Complex, Resilient and Secure Systems: Where Combinatorial Methods and System Testing meet Cyber Security

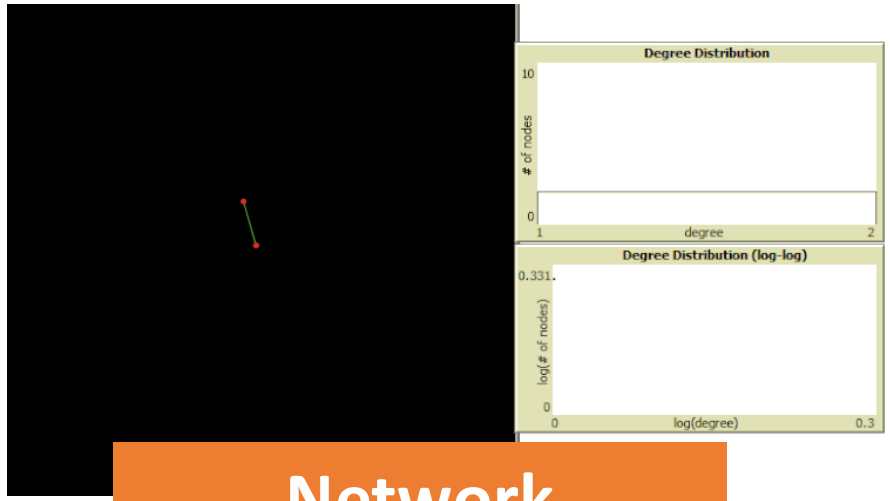
Dimitris Simos
Paris Lodron University of Salzburg (PLUS), Austria
Salzburg University of Applied Sciences (FH Salzburg), Austria
MATRIS Research Lab – PLUS & FH Salzburg, Austria

Keynote, November 05, 2025, Live Webinar

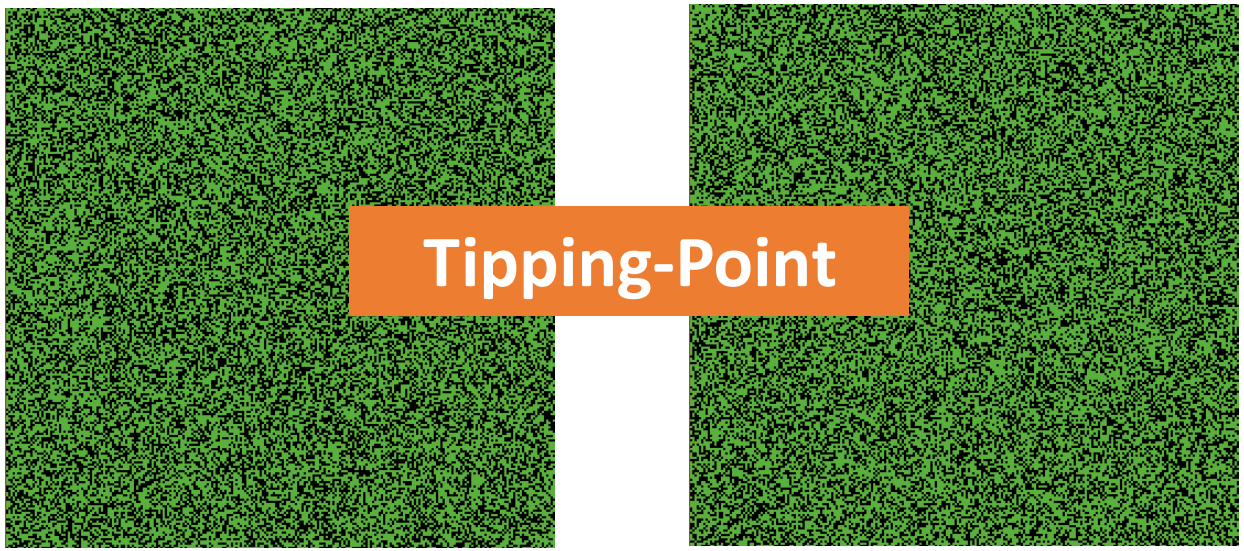
Outline of the Talk

- Introduction to Complex Systems
- Critical Software / Cyber-physical Systems
- Conclusion & Future Outlook

Analysis, Modelling & Simulation of Complex Systems



Network Evolution



Tipping-Point

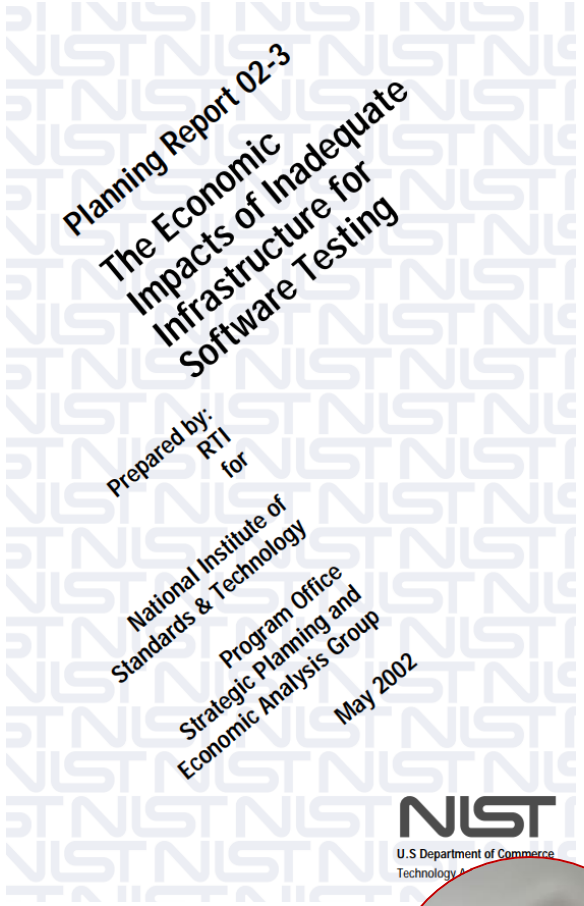
Pattern Identification

Patterns appear in Every Instance of a Complex System

- **Critical Software / Cyber-physical Systems (Systems Under Test)**
  - **Software Faults**
  - **System Failures**
  - **Security Vulnerabilities**

Motivation for Combinatorial Testing in Complex Systems

- **Economic Impact:** Software testing may consume up to half of the overall software development cost
  - **Combinatorial explosion:** Exhaustive search of input space increases time needed exponentially
  - Added level of complexity for system testing and security (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** and **resilience** problems as combinatorial problems and then use efficient methods to tackle them

Testing & Security of Critical Software and Systems

Combinatorics beyond Experimental Design

Example: A Large System for Testing

- Suppose we have a system with on-off switches
- 34 switches = 2^{34} = 1.7×10^{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)

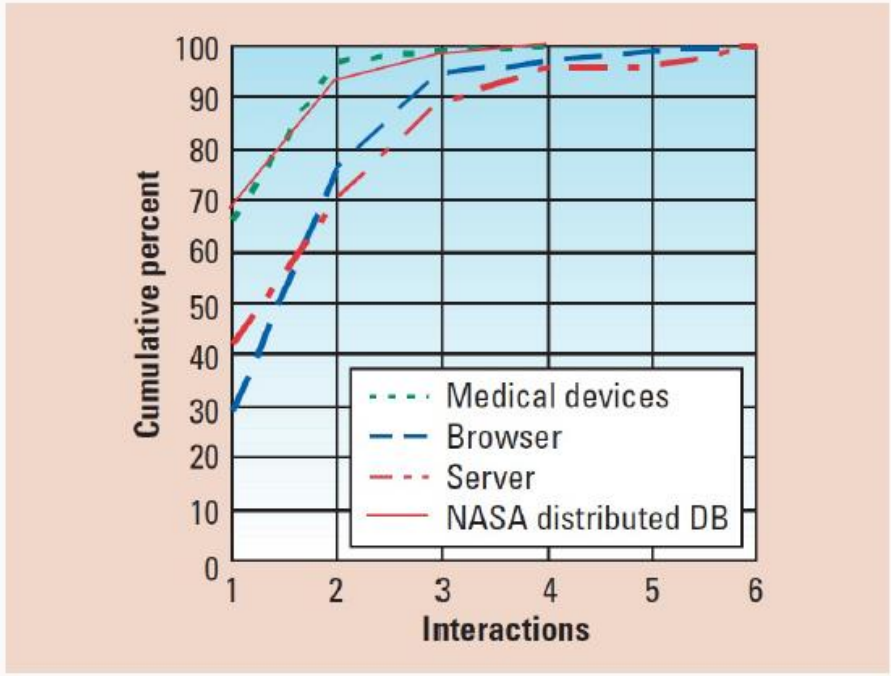
Covering Arrays $CA(N; t, k, v)$ **of Strength** t

- 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 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

TECH

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

Lora Kolodny
@IN/LORAKOLODNY/

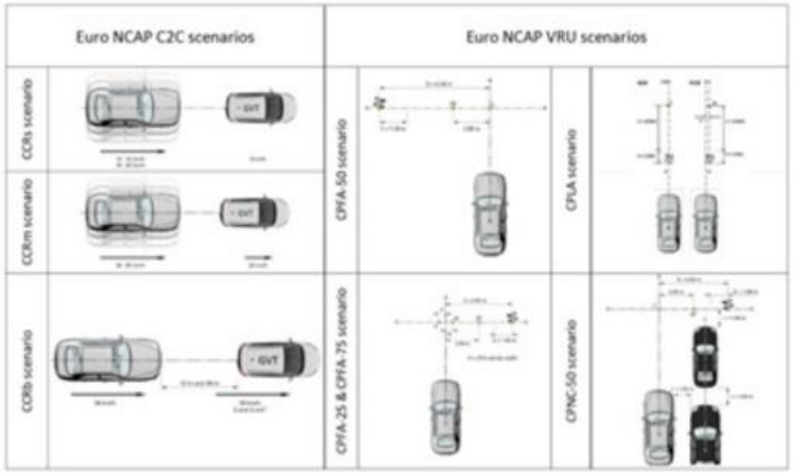
CNBC

SHARE

Virtual Driving Scenarios for testing the AEB function

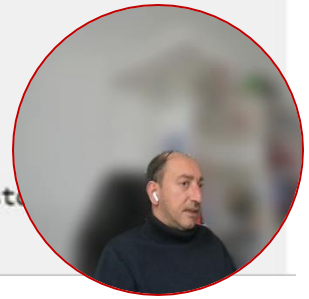
IPM for Driving Scenarios

- 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)

```
[Parameter]
NumberOfVehiclePlayer (enum) : 1, 2, 3
NumberOfPede (enum) : 1, 2, null
EgoVehicle1_Start_speed (enum) : 0, 1.388888889 ,2.777777778  ,4.166666667  ,5.555555556  ,6.944444444  ,8.333333333  ,9.722222222  , ...
EgoVehicle1_VehicleType (enum) : Audi_A3_2009_white, Audi_Q5_2008_red, Audi_S5_2009_black metallic
EgoVehicle1_Offset_s (enum) : -2,-1,0, 1, 2
EgoVehicle1_Offset_t (enum) : 0
EgoVehicle1_Rate (enum) : 5, 6, 7, 8, 9, 10
EgoVehicle1_Target_Speed (enum) : 14, 28, 42, 55
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
Pedestrians_Objects1_Offset_s (enum) : 3, 4, 5,6,7,8,9,10,11,12,13, null
Pedestrians_Objects1_Offset_t (enum) : 10,11,12,13,14,15,16,17,18,19,20, null
Pedestrians_Objects1_Rate (enum) : 0, 1, 2, 3, 4, 5, 6, 7, 8, null
Pedestrians_Objects1_Type (enum) : Adult, Child, Wheelchair, Animal, Ballon, Paper, Dog, Stone, Adult_w_Bicycle, Adult_w_child, custom1, cust
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
Pedestrians_Objects2_Offset_s (enum) : -9,-10,-11,-12,-13,-14,-15,-16,-17,-18,-19,null
```

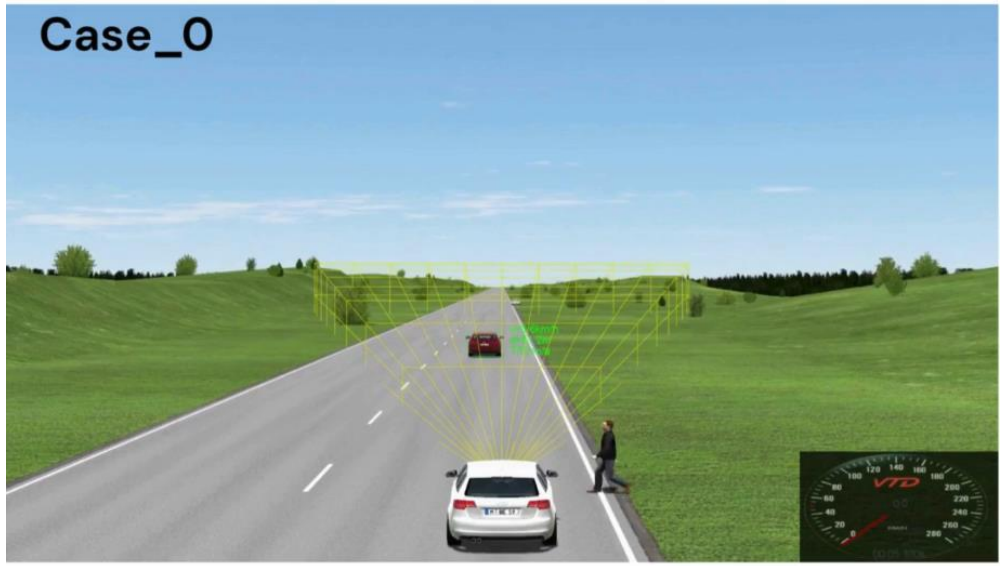


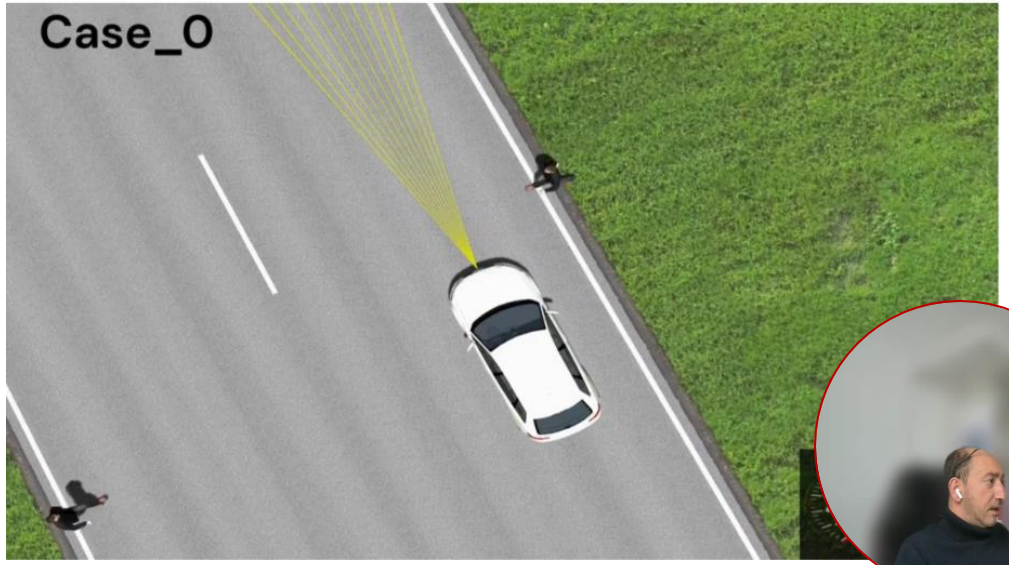
Inspection of Individual Crash Scenario

One Factor at a Time (OFAT) Strategy

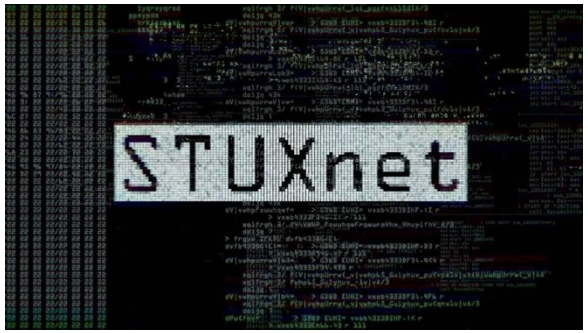
- Consider single crash scenario
- Change every parameter to all other values one by one
- $\Rightarrow \geq$6-way interaction identified
  - $(-,-,-,-,1,-,-,-,-,-,1.39,-,-,-,-,10,20,-,-,\text{-}18,-,-,-,-,-,-,-,-,-,-,65,-,-,-,-,-,-,-,-,-,-)$
  - This 6-way interaction appears in millions of tests (roughly 10^{26})

EgoVehicle1_Offset_s	Pedestrains_Objects1_Start_speed	Pedestrains_Objects1_Offset_s	Pedestrains_Objects1_Offset_t	Pedestrains_Objects2_Offset_s	Vehicles_Players2_Offset_t	Oracle
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





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 | Written by Scott Matteson

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

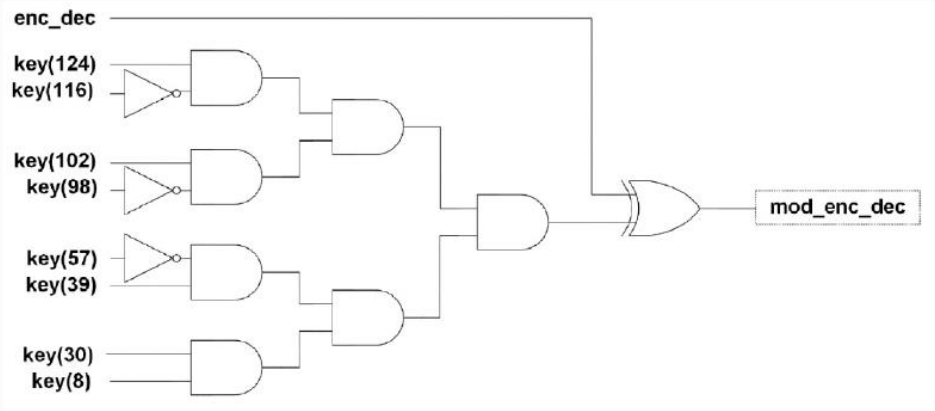
Cryptographic Trojans as Instances of Malicious Hardware

- **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 $k << 128$ 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

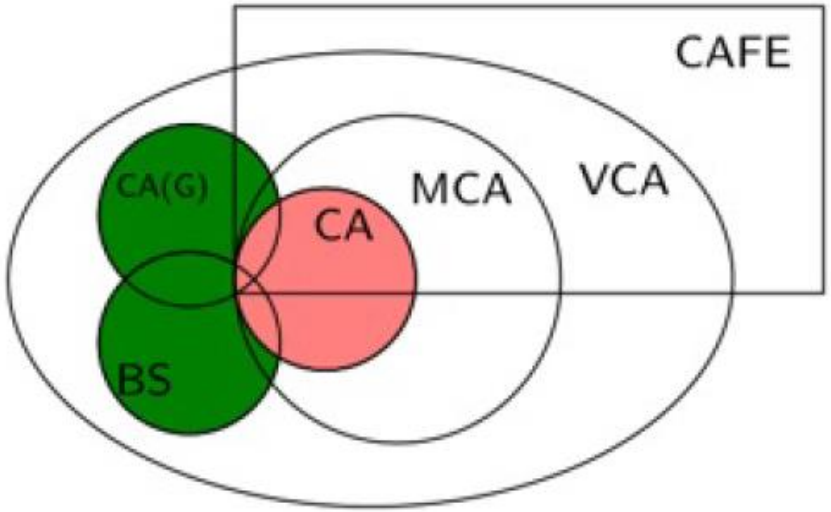
- How to efficiently test all possible k-bit input vectors for Trojan activation?

The General (Combinatorial) Test Generation Problem

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

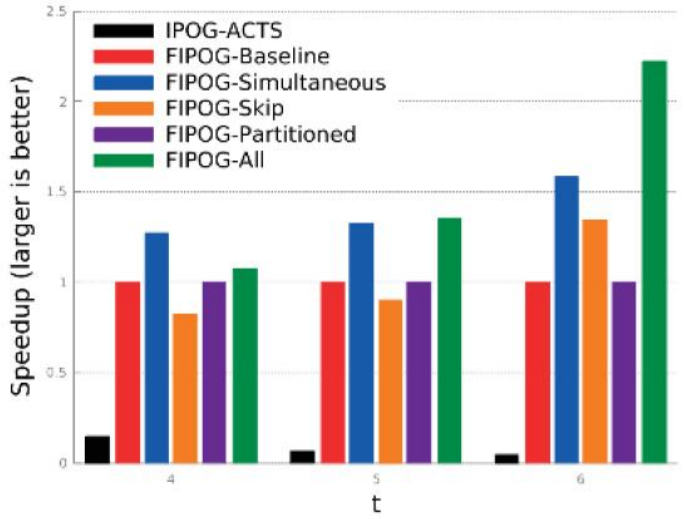
- Equivalent to finding a $CA(N; t, k, v)$ with **minimum** number of rows (also called the t-way covering problem)!
- The t-way covering problem is a **hard** combinatorial optimization problem studied for centuries

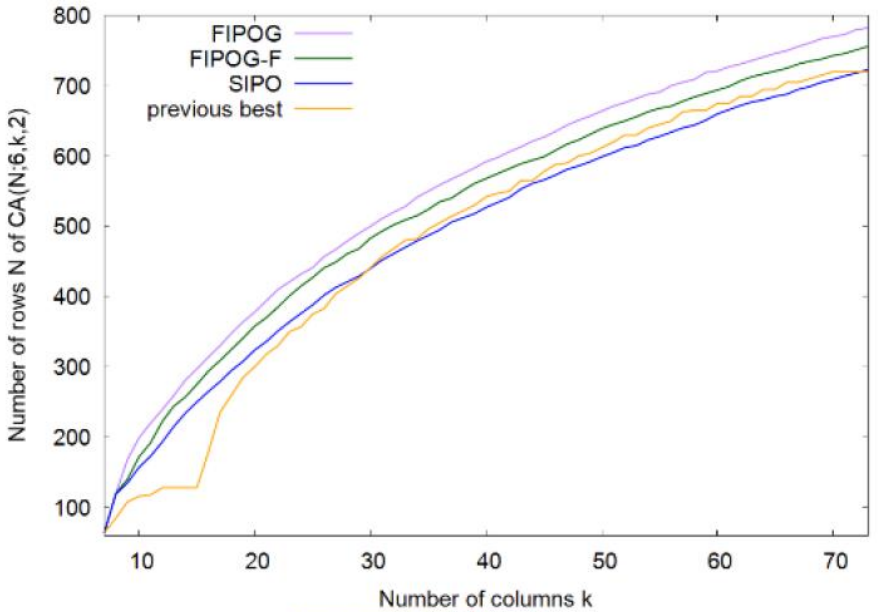
Complexity Classification of Problems of CAs

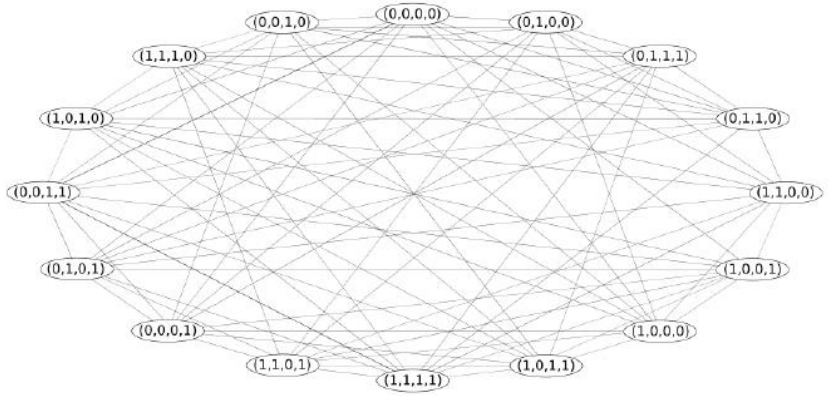


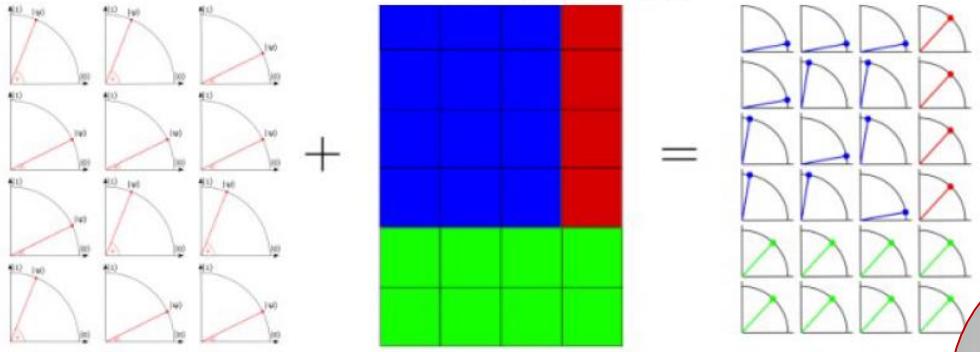
Classes of Covering Arrays	Decide Existence	Decide Size	Determine Size	Generation
optimal $CA_{2,2}$	P	P	P	P
optimal $CA_{t,v}$	P	NP	???	???
optimal $MCA_{t,v}$	P	NP	???	???
optimal BS_d	P	NP-complete	NP-hard	NP-hard
optimal $VCA_{\tau,2}$	P	NP-complete	NP-hard	NP-hard
optimal $VCA_{\tau,v}$	P	NP	???	???
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 Optimal CAs (t-way Test Sets):
Greedy, Learning and Quantum Computing Algorithms

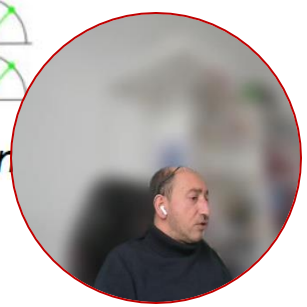






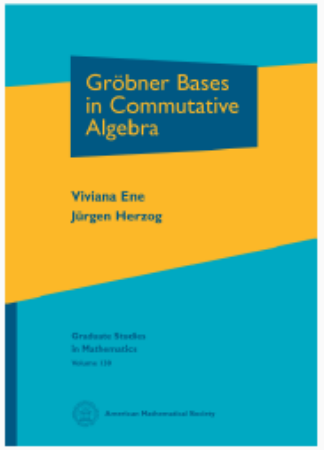


From top left clockwise: (i) speedup of efficient IPO algorithm; (ii) new optimized CAs from metaheuristic-enhan
IPO algorithm; (iii) Artificial neural networks for CA generation; (iv) quantum-inspired IPO algorithm.

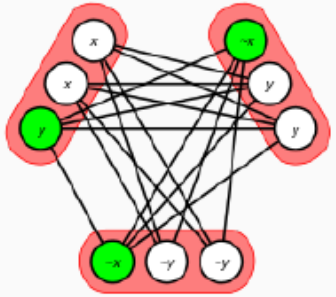


How to Construct Optimal CAs (t-way Test Sets):
Algebraic and SAT Methods

$$\begin{pmatrix} 0 & 0 & \mathbf{x_1} \\ 1 & 0 & \mathbf{x_2} \\ 0 & 1 & \mathbf{x_3} \\ 1 & 1 & \mathbf{x_4} \end{pmatrix} \cdot \begin{pmatrix} a \\ 0 \\ b \end{pmatrix} = \begin{pmatrix} b\mathbf{x_1} \\ a + \mathbf{x_2}b \\ b\mathbf{x_3} \\ a + b\mathbf{x_4} \end{pmatrix}$$



$$\begin{aligned}
x_1(x_1 - 1) &= 0 \\
x_2(x_2 - 1) &= 0 \\
x_3(x_3 - 1) &= 0 \\
x_4(x_4 - 1) &= 0 \\
(bx_1 - b)(bx_2 + a - b)(bx_3 - b)(bx_4 + a - b) &= 0 \\
(bx_1 - b)(bx_2 - b)(bx_3 + a - b)(bx_4 + a - b) &= 0 \\
(bx_1 - a - b)(bx_2 - b)(bx_3 - a - b)(bx_4 - b) &= 0 \\
(bx_1 - a - b)(bx_2 - a - b)(bx_3 - b)(bx_4 - b) &= 0 \\
b^2x_1(bx_2 + a)x_3(bx_4 + a) &= 0 \\
b^2x_1x_2(bx_3 + a)(bx_4 + a) &= 0
\end{aligned}$$



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

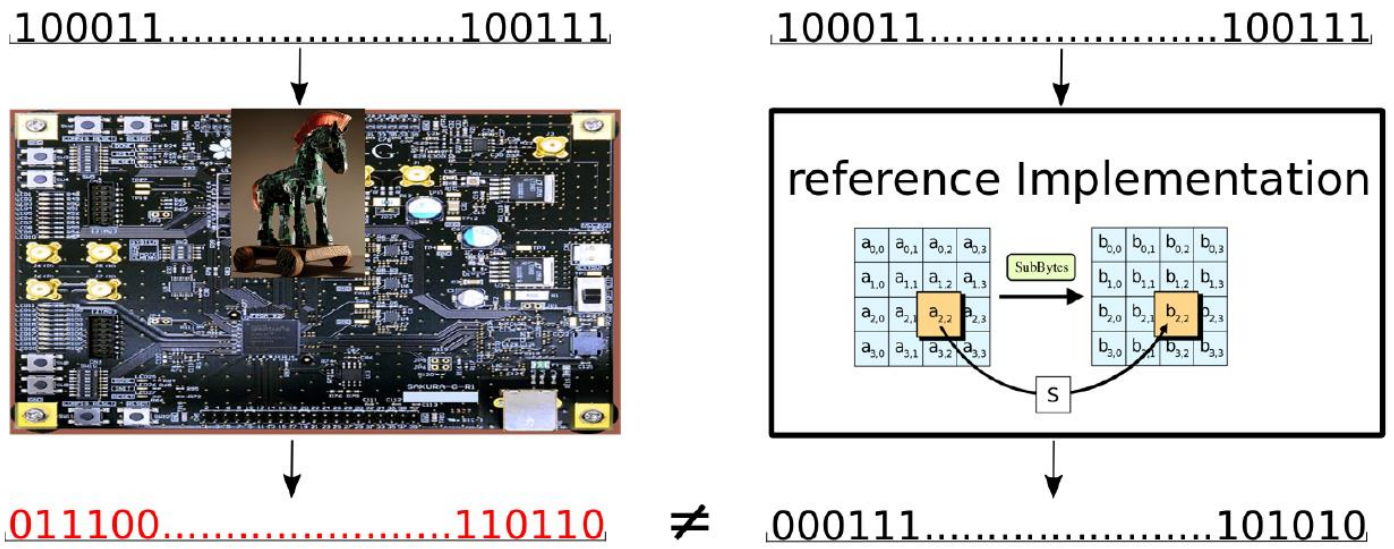
Employed CA Generation Methods:

- 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

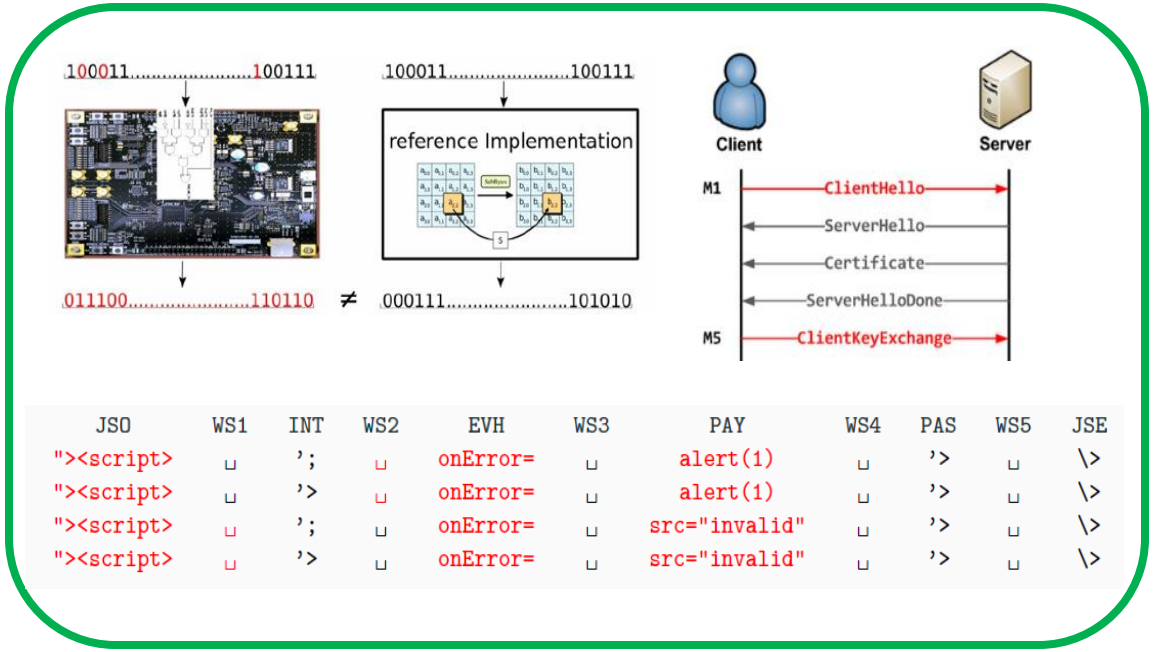
- There are about 366 ***trillion*** possible combinations for the Trojan activation;
- The whole space is covered with less than 18 ***thousand*** vectors
- .. and these vectors activate the Trojan ***hundreds*** of times

Combinatorial Security Testing (CST)

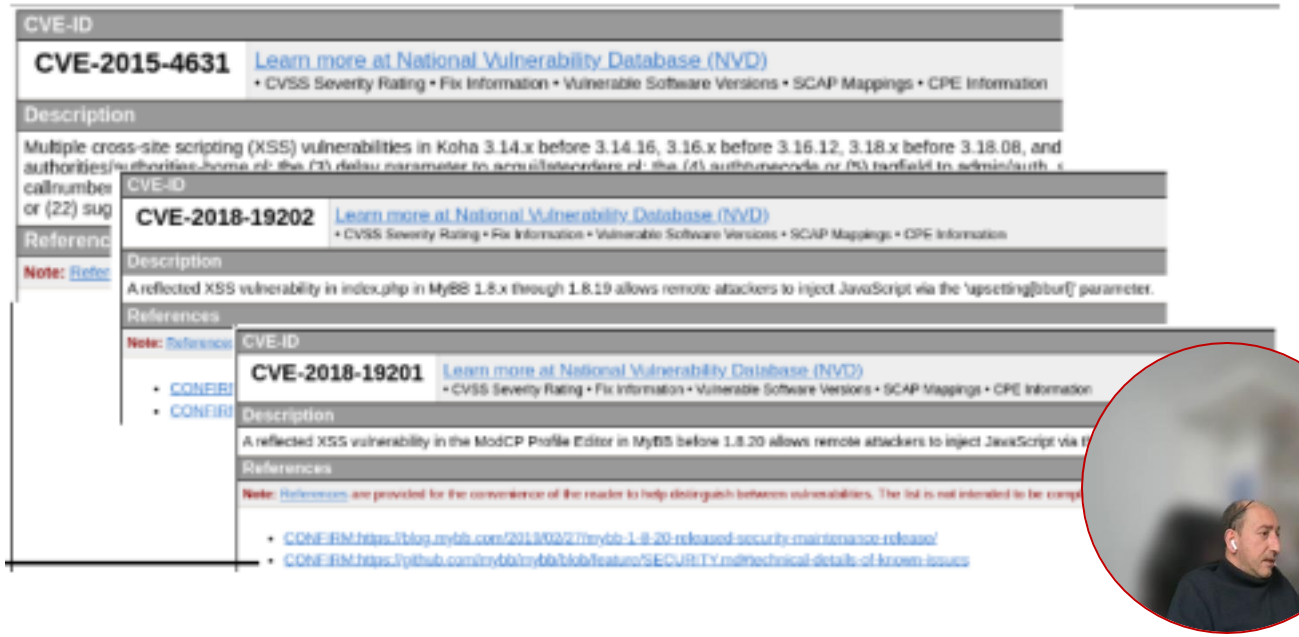
vulnerabilities found via interaction patterns

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



JSO	WS1	INT	WS2	EVH	WS3	PAY	WS4	PAS	WS5	JSE
"><script>	␣	’;	␣	onError=	␣	alert(1)	␣	’>	␣	\>
"><script>	␣	’>	␣	onError=	␣	alert(1)	␣	’>	␣	\>
"><script>	␣	’;	␣	onError=	␣	src="invalid"	␣	’>	␣	\>
"><script>	␣	’>	␣	onError=	␣	src="invalid"	␣	’>	␣	\>



Resilience of Complex Systems

Conclusion & Future Outlook

Complex Systems

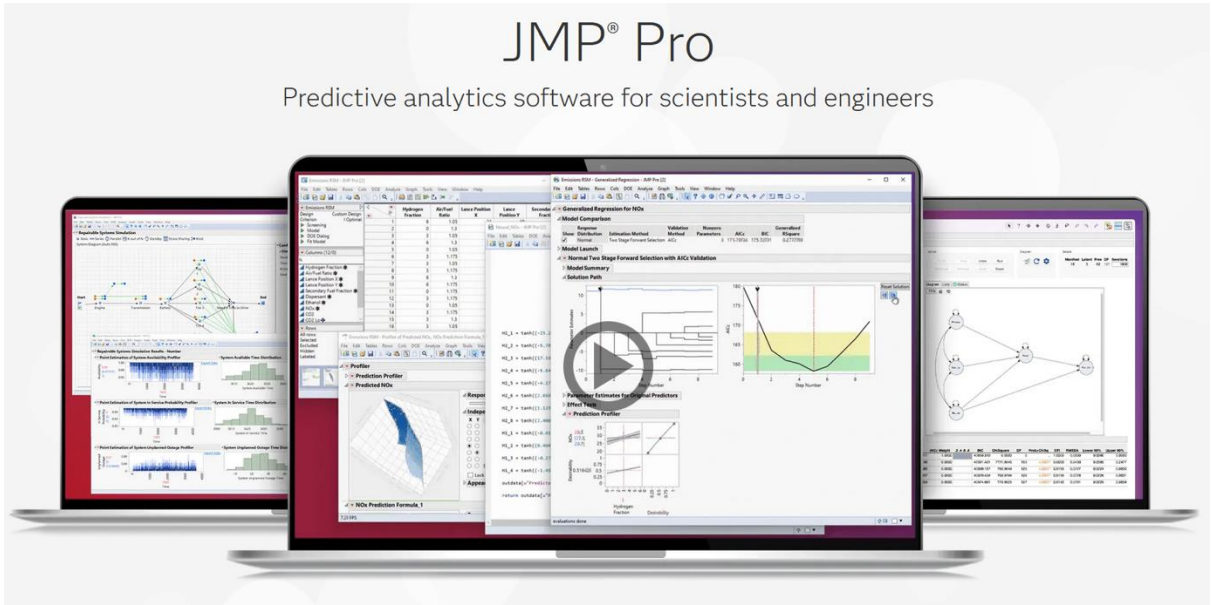
- Critical Software
- Cyber-physical Systems

How are these connected?

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

- **Complex Systems (Systems Under Test / Attack Paths)**
  - **Software Engineering: Software Failures ➔ Interaction Faults**
  - **Autonomous Driving: System Failures ➔ Interaction Faults**
  - **Supply Chain Security: Hardware Trojans ➔ (Optimal) Covering Arrays**

How to Construct Interaction Patterns (**at Large**) ☺?



Core capabilities of JMP Pro

Predictive modeling and cross-validation

Use the set of rich algorithms in JMP Pro to build and validate your models more effectively.

Model screening and comparison

Build a variety of models and determine the best one for the problem you are trying to solve.

Formula Depot and score code

Organize your models and save model score code in SAS, C, Python, JavaScript, or SQL.

Reliability block diagrams

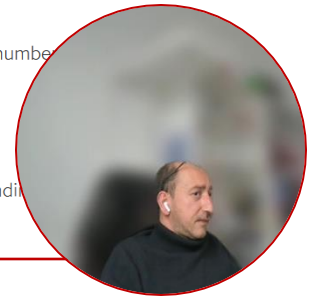
Easily fix weak spots in your system and be better informed to prevent future system failures.

Repairable systems simulation

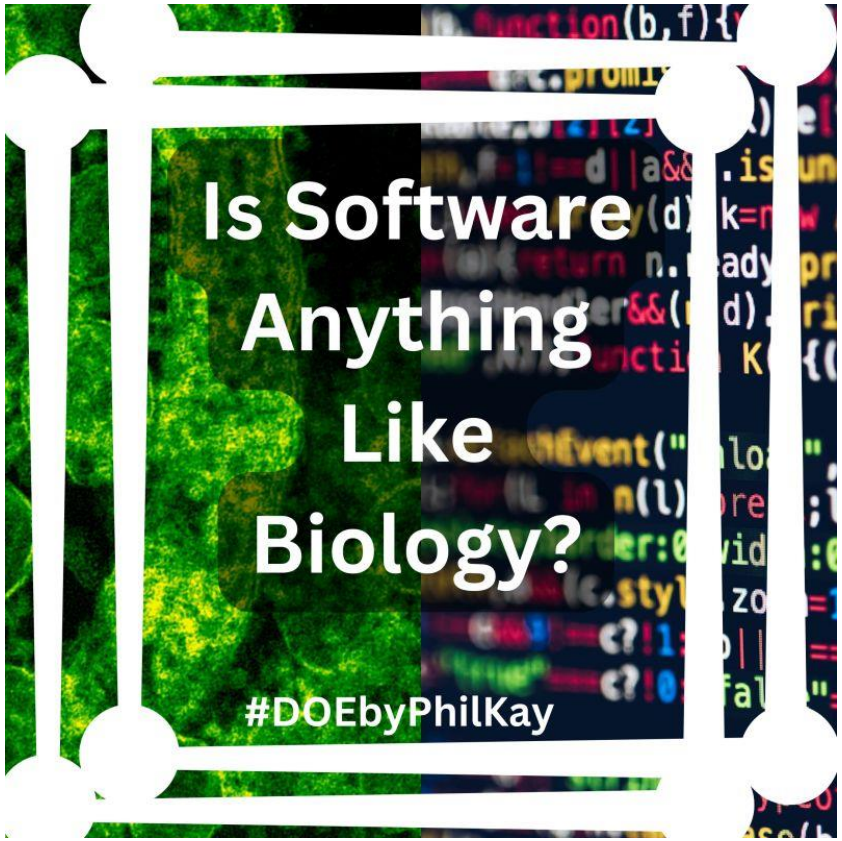
Simulate system repair events to understand downtime, numbe
repairable events.

Covering arrays

Design your experiment to maximize the probability of findi
minimizing cost and time.



Future Outlook: Human Body Augmentations as Complex Software Systems & Medical IT Devices

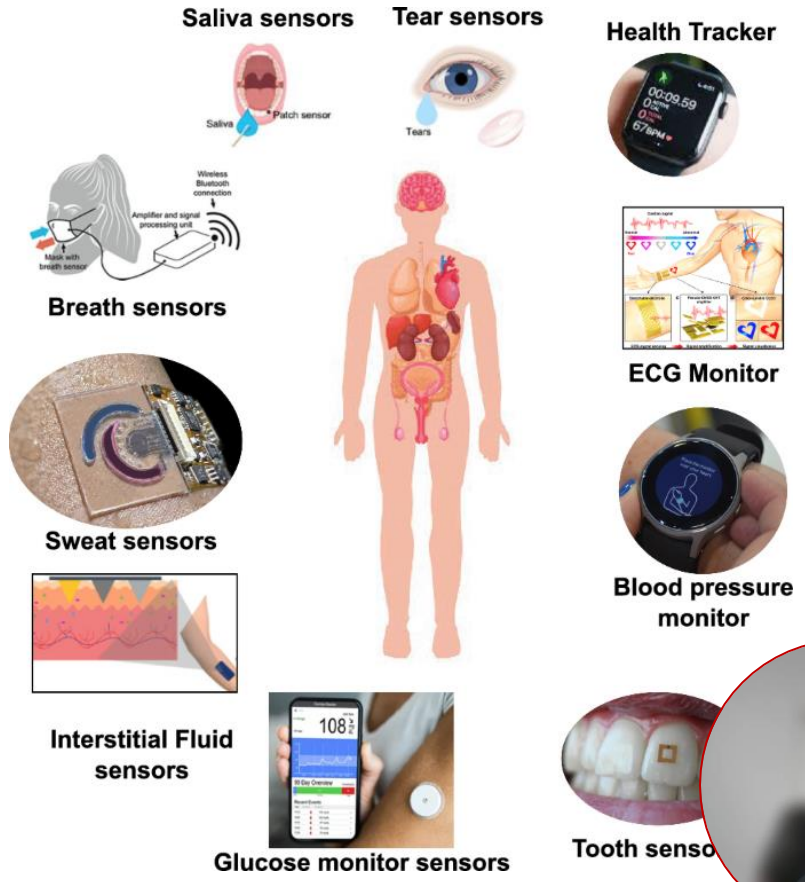


International Journal of Information Security (2024) 23:2225–2268
https://doi.org/10.1007/s10207-024-00826-y

REGULAR CONTRIBUTION

MEDICALHARM: A threat modeling designed for modern medical devices and a comprehensive study on effectiveness, user satisfaction, and security perspectives

Emmanuel Kwarteng[1] · Mumin Cebe[1]



Biomedical security: Software errors and security vulnerabilities can have fatal consequences → need for combinatorial testing!

Conclusion & Highlights

No matter how well a system is engineered, it will always be prone to errors and vulnerabilities

Complex systems have emergent and unpredictable behavior

Resilience & testing is essential in an interconnected digital society

Thank you very much for your attention!

Questions / Comments ?

dimitrios.simos@plus.ac.at
dimitrios.simos@fh-salzburg.ac.at