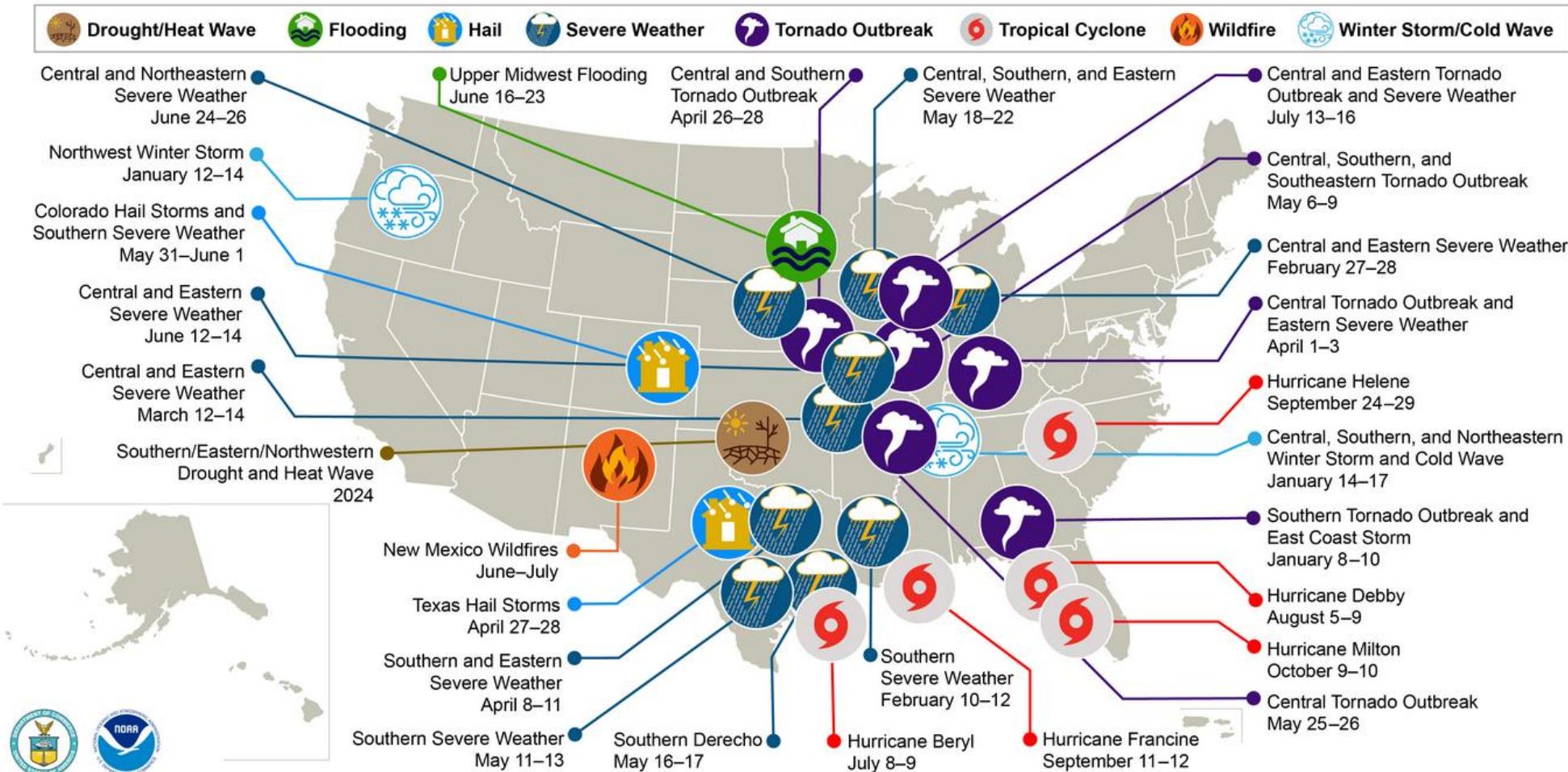
Science, Technology & Innovation (STI) Challenge:

An integrated disaster management system with monitoring, warning and simulation, **response plans** is lacking



Billion-Dollar Weather and Climate Disasters

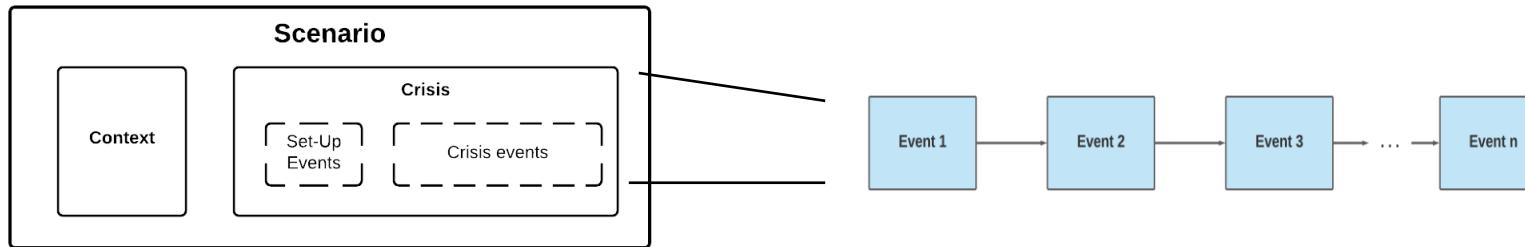
U.S. 2024 Billion-Dollar Weather and Climate Disasters



This map denotes the approximate location for each of the 27 separate billion-dollar weather and climate disasters that impacted the United States in 2024.

Crisis Scenarios in Disaster Research

- Scenarios can be modeled as sequences of events

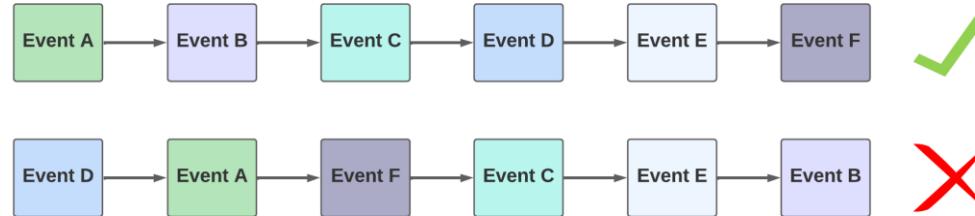


Structure of a crisis - scenario

- Disaster relief strategies as “Disaster Under Test” (DUT)

| Sequences | | | | | |
|-----------|---|---|---|---|---|
| A | B | C | D | E | F |
| D | A | F | C | E | B |
| B | D | E | A | C | F |
| E | D | B | F | C | A |
| E | C | A | F | B | D |
| C | B | F | E | A | D |
| F | A | E | D | B | C |
| F | C | D | B | A | E |

Testing different sequences of events



Combinatorial structures for disaster scenarios

Example for a combinatorial structure from discrete mathematics: *Sequence Covering Array*

Definition:

Let S be a non-empty set with $|S| = s \in \mathbb{N}^*$ and $N, t \in \mathbb{N}^*$ with $1 \leq t \leq s$.

A **sequence covering array (SCA)** of strength t , $SCA(N, S, t)$ is

- an $N \times s$ matrix,
- with entries from the finite set S ,
- such that every t -way permutation of symbols from S occurs in at least one row (not necessarily adjacent),
- each row is a permutation of the s symbols.

Example:

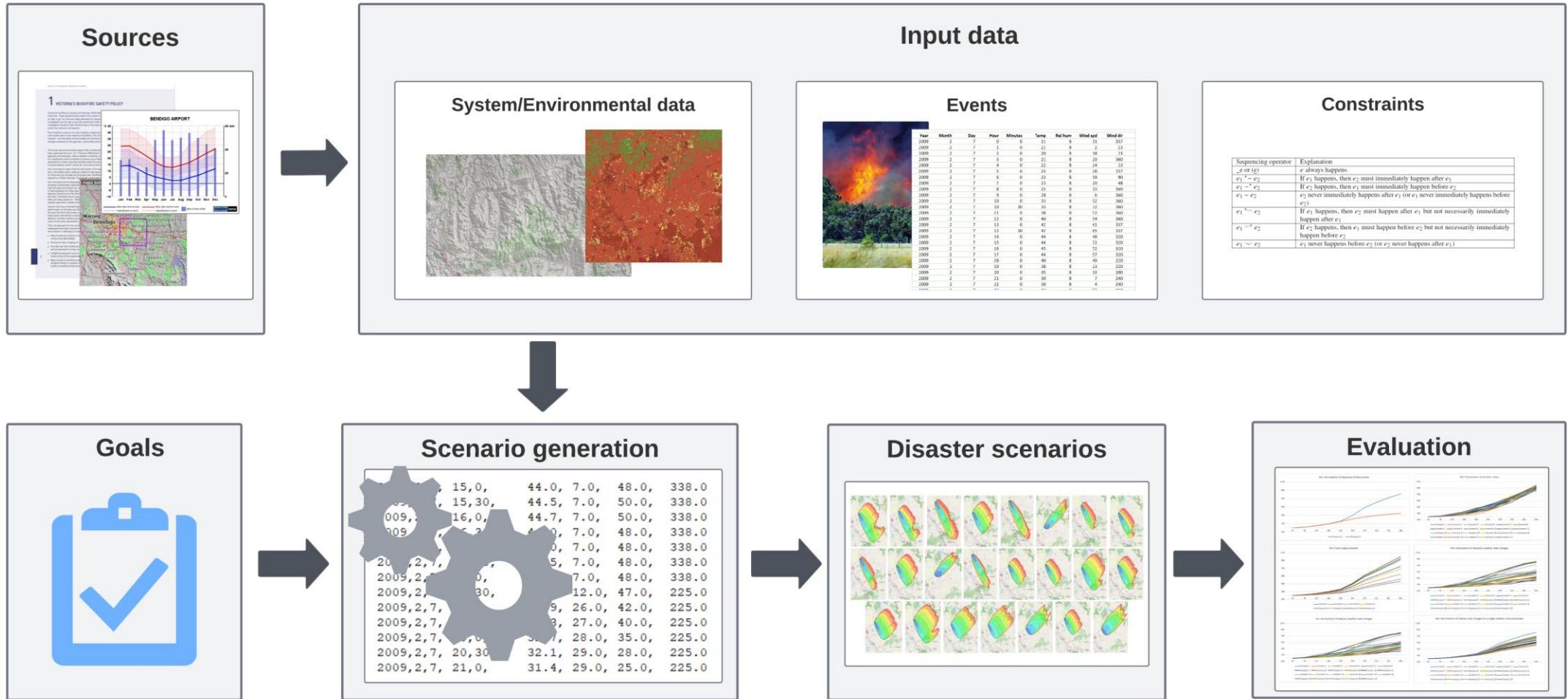
| | | | | |
|-------------------|---|---|---|---|
| Test ₁ | A | D | B | C |
| Test ₂ | B | A | C | D |
| Test ₃ | B | D | C | A |
| Test ₄ | C | A | B | D |
| Test ₅ | C | D | B | A |
| Test ₆ | D | A | C | B |

Sequences covered:

$\{ADB\}, \{ADC\}, \{ABC\}, \{DBC\}$
 $\{BAC\}, \{BAD\}, \{BCD\}, \{ACD\}$
 $\{BDC\}, \{BDA\}, \{BCA\}, \{DCA\}$
 $\{CAB\}, \{CAD\}, \{CBD\}, \{ABD\}$
 $\{CDB\}, \{CDA\}, \{CBA\}, \{DBA\}$
 $\{DAC\}, \{DAB\}, \{DCB\}, \{ACB\}$

SCA of strength 3 of a symbol set $S = \{A, B, C, D\}$ of cardinality four

Combinatorial Fire Simulation Research Prototype



Scenario Generation by Permutation of Weather States

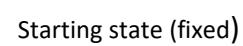
Original sequence of weather states

| Progress of scenario | 44.0 | 0 | 7.0 | 0 | 48.0 | 0 | 338.0 |
|----------------------|------|---|------|---|------|---|-------|
| | 44.5 | 0 | 7.0 | 0 | 50.0 | 0 | 338.0 |
| | 44.7 | 0 | 7.0 | 0 | 50.0 | 0 | 338.0 |
| | 44.0 | 0 | 7.0 | 0 | 48.0 | 0 | 338.0 |
| | 43.0 | 0 | 7.0 | 0 | 48.0 | 0 | 338.0 |
| | 42.5 | 0 | 7.0 | 0 | 48.0 | 0 | 338.0 |
| | 42.0 | 0 | 7.0 | 0 | 48.0 | 0 | 338.0 |
| | 41.1 | 0 | 12.0 | 0 | 47.0 | 0 | 225.0 |
| | 33.9 | 0 | 26.0 | 0 | 42.0 | 0 | 225.0 |
| | 33.3 | 0 | 27.0 | 0 | 40.0 | 0 | 225.0 |
| | 32.7 | 0 | 28.0 | 0 | 35.0 | 0 | 225.0 |
| | 32.1 | 0 | 29.0 | 0 | 28.0 | 0 | 225.0 |
| | 31.4 | 0 | 29.0 | 0 | 25.0 | 0 | 225.0 |

Sequence Covering Array (strength 3, cardinality 12)

New sequence of weather states

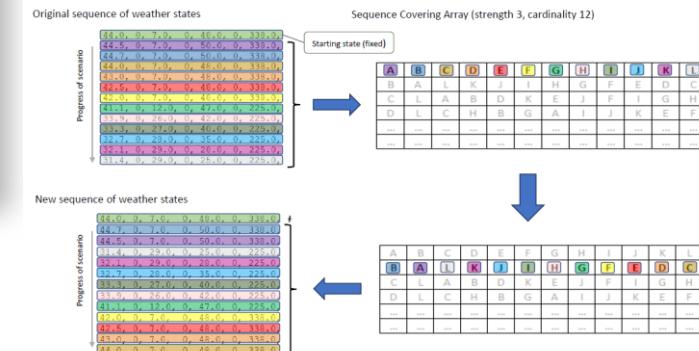
| Progress of scenario | 44.0, 0, 7.0, 0, 48.0, 0, 338.0 |
|----------------------|----------------------------------|
| | 44.7, 0, 7.0, 0, 50.0, 0, 338.0 |
| | 44.5, 0, 7.0, 0, 50.0, 0, 338.0 |
| | 31.4, 0, 29.0, 0, 25.0, 0, 225.0 |
| | 32.1, 0, 29.0, 0, 28.0, 0, 225.0 |
| | 32.7, 0, 28.0, 0, 35.0, 0, 225.0 |
| | 33.3, 0, 27.0, 0, 40.0, 0, 225.0 |
| | 33.9, 0, 26.0, 0, 42.0, 0, 225.0 |
| | 41.1, 0, 12.0, 0, 47.0, 0, 225.0 |
| | 42.0, 0, 7.0, 0, 48.0, 0, 338.0 |
| | 42.5, 0, 7.0, 0, 48.0, 0, 338.0 |
| | 43.0, 0, 7.0, 0, 48.0, 0, 338.0 |
| | 44.0, 0, 7.0, 0, 48.0, 0, 338.0 |



Analysis, Modelling & Simulation of Complex Systems (Bushfires)

MATRIX

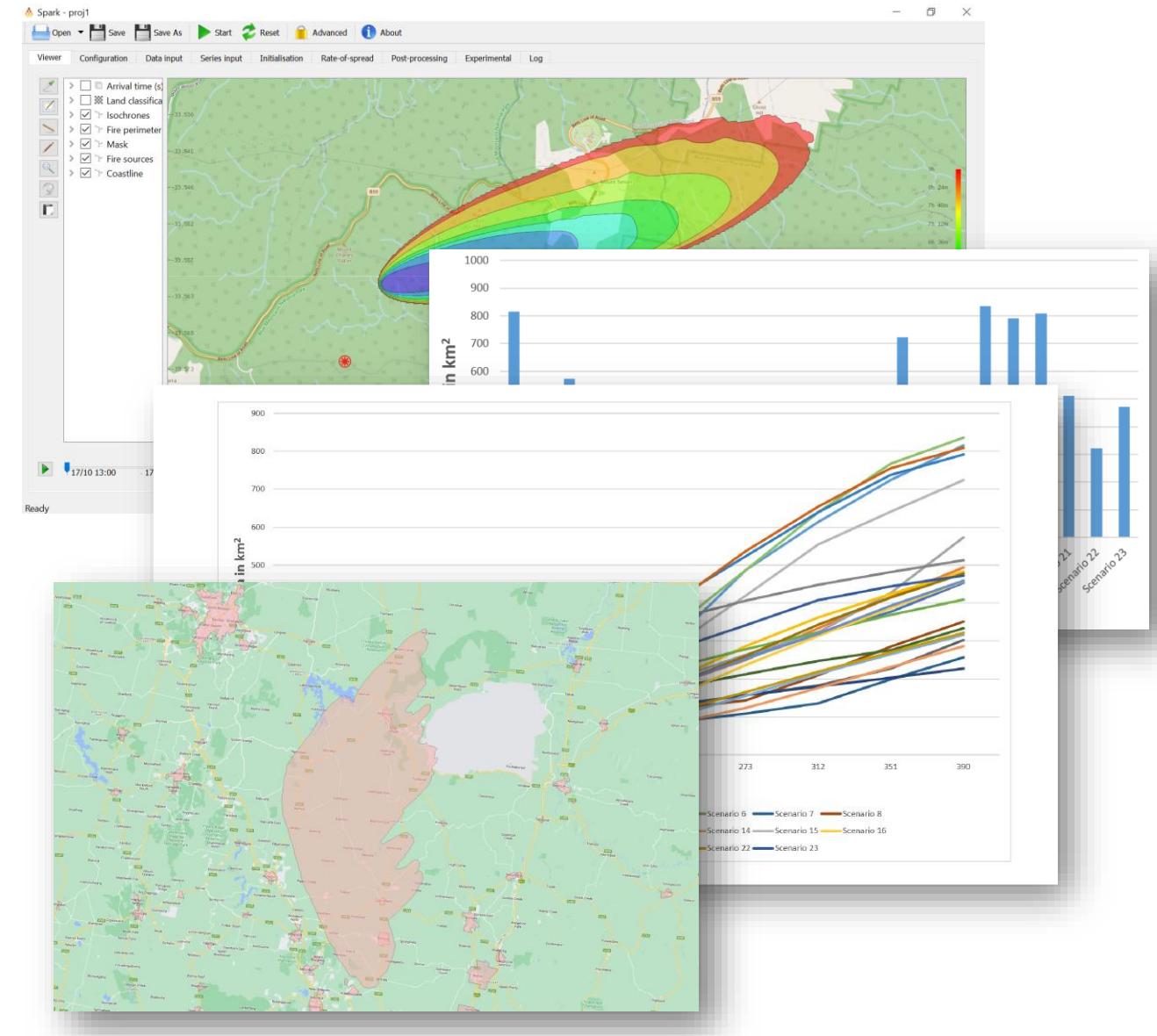
Comparison of Combinatorial Fire Scenarios



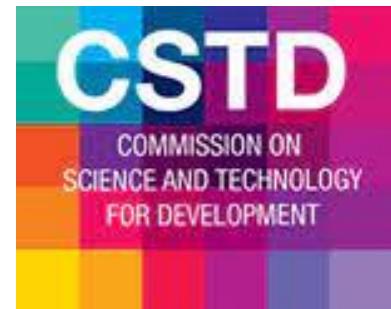
Evaluation

- Speed of fire spread,
- Total area affected,
- Towns/Infrastructure affected
- CSIRO SPARK

<https://research.csiro.au/spark/>



Integration of Combinatorial Design Scenarios into Policy Making



Activity Report

HARNESSING STI FOR DISASTER RISK REDUCTION WORKSHOP

29 February-01 March 2024

Crimson Hotel, Alabang, Muntinlupa City, Philippines

A Joint initiative of the

*Department of Science and Technology (DOST) of the Republic of the Philippines,
Department of State of the United States of America, and
the United Nations Conference on Trade and Development (UNCTAD)*

*Under the Philippines and United States of America membership in the
Commission on Science and Technology for Development*

Decision-making tool based on combinatorial methodology for best-use of resources or option for action intended for government actors and disaster risk managers. The model offers a decision tree following a combination and sequence of events that would trigger a response action or not.

Policy Recommendation Excerpt

Conclusion & Future Outlook

Complex Systems

1. Critical Software / Cyber-physical Systems
2. Natural Hazards / Disasters

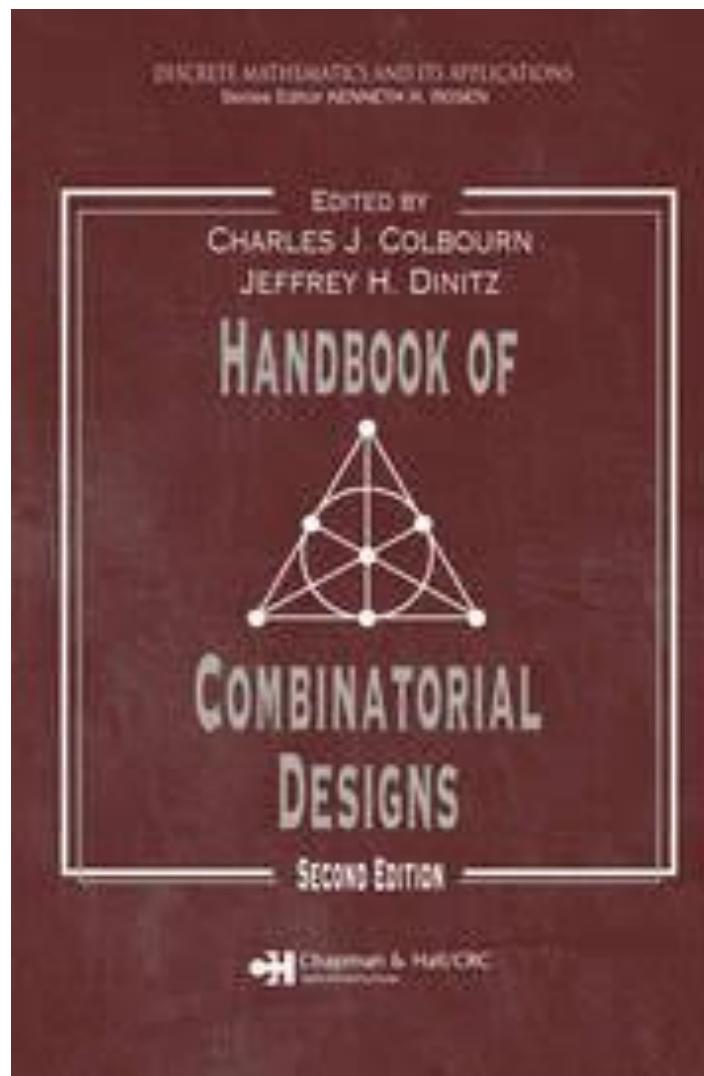
How are these connected?



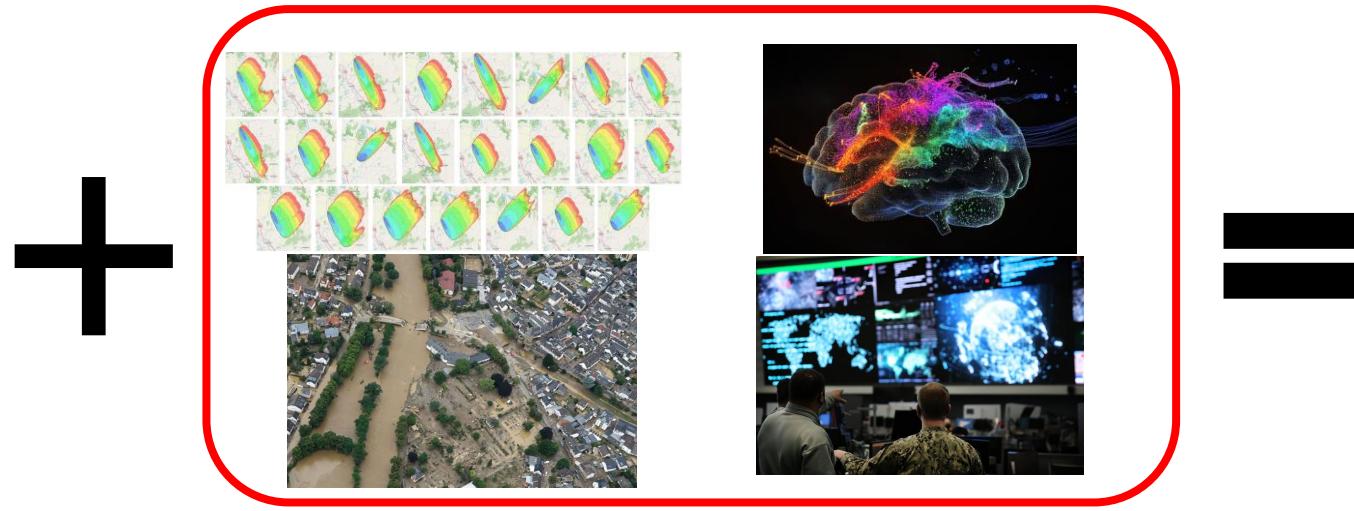
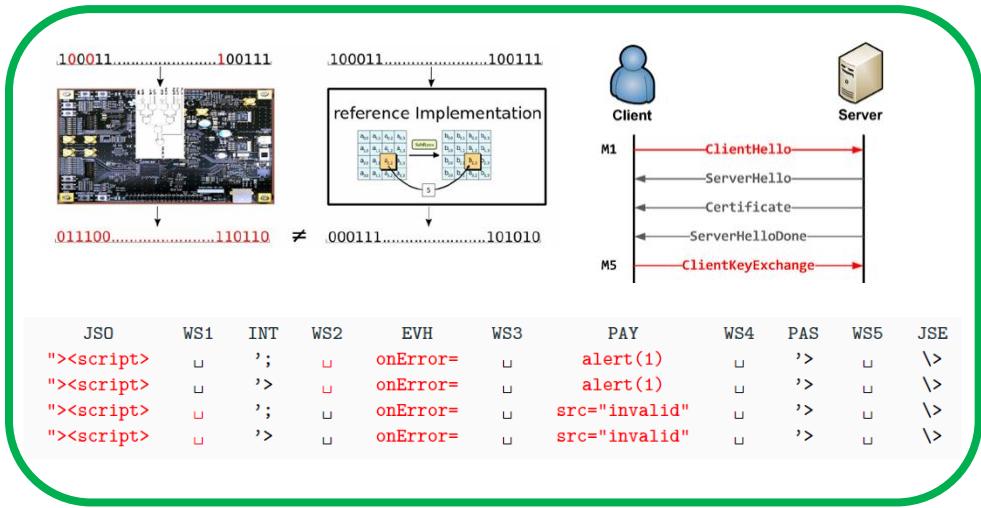
Interaction Patterns can be modelled after Mathematical Structures: Can their Identification enhance the Resilience of Complex Systems?

- **Cyber-physical Systems (Systems Under Test / Attack Paths)**
 - Software Systems Engineering: Software/System Failures → **Interaction Faults**
 - Web Security: Injection Vulnerabilities → **Complex (t-way) Attack Vectors**
 - Supply Chain Security: Hardware Trojans → **(Optimal) Covering Arrays**
- **Disaster Management (Natural / Technological Hazards)**
 - Compound and Cascading Effects of Hazards → **(Sequence) Covering Arrays**

How to Construct Interaction Patterns (at Large) ☺?



Scientific Vision for Combinatorial Designs in Systems Science



The concept of **interaction patterns** in **complex systems** can lead to an **integrated cyber security & resilience strategy**, encompassing diverse simulations, rigorous testing and resilient response plans, implemented across **participating stakeholders***.

(* <https://unctad.org/meeting/workshop-harnessing-science-technology-and-innovation-disaster-risk-reduction>

Thank you very much for your
attention!

Questions / Comments ?

dsimos@sba-research.org

