Evolutionary Computation and Machine Learning in Cryptology

Stjepan Picek* and Domagoj Jakobovic**

* TU Delft, The Netherlands
** University of Zagreb, Croatia
s.picek@tudelft.com, domagoj.jakobovic@fer.hr
www.aisylab.com

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author(s).
GECCO '21 Companion, Lille, France
© 2021 Copyright is held by the owner/author(s).
ACM ISBN 978-1-4503-8351-6/21/07. . . $15.00.
doi:10.1145/3449726.3461420

Instructors

Stjepan Picek finished his PhD in 2015 as a double doctorate under the supervision of Lejla Batina, Elena Marchiori (Radboud University Nijmegen, The Netherlands), and Domagoj Jakobovic (Faculty of Electrical Engineering and Computing, Croatia). Currently, Stjepan is working as an assistant professor at TU Delft, The Netherlands. Before that, Stjepan worked at MIT, USA and KU Leuven, Belgium. Stjepan also has several years of experience working in industry and government. He regularly publishes papers in both evolutionary computation and cryptographic conferences and journals. Besides that, he is a member of several professional societies (ACM, IEEE, IACR).

Domagoj Jakobovic received his Ph.D. degree in 2005 at the University of Zagreb, Croatia, on the subject of generating scheduling heuristics with genetic programming. He is currently a full professor at the Department of Electronics, Microelectronics, Computer and Intelligent Systems at the University of Zagreb. His research interests include evolutionary algorithms, optimization methods, and parallel algorithms. Most notable contributions are in the area of a machine supported scheduling, optimization problems in cryptography, parallelization, and improvement of evolutionary algorithms. He has published more than 100 papers, lead several research projects, and serves as a reviewer for many international journals and conferences.

Outline

Evolutionary Computation

- Research area within computer science that draws inspiration from the process of natural evolution.
- Evolutionary computation (EC): population based metaheuristic optimization methods that use biology inspired mechanisms like selection, crossover or survival of the fittest.
- Genetic Algorithm (GA), Tree based Genetic Programming (GP), Cartesian Genetic Programming (CGP), Evolution Strategy (ES), NSGA-II, etc.
- [60, 77, 113, 11, 40]

Machine Learning

- Machine Learning (ML) is a subfield of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence.
- Deep learning (DL) is a special type of machine learning.
- Deep learning is designed to overcome problems that traditional machine learning cannot.
- Such problems are working with high-dimensional data, how to achieve generalization.
- In cryptology applications, mostly supervised learning is used.
- Supervised learning represents learning algorithms that learn to associate some input with some output, given a training set of examples of inputs and outputs.
- [13, 114, 50]

Cryptology

- Cryptology (from Greek words kryptos, which means hidden and logos, which means word) is the scientific study of cryptography and cryptanalysis.
- Cryptography is a science (and art) of secret writing to hide the meaning of a message.
- Cryptanalysis is a science of analyzing ciphers in order to find weaknesses in them.

What is Cryptography?

Historically: cryptography has been the art of hiding the meaning of messages to protect their confidentiality.

- Origins dating back to ancient Egypt (˜2000 BCE).
- Combination of the Greek words *kryptòs* (hidden) and *graphìa* (writing).
- Mainly relied on unsound methods till the 20th century, e.g.:
  - Monoalphabetic and polyalphabetic substitutions (*Caesar's cipher*, *Vigenère's cipher*, ...)
  - Transpositions (The *Scytale*, ...)
- Collectively, these methods are also known as *classical* cryptography.
- We will not be investigating AI for classical cryptography!

Modern Cryptography

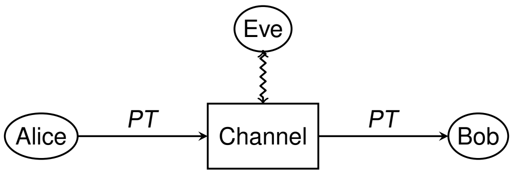
Modern cryptography is a science that studies the methods to allow *secure communication* in presence of *adversaries*.

- Started in the mid 20th century, with the seminal work by Shannon.
- Reliance on *precise* mathematical definitions and *rigorous* proofs to guarantee certain security levels.
- Used also for other goals other than message confidentiality:
  - Message integrity
  - Authentication
  - Non-repudiation
- Nowadays, modern cryptography is at the core of many protocols for secure digital communication (e.g., SSL/TLS, ...).

The Basic Scenario

- *Alice* wants to send a message to *Bob* over a communication *channel*, so that only Bob can read it (confidentiality).
- We call the message *plaintext* (PT), and assume that it is a string over an alphabet Σ, that is

$$PT \in \Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i, \quad \Sigma^i \text{ is the set of strings of length } i$$



- *Eve*, the adversary, can read *everything* transmitted over the channel.

The Basic Scenario – Introducing Cryptosystems

To prevent Eve from reading PT, Alice and Bob adopt the following protocol:

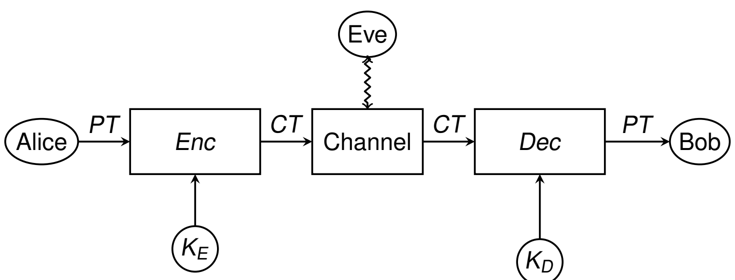
1. Alice uses an *Encryption function*, *Enc*, which depends on an *encryption key* K_E, and transforms PT into a *ciphertext* CT:

$$CT = Enc_{K_E}(PT)$$

2. Alice sends CT over the channel, and Eve observes CT.
3. Bob uses a *Decryption function*, *Dec*, that depends on a *decryption key* K_D, to transform CT back into PT:

$$PT = Dec_{K_D}(CT)$$

Cryptosystem – Block scheme



0.5

- PT: plaintext
- *Enc*: encryption function
- K_E: encryption key

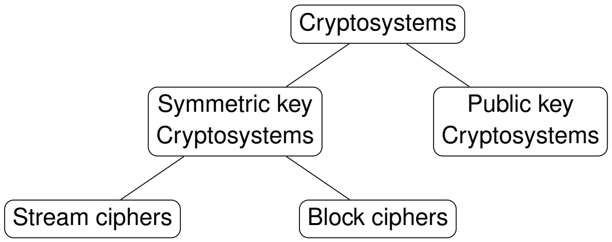
0.5

- CT: ciphertext
- *Dec*: decryption function
- K_D: decryption key

Cryptosystem – Basic properties

- *Unique decodability*: given K_E and K_D, for all plaintext PT it must be the case that $Dec_{K_D}(Enc_{K_E}(PT)) = PT$.
- For all PT, it must be *easy* for Alice to compute the ciphertext $CT = Enc_{K_E}(PT)$ by knowing K_E and *Enc*.
- For all CT, it must be *easy* for Bob to recover the plaintext $PT = Dec_{K_D}(CT)$ by knowing K_D and *Dec*.
- Given CT, it must be *extremely difficult* for Eve to recover PT *without* knowing the decryption key K_D.
- It is always assumed that the encryption and decryption functions are known to Eve (*Kerchoff's principle*).

Classification of Cryptosystems



Symmetric and Public Key Cryptosystems

Symmetric key cryptosystems:

- The key used for encryption and decryption is the same.
- Alice and Bob must agree on this key before the communication takes place.
- Can be further classified in:
  - *Stream ciphers*: Encryption and decryption process single symbols of the plaintext and the ciphertext.
  - *Block ciphers*: Encryption and decryption work over *blocks* of fixed length of symbols.

Public key (or *asymmetric key*) cryptosystems:

- The keys used for encryption and decryption differ.
- Alice uses Bob's *public key* K_E to encrypt, while Bob uses his own *private key* K_D to decrypt.
- [72, 75, 106, 181]

Cryptanalysis

- A study of methods for obtaining the meaning of encrypted information, without access to the secret information that is typically required to do so.
- Commonly, it involves knowing how the system works and finding a secret key.
- The major categories of cryptanalysis are *ciphertext only*, *known plaintext*, *chosen plaintext*, and *chosen ciphertext*.
- Common examples are linear and differential cryptanalysis.

Implementation Attacks

- Implementation attacks do not aim at the weaknesses of the algorithm itself but at the actual implementations on cryptographic devices.
- Power, sound, light, electromagnetic radiation.
- Implementation attacks are among the most powerful known attacks against cryptographic devices.
- Common types of implementation attacks are side-channel attacks and fault injection attacks.
- Side-channel attacks are passive and non-invasive attacks [96].
- Fault injection attacks are active attacks since they enforce the target to work outside the nominal operating range.

Evolutionary Computation and Machine Learning in Cryptology

- We will notice that such artificial intelligence (AI) techniques are used more often in **attacks** than in **constructions**.
- More precisely, they are used more often in attacks by the crypto community.
- There are two main reasons for this:
  1. It is easier to validate that the attack works. Indeed, we require only a successful attack as proof. For constructions, it is difficult to capture all the notions of security when using data or fitness functions.
  2. Attacks are made after the constructions are done. So, there is the effect of timeliness. For constructions, one needs to use AI while designing the system, which is often not possible. Later, even if AI produces improved constructions, it is hard to change the already made design.
- [137, 24, 135, 136].

Evolutionary Computation and Machine Learning in Cryptology

- For machine learning, a common tool is scikit-learn.
- For deep learning, Keras.
- For EC, different approaches use different tools.
- ECF is a C++ framework intended for application of any type of evolutionary computation: http://gp.zemris.fer.hr/
- Details about projects concerning evolutionary computation and cryptology: http://evocrypt.zemris.fer.hr/

Evolutionary Computation and Machine Learning in Cryptology

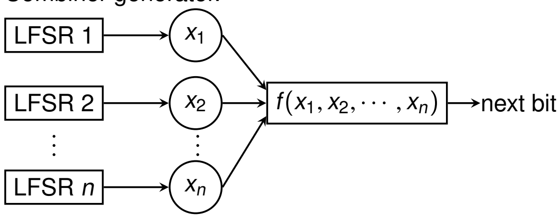
- How to solve hard problems in cryptology?
- Problems need to be hard (to be worthwhile), but not too difficult (to be impossible to solve).
- Plenitude of problems and possible methods to solve them.

Boolean Functions

- The easiest problem to start.
- A natural mapping between the truth table representation of Boolean functions and representation of solutions in EC.
- Boolean functions are important cryptographic primitive and often used in stream ciphers as the source of nonlinearity.
- Boolean functions are commonly used in combiner or filter generators.

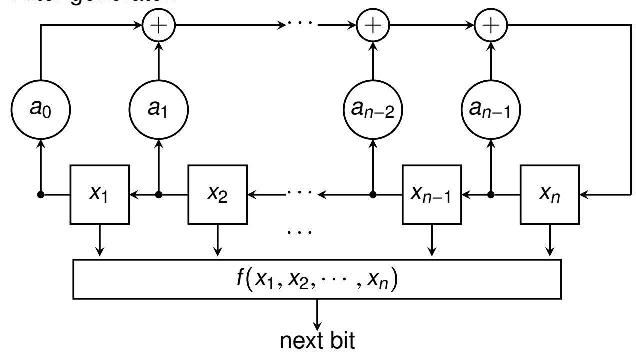
Boolean Functions

Combiner generator.



Boolean Functions

Filter generator.



Boolean Functions

- Filter and combiner generators, as depicted in previous slides are not commonly used in crypto anymore.
- Evolving Boolean functions is more interesting from the perspective of a difficult optimization problem, and not designing cryptographic primitive that will be used in ciphers.

Truth table input		Truth table output
x_1	x_0	y
0	0	0
0	1	1
1	0	1
1	1	0

Figure: Boolean function representation with truth table (two variables).

Boolean Functions

- Three main directions in the evolution of Boolean functions:
  1. Evolution of Boolean functions fulfilling a number of cryptographic properties (that can be used in combiner or filter generators). Additionally, with some special properties that are useful for masking countermeasures against side-channel attacks.
  2. Evolution of bent Boolean functions. Bent Boolean functions are maximally nonlinear but not balanced, and as such, not directly usable in crypto. Still, this represents an interesting benchmark problem.
  3. Evolution of algebraic constructions that are used to design Boolean functions.

Boolean Functions

- Depending on the setting, we are interested in a number of properties (balancedness, nonlinearity, algebraic degree, correlation immunity, algebraic immunity), where some of those properties are conflicting.
- Search space size is 2^{2^n}.
- Representing solutions in the truth table form requires a string of bits of length 2^n.
- For smaller sizes, bitstring, integer, and floating-point also give good results.
- Currently, the best results, in general, are achieved with GP/CGP.
- Such results are comparable with those from algebraic constructions.

Boolean Functions

- The problem is scaling as algebraic constructions work for many dimensions, while EC has problems when considering Boolean functions with more than, e.g., 16 variables.
- Interesting research directions: *i*) finding Boolean functions with specific properties that were not found with algebraic constructions, *ii*) extending EC to work with larger Boolean functions, and *iii*) evolving constructions of Boolean functions.
- General information about Boolean functions in cryptography [19, 31, 20].
- [112, 108, 154, 140, 109, 2, 110, 111, 27, 28, 25, 64, 159, 156, 143, 163, 98, 97, 101, 148, 100, 99, 134, 167, 142, 152, 155, 15, 102, 70].

Boolean Functions

Bitstring representation

```
<Individual size="1">
        <FitnessMax value="116.727"/>
        <BitString size="256">11110000001011100110101111010001110110000010110
        011100000001010001110101011010101001010001001110110000100001111001100
        1000010110000001010101111111111111101110110000111101011101111000101010
        001011110000001101001010010001111101001111011101000111010010010001001
        00</BitString>
</Individual>

correlation immunity: 0; nonlinearity: 116; algebraic degree: 6
```

- Boolean function of eight variables represented with a binary array of size 256 (ECF).
- Optimization of nonlinearity while maintaining balancedness.

Boolean Functions

Floating point representation

```
<Individual size="1">
        <FitnessMax value="114.938"/>
        <FloatingPoint size="32">       0.26875         0.669872        0.762153        0.246787
        0.443393        0.498733        0.664411        0.00021305        0.278248        0.622918
        0.889779        0.321942        0.982994        0.554419        0.0779042        0.663329
        0.125795        0.595173        0.540512        0.132081        0.112745        0.59266
        0.847716        0.888488        0.592867        0.655954        0.770198        0.198452
        0.348636        0.620424        0.767249        0.673829</FloatingPoint>
</Individual>

Truth table:
0100010010101011110000110011111101110001011111111010101000000000010001111
00111111110001101010010111110111000110100010011101010010010000001001100010
0010100010000100011100100101111101100111100011100101111010011111000101001
1001001011001100111101100010010101100
correlation immunity: 0; nonlinearity: 114; algebraic degree: 7
```

- Boolean function of eight variables represented with a floating point array (ECF).
- In this example, each floating point value maps to eight bits in the truth table (either binary or Gray encoding, concatenated or distributed bits).

Boolean Functions

GP representation

```
<Individual size="1">
        <FitnessMax value="120"/>
        <Tree size="29">XOR XOR OR XNOR v0 XOR v5 v3 AND NOT v0 NOT v3
        NOT XOR OR v1 v6 OR v7 v3 XOR AND2 v5 v6 IF v4 v2 v1 </Tree>
</Individual>

infix: ((((v0 XNOR (v5 XOR v3)) OR (˜(v0) AND ˜(v3))) XOR ˜(((v1 OR v6)
XOR (v7 OR v3)))) XOR ((v5 AND2 v6) XOR IF(v4, v2, v1)))

Truth table:
0101010111111111011010011100001111111111010101011100001101101001100110010
0110011101001010000011111100110001100110111100000101101010101010111111110
0101101100001100000000001010101001111000110100101100110001100110101011010000
0111100110011011001100000111101011010
correlation immunity: 0; nonlinearity: 120; algebraic degree: 2
```

- Boolean function of eight variables represented with a GP tree (ECF).
- Optimizing for maximally nonlinear functions (bent functions).

S-boxes

- Natural extension from the Boolean function case.
- S-boxes (Substitution Boxes) are also called vectorial Boolean functions.
- Often used in block ciphers as a source of nonlinearity.
- However, this problem is much more difficult!
- S-box of dimension $n \times m$ has m output Boolean functions, but for several cryptographic properties we need to check all linear combinations of those functions (there are $2^n - 1$ linear combinations to consider).

Truth table input		Linear combinations			Walsh-Hadamard spectrum		
x_1	x_0	y_1	y_0	$y_1 \oplus y_0$	y_1	y_0	$y_1 \oplus y_0$
0	0	0	1	1	0	0	0
0	1	1	0	1	0	-4	0
1	0	1	1	0	0	0	-4
1	1	0	0	0	4	0	0

Figure: S-box with two inputs x and outputs y.

S-boxes

- For an S-box of size $n \times m$, the search space size equals 2^{m2^n}.
- Commonly (with EC), we explore cases where $n = m$, which means that for $n = m = 8$, the search space size equals $2^{2\,048}$.
- Common sizes to evolve with EC are from 3×3 to 8×8.
- Common solution representations are the same as for Boolean functions, plus permutation (which enforces bijectivity).
- Note, if using the tree representation, one actually evolves n trees.
- For smaller sizes, (up to 4×4) all solution representations work well.

S-boxes

- Similar to the Boolean function case, there are three main approaches to construct S-boxes: *i*) algebraic constructions, *i*) random search, and *iii*) heuristics.
- EC is commonly used to:
  1. Find bijective S-boxes with high nonlinearity (and low differential uniformity). Note that for such S-boxes, we know several algebraic constructions.
  2. To find S-boxes with additional properties. These commonly go into the direction of resilience against side-channel attacks.
  3. To find more efficient implementations of S-boxes (efficient in terms of area, power).

S-boxes

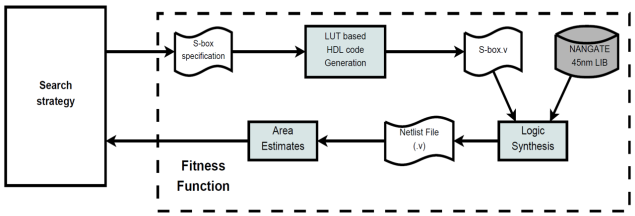


Figure: When considering S-box implementation properties, it is important to be able to communicate between the EC system and the system that checks the implementation perspective.

S-boxes

- The best results are obtained with tree representation and the cellular automata approach.
- CA representation was the first to obtain bijective S-boxes with optimal cryptographic properties for sizes up to 7×7 (not including 6×6 as there, no EC technique found the bijective S-box with the best possible differential uniformity).
- Already for size 8×8, EC results are far from those obtained with algebraic constructions (except if the initial population is seeded with good S-boxes).

S-boxes

Table: Best known values for bijective S-boxes. For 8×8, we give the best known results while for smaller sizes, we give the optimal values. For bijective S-boxes (and in $\mathbb{F}_2$), both nonlinearity and differential uniformity (δ) can be even values only. The worst possible values are 0 for nonlinearity (i.e., the S-box is linear), and 2^n for differential uniformity.

Property	3×3	4×4	5×5	6×6	7×7	8×8
nonlinearity	2	4	12	24	56	112
δ	2	4	2	2	2	4

S-boxes

- S-boxes are becoming less popular due to the rise of permutation-based cryptography, but they are still widely used.
- Naturally, most EC solutions are obtained much after the cipher design, so it is impractical to change the whole cipher simply to accommodate a new S-box.
- Interesting challenges for EC are *i*) obtain S-box of size 8×8 with the same properties as with algebraic constructions, and *ii*) evolve algebraic constructions of S-boxes (either primary or secondary).
- General information about S-boxes [19].

S-boxes

- Tool that evaluates cryptographic properties of Boolean functions and S-boxes [141].
- Discussion on benchmarking EC with cryptographic problems [157].
- [26, 39, 43, 66, 67, 107, 147, 130, 162, 164, 165, 104, 161, 160, 103, 16, 188, 6, 168, 170, 146, 132, 21, 133, 89, 46, 69, 153, 35].

S-boxes

```
<Individual size="1">
        <FitnessMax value="84.0938"/>
        <Tree size="13">XOR v1 IF v3 IF v0 v5 v3 XOR NOR v5 v0 v2 </Tree>
</Individual>
infix: (v1 XOR IF(v3, IF(v0, v5, v3), ((v5 NOR v0) XOR v2)))
Sbox:
1001011010011001100101101001100100111100110011000011110011001100
1000011101111000110100101101001010000111011110001101001011010010
1100000000111111001111111100000011110011000011001111001100001100
1111010100000101000010101111101001011111101011111010000001010000
1011101101110111010001000111011101000100100010001011101110001000
1001101010011010011010100110101001100101011001010110101001101010
Permutation:
63 12 24 57 48 11 51 26 33 18 22 55 39 28 52 29 3 50 36 7 44 21 47 4 15 62 56 27
41 16 58 17 6 6037 13 9 59 14 46 25 35 42 2 31 45 8 40 30 38 61 23 49 1 54 20 19
43 32 10 53 5 34 0
```

- S-box CA representation (cellular automata rule for a single column repeated six times).
- End result is a bijective S-box with six inputs and outputs (ECF).

Pseudorandom Number Generators

- In the same way one builds Boolean functions and S-boxes as cryptographic primitives, it is possible to extend the approach and build pseudorandom number generators (PRNGs).
- In cryptography, random number generators (RNGs) play an important role.
- Most of the time, we need true random number generators (TRNGs), but still, there are applications for pseudorandom number generators.
- TRNG is a device for which the output values depend on some unpredictable source that produces entropy.
- PRNGs represent mechanisms that produce random numbers by performing a deterministic algorithm on a randomly selected seed.

Pseudorandom Number Generators

- Commonly, we want to find extremely fast and small PRNGs that pass all NIST tests [5].
- We can use GP and CGP to evolve PRNGs.
- CGP has the advantage that it can have multiple outputs, which means it can output more bits.
- Fitness function needs to be simple yet powerful.
- We can use the approximate entropy test from the NIST statistical test suite as a fitness function.

Pseudorandom Number Generators

```
void CGP(uint x0,uint x1,uint x2,uint x3,
uint* z0,uint* z1,uint* z2,uint* z3) {
    uint y4 = x0 & x1;
    uint y5 = x2 ^ x3;
    uint y6 = (y5 >> 1) | (y5 << 31);
    uint y7 = p1(y6);
    uint y8 = x3 ^ y7;
    uint y9 = p1(y8);
    uint y10 = y6 ^ y9;
    uint y11 = (y9 << 1) | (y9 >> 31);
    uint y12 = const;
    uint y13 = p1(y10);
    uint y14 = y12 ^ y11;
    uint y15 = y12 ^ y13;
    uint y16 = (y15 >> 1) | (y15 << 31);
    uint y17 = y10 ^ y16;
    uint y18 = p1(y17);
    uint y19 = y18 >> 1;
    uint y20 = y18 ^ y4;
    uint y21 = p1(y20);
    uint y22 = y18 ^ y21;
    uint y23 = p1(y18);
    uint y24 = y19 ^ y18;
    uint y25 = y23 ^ y19;
    uint y26 = y22 ^ y14;
    *z0 = y18;
    *z1 = y25;
    *z2 = y26;
    *z3 = y24;
}
```

Figure: Example of a PRNG evolved with CGP.

Pseudorandom Number Generators

- The same technique can be used to produce PRNGs on-the-fly.
- Then, we can use evolvable hardware [182] that constantly updates the PRNG part.

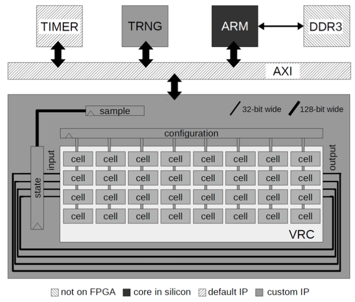


Figure: Example of a system that uses CGP and evolves PRNGs on-the-fly.

Pseudorandom Number Generators

- Common challenges are to find fast, reliable, and small PRNGs with EC.
- Usually, the problem is to have an efficient fitness function.
- [84, 128, 76, 54, 172, 200, 171, 169]

Ciphers

- Instead of evolving only parts of ciphers as we discussed up to now, we can consider whether AI could build the whole cipher.
- There are commonalities with the previous topic as one can consider PRNG as the whole system and not only a part of it.
- The first effort in this direction uses adversarial neural networks.
- There are three networks representing Alice, Bob, and Eve and they compete to find a cipher that is usable and secure [1].

Ciphers

- It is also possible to evolve ciphers with evolutionary algorithms [158].
- One can use competitive coevolutionary algorithms to train a cipher (Alice) and attacker (Eve).
- Still, these works can be regarded as proofs of concept only.
- There are multiple possible research directions: *i*) making the ciphers more efficient and secure as now, they work with very small inputs and are trivial to break, *ii*) making the attacker strategies smarter, *iii*) explore different levels of information provided to Alice and Eve – do they design all from scratch, or do they have background info? *iv*) extending to other concepts and not only block ciphers (and in block ciphers, consider whether evolving the round function or the whole cipher), and *v*) AI “reinventing” of crypto techniques that are well-known and commonly used.

Ciphers

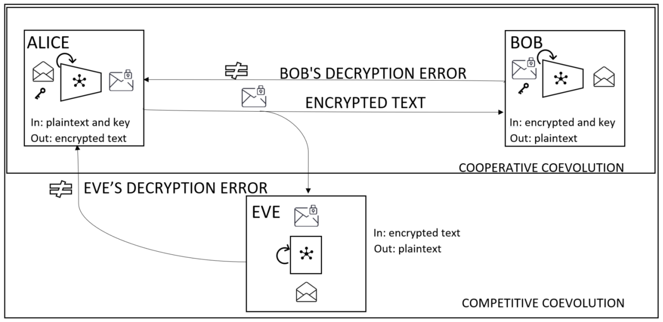


Figure: Coevolutionary system for the evolution of ciphers.

Attacks on Ciphers

- Up to now, EC has been successfully applied to break classical cryptography, see, e.g., [120] or simplified ciphers [38].
- Additionally, EC was also used as a helper tool in SAT-based attacks [123].
- To the best of our knowledge, these results are still far from being able to say that EC was used to conduct cryptanalysis of a modern cipher.
- On the other hand, neural networks proved to be a more suitable option here.
- A common direction is to use neural networks to exploit the properties of ciphers.

Attacks on Ciphers

- For example, train neural networks to distinguish the output of a cipher with a given input difference from random data.
- These approaches represent very interesting research direction as with neural networks, one can reach/surpass state-of-the-art cryptanalysis results.
- [49, 85, 83, 33, 32, 23, 3, 10, 187, 74, 63, 65, 9].

Quantum Protocols

- Instead of operating on primitive or cipher level, EC can be used to evolve protocols also.
- As an example, it is possible to use EC to evolve novel quantum key distribution (QKD) protocols designed to counter attacks against the system in order to optimize the speed of secure communication.
- In essence, the goal is to evolve protocols as quantum circuits [185].
- Then, it is possible to define whether EC must work on the specific, user-defined template of a protocol or without explicit rules on how to access quantum communication channel.

Quantum Protocols

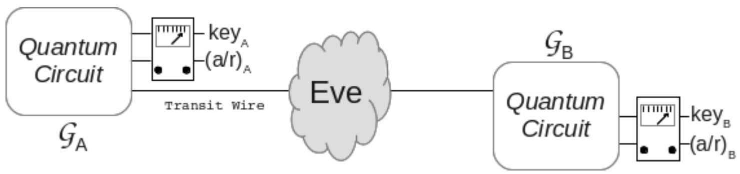


Figure: QKD protocol depicted as two circuits G_A and G_B. First, run circuit G_A, and then, the wire is measured, yielding a classical bit. Then, Eve is allowed to attack the transit line. Finally, G_B circuit is run, acting on the transit wire, and additional wires private to party B. Then, B's wire is measured.

Quantum Protocols

- This research direction looks very interesting as the current results are very good and there are not many other approaches to solve this.
- [78, 79, 80].

Quantum Protocols


```
<Individual size="5">
        <FitnessMax value="0.152422"/>
        <IntGenotype size="5">        2        0        3        0        0</IntGenotype>
        <IntGenotype size="5">        2        0        2        1        3</IntGenotype>
        <BitString size="5">01011</BitString>
        <IntGenotype size="5">        2        2        2        0        1</IntGenotype>
        <FloatingPoint size="15">        0.471412        0.248858        0.560975        0.554435
        0.397848        0.448713        0.328381        0.582966        0.310332        0.273257
        0.623984        0.782299        0.0205793        0.799614        0.0182854</FloatingPoint>
</Individual>
Gates A:
type 2 target 2 mode 0 control: 2
type 0 target 0 mode 1 control: 2
type 3 target 2 mode 0 control: 2
Gates B:
type 0 target 3 mode 1 control: 2
type 0 target 2 mode 1 control: 3
```


- QKD circuit representation (ECF).
- Optimization of a one-way QKD protocol, five gates total.
- The fitness function reflects the maximum obtainable key-rate.

Physically Unclonable Functions

- Physically Unclonable Functions (PUFs) are embedded or standalone devices used as a means to generate either a source of randomness or to obtain an instance-specific uniqueness for secure identification.
- This is achieved by relying on inherent uncontrollable manufacturing process variations, which results in each chip having a unique response.
- No two PUFs will give the same response when supplied with the same challenge.
- There exists no ideal PUF.
- Ideal PUF is unpredictable and without noise.
- Practical realizations depend on noise, aging, environmental variables, and process variations.

Physically Unclonable Functions

- Two types of PUFs: strong and weak.
- The difference with respect to the number of challenge-response pairs (CRPs) an attacker is allowed to obtain.
- The number of unique challenges *c* scales polynomially with the circuit area of a weak PUF.
- The number of unique challenges *c* scales exponentially with the circuit area of a strong PUF.

Physically Unclonable Functions

- Weak PUF has a limited number (typically, one or few) of responses to challenges.
- Strong PUFs have a large number of responses (with respect to different challenges).
- Strong PUFs have a virtually unlimited number of challenges *c*, but their CRPs are highly correlated.
- Given enough (often small amount) of CRPs, it is possible to build a predictive model of a strong PUF (in a way, we build a mathematical clone since it is not feasible to make analog physical clone).
- There exists no validated design of a strong PUF that is fully resilient against modeling attacks.

Physically Unclonable Functions

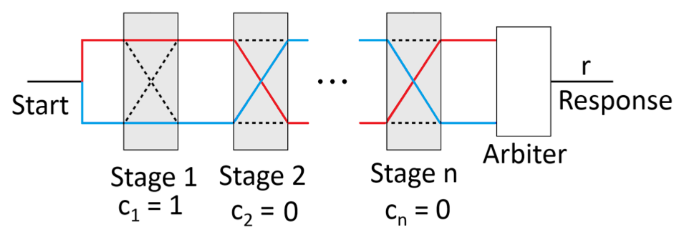


Figure: An example of a strong PUF - Arbiter PUF with *n* stages.

Physically Unclonable Functions

- Several techniques are commonly used to break strong PUFs.
- From ML domain, logistic regression, and from EC, evolution strategy.
- This domain is very interesting as AI provided results that were not possible to obtain with any other technique.
- What is more, even simple AI techniques can easily break strong PUFs.
- This also means there is not much development in the domain as attacks are easy to do, so no clear benefit of using more complex techniques, e.g., deep learning.
- [8, 198, 178, 177, 176, 7, 191, 34, 190, 186, 199, 44, 207].

Hardware Trojans

- Hardware Trojans (HTHs) are malicious hardware components that intend to leak secret information or cause malfunctioning at run-time in the chip in which they are integrated.
- Over the last decade, HTHs have gained increasing attention.
- There are no HTHs in ICs reported in real-world applications yet, there are many examples of academic research results, both on injecting and detecting/preventing HTHs.
- HTHs can be inserted by untrusted foundries and actors at different stages in the design and development of FPGAs and ASICs (Application-Specific Integrated Circuits).

Hardware Trojans

- Conventional HTHs typically modify the functionality of target circuits at the register transfer level, net-list level, layout level, or dopant level to obtain secret information directly, to induce a fault for differential fault analysis, or to disable/degrade an embedded (pseudo) random number generator.
- HTH consists of two parts: a trigger and a payload.
- The trigger usually corresponds to a rare data input (sequence), while the payload is the activity that causes the data leakage or the malfunctioning when the HTH is triggered.
- HTHs are usually inserted in special places in the design that have low testability or high slack time.
- Testability is measured through two parameters: controllability and observability.

Hardware Trojans

- Compared to research concerned with the design of Hardware Trojans, considerably more results exist related to different Hardware Trojan detection mechanisms and countermeasures.
- Most research focuses on detecting Hardware Trojans inserted during manufacturing.
- In many cases, a golden model is used that is supposed to be Trojan free to serve as a reference.
- One important question is how to get to a Trojan-free golden model.

Hardware Trojans

- Common research directions include EC for detection and prevention of HTH, ML for detection, but also EC to insert HTH.
- [53, 68, 81, 71, 179, 45, 82, 91].

Side-channel Analysis

- SCAs represent one of the most powerful categories of attacks on crypto devices.
- Profiled attacks have a prominent place as the most powerful among side-channel attacks.
- Some machine learning techniques can also serve as profiled attacks.
- There is a natural mapping between the profiled attacks and supervised learning.
- Recently, deep learning started to gain attention in the SCA community.

Side-channel Analysis

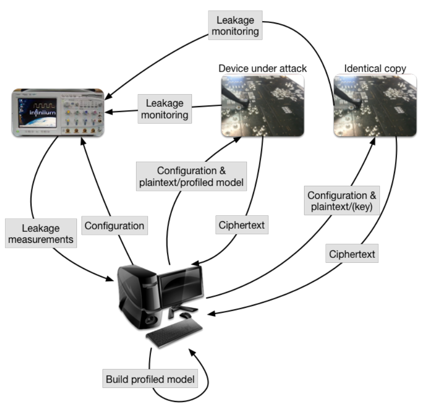


Figure: A depiction of profiled SCA.

Side-channel Analysis

- ML in SCA is an active domain for almost 20 years.
- Even the first profiled attacks - template attack and stochastic attacks are well-known techniques, quadratic discriminant analysis, and linear regression, respectively.
- Machine learning techniques were successfully applied in the attack phase, but also for pre-processing and feature engineering.
- Deep learning thrives from advantages that it does not require feature selection and that it can break implementations protected with countermeasures.
- Interesting additional ML applications are noise removal and data augmentation.

Side-channel Analysis

- This is a very active research domain with new papers appearing all the time.
- This makes this domain attractive for new researchers but also difficult due to large competition.
- There are some extra problems like lack of publicly available datasets, inconsistency between ML metrics and SCA metrics.
- Common techniques are random forest, support vector machines, multilayer perceptron, convolutional neural networks.
- While the common attack target is AES (block cipher), recently, more people are interested in attacking public-key cryptosystems also.
- While ML is mostly used in this domain, there are attempts to use evolutionary algorithms to find better leakage models (e.g., to use information from multiple S-boxes).

Side-channel Analysis

- A framework for deep learning-based SCA [126].
- [149, 57, 151, 56, 17, 93, 47, 173, 166, 73, 4, 61, 55, 150, 88, 58, 62, 138, 197, 131, 208, 124, 59, 125, 204, 210, 209, 202, 105, 127, 175, 196, 192, 203, 90, 12, 189].
- EC applications [211, 194].

Fault Injection

- A fault injection (FI) attack is successful if, after exposing the device to a specially crafted external interference, it shows an unexpected behavior exploitable by the attacker.
- Insertion of signals has to be precisely tuned for the fault injection to succeed.
- Finding the correct parameters for a successful FI can be considered as a search problem where one aims to find, within a minimum time, the parameter configurations which result in a successful fault injection.
- The source of fault can be, e.g., voltage glitching, laser, electromagnetic radiation.
- Depending on the source of the fault, the search space of possible parameters changes significantly.
- In general, the search space is too big to conduct an exhaustive search.

Fault Injection

- Commonly, one defines several possible classes for classifying a single measurement:
  1. NORMAL: smart card behaves as expected, and the glitch is ignored
  2. RESET: smart card resets as a result of the glitch
  3. MUTE: smart card stops all communication as a result of the glitch
  4. CHANGING: the response is changing when repeating measurements.
  5. SUCCESS: smart card response is a specific, predetermined value that does not happen under normal operation

Fault Injection

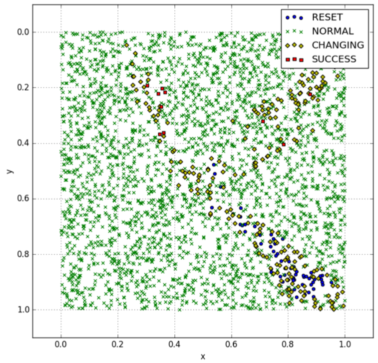


Figure: A depiction of search space for voltage glitching and two parameters.

Fault Injection

- Research domain with not many results from AI.
- Commonly used techniques are random search and grid search, so EC makes a strong alternative.
- The main issue is very expensive equipment to run fault injection campaigns.
- Mostly, EC is used, but recently, also deep learning found its place.
- Deep learning can be used to predict what a target would respond to a specific parameter combination.

Fault Injection

- Interesting research directions include *i*) working with more relevant parameters, *ii*) attacking targets with countermeasures, and *iii*) making the search algorithm more powerful, especially by considering the differences between faults and exploitable faults (those that would result in target break).
- [22, 129, 139, 205, 95, 94, 180].

Addition Chains

- Up to now, we considered EC and ML applications that directly contributed to the security of ciphers or, on the other hand, were used to break ciphers.
- Still, it is possible to use AI techniques as a helper tool to improve the systems, either from an attack or defense perspective.
- In a way, one could consider the evolution of S-boxes with good implementation properties to also belong to the setting that helps improve the system, but not necessarily its security.
- Another example of a problem like that is the finding of short *addition chains*.
- Addition chain: a sequence of positive integers where each value is a sum of two values appearing previously in the chain.
- Addition chains are used in public-key cryptosystems (e.g., for modular exponentiation).

Addition Chains

- Example: form an addition chain to target value of 60.
- Binary method: write 60 in binary: 111100; replace “1” with “DA” and “0” with “D”; cross out the first “DA” on the left; “DADADADD”, calculate:

$$1 \to 2 \to 3 \to 6 \to 7 \to 14 \to 15 \to 30 \to 60$$

- Addition chain (7 operations):

$$A^1; A^2 = A^1 * A^1; A^4 = A^2 * A^2; A^6 = A^4 * A^2; A^{12} = A^6 * A^6;$$

$$A^{24} = A^{12} * A^{12}; A^{30} = A^{24} * A^6; A^{60} = A^{30} * A^{30}$$

Addition Chains

- The problem of finding the shortest addition chain for a given exponent is relevant in cryptography.
- The problem is believed to be NP-hard.
- There is no single algorithm that can be used for any exponent.
- The best solutions so far are obtained by pen and paper method (!).
- Huge numbers, so exhaustive search is impossible.
- Heuristics should be able to help.
- The values in the ascending addition chain have the property that they are the sum of two values appearing previously in the chain.

Addition Chains

- Common solution representations are binary encoding where value 1 means that the entry number is in the chain, and 0 means the opposite, and integer encoding where every value represents an element of the chain.
- Extra care needs to be taken with crossover and mutation operators: as one changes elements, it is required to ensure that the chain remains valid!

Addition Chains

- Interesting research directions are *i*) improve the speed of the algorithm, *ii*) look for optimal chains for even larger numbers, *iii*) support special structures of numbers, and *iv*) explore different types of addition chains.
- [115, 116, 118, 30, 29, 122, 87, 145, 144, 117, 119, 36, 37].

Addition chains

Custom addition chain genotype

```
<Individual size="1">
        <FitnessMin value="37"/>
        <Chain size="38">1 2 4 8 16 32 64 128 256 512 1024 2048 4096 8192 16384
        32768 65536 131072 262144 524288 1048576 2097152 4194304 8388608 16777216
        17039360 17040384 17040385 34080769 68161538 68177922 136355844 204533766
        340889610 374970379 375494667 375494699 375494703 </Chain>
</Individual>
construction:
0/0 1/1 2/2 3/3 4/4 5/5 6/6 7/7 8/8 9/9 10/10 11/11 12/12 13/13 14/14 15/15 16/16
17/17 18/18 19/19 20/20 21/21 22/22 23/23 18/24 10/25 0/26 26/27 28/28 14/29 30/30
30/31 31/32 28/33 19/34 5/35 2/36
```

- Chain optimization with target exponent value 375 494 703 (ECF).
- The construction steps are reproduced below the chain.

Cybersecurity

- Cybersecurity is the security of computer systems and networks from information disclosure.
- While often it is considered to include only the defensive side (i.e., protection), we also consider the attack perspective.
- Cryptography is just one aspect of cybersecurity.
- In general, going from cryptography to cybersecurity increases the number of AI applications.
- We can also talk about AI for cybersecurity and the security of AI.

Intrusion Detection

- Intrusion detection: the process of monitoring the events occurring in a computer system or network and analyzing them for intrusions.
  - Intrusions: attempts to bypass the security mechanisms of a computer or network
- Detection can be *signature based* and *anomaly detection based*.
- Anomaly detection: recognizes normal network traffic and categorizes different traffic as an anomaly
- Common approaches: supervised/unsupervised classification (machine learning).
- Problems with anomaly detection: 1) the data is usually too expensive to be labeled manually, 2) data is imbalanced (much more normal than anomaly traffic), and 3) the existence of unknown attacks.
- Assumption: only one class of data (normal network traffic) is available for training!

Intrusion Detection

- GP can be used in classification (e.g., decision tree, regression tree).
- Use regression GP as *one-class classifier*.
- Procedure: learn the model on normal traffic data, test on unseen data containing anomalies (intrusions).
- Learn a model (GP function) that forces the tree output (function value) to a certain output range.
  - e.g., [1, 2], [4, 5] or [8, 9]; *same* range for all normal traffic instances
- Penalize 'trivial' models - reward/force the use of most features.
- Test the model on unseen data containing intrusions (heavily imbalanced!).
- Interpretation: outputs falling outside the defined range are classified as anomalies.
- GP results: comparable to one-class SVM.
- References: [18, 52, 14, 183, 206, 121, 41, 184, 42]

Fuzzing

- Fuzzing (fuzz testing): automated software testing technique.
- Generating inputs and feeding them to the program being tested in the hope of evoking erroneous behavior or increasing code coverage.
- *Mutation-based fuzzing*: uses a dataset of test cases (a corpus):
  - selects a test case,
  - modifies it by applying *mutation operators*,
  - feeds it to the tested program.
- Example mutation operators: bit flip, random byte value, set byte to *interesting* value, insert byte, delete byte, ...
- Optimization of fuzzing: finding an appropriate sequence of mutation operators (*mutation scheduling*).

Fuzzing

- The fuzzing loop contains a deterministic stage (uses a predefined order of mutations) and randomized stage (random choice of mutations).
- The optimization is applied to the randomized stage, with the goal of finding the effective. *probability distribution* of mutation operators
- An example of *online* learning: optimization is performed concurrently with the process being optimized:
  - the mutation scheduler uses the current, solution (probability distribution) to select mutation operators,
  - the reported feedback is used to optimize the distribution,
  - the updated probability distribution is applied in the next iteration.
- Optimization is made under uncertainty: the fitness of a solution may only be estimated.

Fuzzing

- Examples including evolutionary optimization in mutation-based fuzzing:
- MOPT fuzzer: uses a variant of PSO to learn globally optimal mutation probability distribution
  - heavily dependent on choice of parameters and the tested program
  - may exhibit slow convergence due to multiple solutions in the population
- An approach with single-solution metaheuristic (e.g. $(\mu + \lambda)$ ES) could increase effectiveness:
  - focuses on increasing the speed of convergence,
  - more robust over different target programs.
- EC-based mutation schedulers outperform standardized fuzzing platforms.
- References: [193, 51, 201, 174, 48, 92, 86, 195]

Conclusions - EC

- Up to now, EC proved to be successful in cryptography and cybersecurity.
- EC is used:
  1. When there exist no other, specialized approaches.
  2. To rapidly check whether some concept (e.g., formula) is correct.
  3. To assess the quality of some other method.
  4. To produce “good-enough” solutions.
  5. To produce novel and human-competitive solutions (solutions produced by EC that can rival the best solutions created by humans).

Conclusions - ML

- Machine learning is a data-driven approach, and as such limited to scenarios where we can obtain data.
- In cryptography, in domains like attacks on strong PUFs and profiled SCA, machine learning achieved excellent results.
- In cybersecurity, the applications are more diverse: intrusion detection and fuzzing are examples of well-explored and active research domains.
- While this tutorial covers topics that could be considered as AI for security, security of AI is also very active domain!

Conclusions

- We presented here only a handful of applications, there are more options.
- Even for each of the applications, there is a plethora of options still to try:
  1. New algorithms.
  2. Combinations of parameters.
  3. Representations.
  4. Fitness functions.
- The results obtained up to now are good, but there is still much room for improvement.

Conclusions

- As already said, there is a difference between attack and constructive perspectives, which makes it often easier for AI techniques to be used as attack mechanisms (at least for now).
- As a general research direction, it would be interesting to consider new applications (while some of the current ones are still very active, it would not be necessarily easy for new researchers to join the research effort).

Thank you for your attention!

Questions?

References I

Martín Abadi and David G. Andersen.
Learning to protect communications with adversarial neural cryptography.
CoRR, abs/1610.06918, 2016.

Hernán Aguirre, Hiroyuki Okazaki, and Yasushi Fuwa.
An evolutionary multiobjective approach to design highly non-linear boolean functions.
In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, page 749–756, New York, NY, USA, 2007. Association for Computing Machinery.

Anubhab Baksi, Jakub Breier, Yi Chen, and Xiaoyang Dong.
Machine learning assisted differential distinguishers for lightweight ciphers (extended version).
Cryptology ePrint Archive, Report 2020/571, 2020.
https://eprint.iacr.org/2020/571.

Timo Bartkewitz and Kerstin Lemke-Rust.
Efficient template attacks based on probabilistic multi-class support vector machines.
In Stefan Mangard, editor, *Smart Card Research and Advanced Applications*, pages 263–276, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

Lawrence E. Bassham, Andrew L. Rukhin, Juan Soto, James R. Nechvatal, Miles E. Smid, Elaine B. Barker, Stefan D. Leigh, Mark Levenson, Mark Vangel, David L. Banks, Nathanael Alan Heckert, James F. Dray, and San Vo.
Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications.
Technical report, Gaithersburg, MD, USA, 2010.

Lejla Batina, Domagoj Jakobovic, Nele Mentens, Stjepan Picek, Antonio de la Piedra, and Dominik Sisejkovic.
S-box pipelining using genetic algorithms for high-throughput aes implementations: How fast can we go?
In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology – INDOCRYPT 2014*, pages 322–337, Cham, 2014. Springer International Publishing.

References II

G. T. Becker.
On the pitfalls of using arbiter-pufs as building blocks.
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 34(8):1295–1307, 2015.

Georg T. Becker.
The gap between promise and reality: On the insecurity of xor arbiter pufs.
In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems – CHES 2015*, pages 535–555, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

Emanuele Bellini and Matteo Rossi.
Performance comparison between deep learning-based and conventional cryptographic distinguishers.
Cryptology ePrint Archive, Report 2020/953, 2020.
https://eprint.iacr.org/2020/953.

Adrien Benamira, David Gerault, Thomas Peyrin, and Quan Quan Tan.
A deeper look at machine learning-based cryptanalysis.
Cryptology ePrint Archive, Report 2021/287, 2021.
https://eprint.iacr.org/2021/287.

Hans-Georg Beyer and Hans-Paul Schwefel.
Evolution strategies –a comprehensive introduction.
Natural Computing: An International Journal, 1(1):3–52, May 2002.

Shivam Bhasin, Anupam Chattopadhyay, Annelie Heuser, Dirmanto Jap, Stjepan Picek, and Ritu Ranjan Shrivastwa.
Mind the portability: A warriors guide through realistic profiled side-channel analysis.
In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

Christopher M. Bishop.
Pattern Recognition and Machine Learning (Information Science and Statistics).
Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

References III

T. Bowen, A. Poylisher, C. Serban, R. Chadha, C. Y. Jason Chiang, and L. M. Marvel.
Enabling reproducible cyber research - four labeled datasets.
In *MILCOM 2016 - 2016 IEEE Military Communications Conference*, pages 539–544, Nov 2016.

Linda Burnett, W Millan, Edward Dawson, and A Clark.
Simpler methods for generating better boolean functions with good cryptographic properties.
Australas. J. Combin., 29:231–248, 2004.

Linda Dee Burnett.
Heuristic Optimization of Boolean Functions and Substitution Boxes for Cryptography.
PhD thesis, Queensland University of Technology, 2005.

Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff.
Convolutional Neural Networks with Data Augmentation Against Jitter-Based Countermeasures - Profiling Attacks Without Pre-processing.
In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 45–68, 2017.

Van Loi Cao, Miguel Nicolau, and James McDermott.
One-Class Classification for Anomaly Detection with Kernel Density Estimation and Genetic Programming.
In *Genetic Programming - 19th European Conference, EuroGP 2016, Porto, Portugal, March 30 - April 1, 2016, Proceedings*, pages 3–18, 2016.

Claude Carlet.
Boolean Functions for Cryptography and Coding Theory.
Cambridge University Press, 2021.

Claude Carlet and Sylvain Guilley.
Correlation-immune Boolean functions for easing counter measures to side-channel attacks, pages 41 – 70.
De Gruyter, Berlin, Boston, 2014.

References IV

Claude Carlet, Annelie Heuser, and Stjepan Picek.
Trade-offs for s-boxes: Cryptographic properties and side-channel resilience.
In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *Applied Cryptography and Network Security*, pages 393–414, Cham, 2017. Springer International Publishing.

Rafael Boix Carpi, Stjepan Picek, Lejla Batina, Federico Menarini, Domagoj Jakobovic, and Marin Golub.
Glitch it if you can: Parameter search strategies for successful fault injection.
In Aurélien Francillon and Pankaj Rohatgi, editors, *Smart Card Research and Advanced Applications*, pages 236–252, Cham, 2014. Springer International Publishing.

Jung-Wei Chou, Shou-De Lin, and Chen-Mou Cheng.
On the effectiveness of using state-of-the-art machine learning techniques to launch cryptographic distinguishing attacks.
In *Proceedings of the 5th ACM Workshop on Security and Artificial Intelligence*, AISec '12, page 105–110, New York, NY, USA, 2012. Association for Computing Machinery.

Andrew J. Clark.
Optimisation heuristics for cryptology.
PhD thesis, Queensland University of Technology, 1998.

J. A. Clark, J. L. Jacob, S. Maitra, and P. Stanica.
Almost boolean functions: the design of boolean functions by spectral inversion.
In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 3, pages 2173–2180 Vol.3, 2003.

J. A. Clark, J. L. Jacob, and S. Stepney.
The design of s-boxes by simulated annealing.
In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, pages 1533–1537 Vol.2, 2004.

References V

John A. Clark and Jeremy L. Jacob.
Two-stage optimisation in the design of boolean functions.
In E. P. Dawson, A. Clark, and Colin Boyd, editors, *Information Security and Privacy*, pages 242–254, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

John A. Clark, Jeremy L. Jacob, Susan Stepney, Subhamoy Maitra, and William Millan.
Evolving boolean functions satisfying multiple criteria.
In Alfred Menezes and Palash Sarkar, editors, *Progress in Cryptology — INDOCRYPT 2002*, pages 246–259, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

N. Cruz-Cortes, F. Rodriguez-Henriquez, and C. A. Coello Coello.
An artificial immune system heuristic for generating short addition chains.
IEEE Transactions on Evolutionary Computation, 12(1):1–24, 2008.

Nareli Cruz-Cortés, Francisco Rodríguez-Henríquez, Raúl Juárez-Morales, and Carlos A. Coello Coello.
Finding optimal addition chains using a genetic algorithm approach.
In Yue Hao, Jiming Liu, Yuping Wang, Yiu-ming Cheung, Hujun Yin, Licheng Jiao, Jianfeng Ma, and Yong-Chang Jiao, editors, *Computational Intelligence and Security*, pages 208–215, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

Thomas W. Cusick and Pantelimon Stănică.
Cryptographic Boolean Functions and Applications.
Elsevier Inc., San Diego, USA, 2009.

M. Danziger and M. A. A. Henriques.
Improved cryptanalysis combining differential and artificial neural network schemes.
In *2014 International Telecommunications Symposium (ITS)*, pages 1–5, 2014.

Flávio Luis de Mello and José A. M. Xexéo.
Identifying encryption algorithms in ECB and CBC modes using computational intelligence.
J. UCS, 24(1):25–42, 2018.

References VI

J. Delvaux.
Machine-learning attacks on polypufs, ob-pufs, rpufs, lhs-pufs, and puf–fsms.
IEEE Transactions on Information Forensics and Security, 14(8):2043–2058, 2019.

Marko Djurasevic, Domagoj Jakobovic, and Stjepan Picek.
One property to rule them all? on the limits of trade-offs for s-boxes.
In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 1064–1072, New York, NY, USA, 2020. Association for Computing Machinery.

Saúl Domínguez-Isidro, Efrén Mezura-Montes, and Luis Guillermo Osorio-Hernández.
Addition chain length minimization with evolutionary programming.
In *13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, Companion Material Proceedings, Dublin, Ireland, July 12-16, 2011*, pages 59–60, 2011.

Saúl Domínguez-Isidro, Efrén Mezura-Montes, and Luis Guillermo Osorio-Hernández.
Evolutionary programming for the length minimization of addition chains.
Eng. Appl. of AI, 37:125–134, 2015.

Kamil Dworak and Urszula Boryczka.
Cryptanalysis of sdes using modified version of binary particle swarm optimization.
In Manuel Núñez, Ngoc Thanh Nguyen, David Camacho, and Bogdan Trawiński, editors, *Computational Collective Intelligence*, pages 159–168, Cham, 2015. Springer International Publishing.

B. Ege, K. Papagiannopoulos, L. Batina, and S. Picek.
Improving dpa resistance of s-boxes: How far can we go?
In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2013–2016, 2015.

A. E. Eiben and James E. Smith.
Introduction to Evolutionary Computing.
Springer Publishing Company, Incorporated, 2nd edition, 2015.

References VII

S. Elsayed, R. Sarker, and J. Slay.
Evaluating the performance of a differential evolution algorithm in anomaly detection.
In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2490–2497, May 2015.

Gianluigi Folino, Clara Pizzuti, and Giandomenico Spezzano.
Gp ensemble for distributed intrusion detection systems.
In Sameer Singh, Maneesha Singh, Chid Apte, and Petra Perner, editors, *Pattern Recognition and Data Mining: Third International Conference on Advances in Pattern Recognition, ICAPR 2005, Bath, UK, August 22-25, 2005, Proceedings, Part I*, pages 54–62, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

J. Fuller, W. Millan, and E. Dawson.
Multi-objective optimisation of bijective s-boxes.
In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, pages 1525–1532 Vol.2, 2004.

Fatemeh Ganji, Shahin Tajik, Pascal Stauss, Jean-Pierre Seifert, Domenic Forte, and Mark Tehranipoor.
Rock'n'roll pufs: Crafting provably secure pufs from less secure ones.
In Karine Heydemann, Ulrich K\"uhne, and Letitia Li, editors, *Proceedings of 8th International Workshop on Security Proofs for Embedded Systems*, volume 11 of *Kalpa Publications in Computing*, pages 33–48. EasyChair, 2019.

Samaneh Ghandali, Georg T. Becker, Daniel Holcomb, and Christof Paar.
A design methodology for stealthy parametric trojans and its application to bug attacks.
In Benedikt Gierlichs and Axel Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems – CHES 2016*, pages 625–647, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

Ashrujit Ghoshal, Rajat Sadhukhan, Sikhar Patranabis, Nilanjan Datta, Stjepan Picek, and Debdeep Mukhopadhyay.
Lightweight and side-channel secure 4×4 s-boxes from cellular automata rules.
IACR Transactions on Symmetric Cryptology, 2018(3):311–334, Sep. 2018.

References VIII

R. Gilmore, N. Hanley, and M. O'Neill.
Neural network based attack on a masked implementation of AES.
In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 106–111, May 2015.

Patrice Godefroid, Hila Peleg, and Rishabh Singh.
Learn&fuzz: Machine learning for input fuzzing.
In *Proceedings of the 32Nd IEEE/ACM International Conference on Automated Software Engineering*, 2017.

Aron Gohr.
Improving attacks on round-reduced speck32/64 using deep learning.
In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 150–179, Cham, 2019. Springer International Publishing.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville.
Deep Learning.
MIT Press, 2016.
http://www.deeplearningbook.org.

Google.
Fuzzbench: 2020-09-28 report.
https://www.fuzzbench.com/reports/2020-09-28/index.html.
Accessed: 2020-09-28.

A. Habibi and et al.
Unb iscx nsl-kdd dataset, January 2017.

K. Hasegawa, Y. Shi, and N. Togawa.
Hardware trojan detection utilizing machine learning approaches.
In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1891–1896, 2018.

References IX

J. C. Hernandez, A. Seznec, and P. Isasi.
On the design of state-of-the-art pseudorandom number generators by means of genetic programming.
In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, pages 1510–1516 Vol.2, 2004.

Benjamin Hettwer, Stefan Gehrer, and Tim Güneysu.
Profiled power analysis attacks using convolutional neural networks with domain knowledge.
In Carlos Cid and Michael J. Jacobson Jr., editors, *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, volume 11349 of *Lecture Notes in Computer Science*, pages 479–498. Springer, 2018.

A. Heuser, S. Picek, S. Guilley, and N. Mentens.
Lightweight ciphers and their side-channel resilience.
IEEE Transactions on Computers, PP(99):1–1, 2017.

Annelie Heuser, Stjepan Picek, Sylvain Guilley, and Nele Mentens.
Side-channel analysis of lightweight ciphers: Does lightweight equal easy?
In *Radio Frequency Identification and IoT Security - 12th International Workshop, RFIDSec 2016, Hong Kong, China, November 30 - December 2, 2016, Revised Selected Papers*, pages 91–104, 2016.

Annelie Heuser and Michael Zohner.
Intelligent Machine Homicide - Breaking Cryptographic Devices Using Support Vector Machines.
In Werner Schindler and Sorin A. Huss, editors, *COSADE*, volume 7275 of *LNCS*, pages 249–264. Springer, 2012.

Anh-Tuan Hoang, Neil Hanley, and Maire O'Neill.
Plaintext: A missing feature for enhancing the power of deep learning in side-channel analysis? breaking multiple layers of side-channel countermeasures.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(4):49–85, Aug. 2020.

References X

John H. Holland.
Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.
MIT Press, Cambridge, MA, USA, 1992.

G Hospodar, E De Mulder, and B Gierlichs.
Least squares support vector machines for side-channel analysis.
Center for Advanced Security Research Darmstadt, pages 99–104, 01 2011.

Gabriel Hospodar, Benedikt Gierlichs, Elke De Mulder, Ingrid Verbauwhede, and Joos Vandewalle.
Machine learning in side-channel analysis: a first study.
Journal of Cryptographic Engineering, 1:293–302, 2011.
10.1007/s13389-011-0023-x.

Zezhou Hou, Jiongjiong Ren, and Shaozhen Chen.
Cryptanalysis of round-reduced simon32 based on deep learning.
Cryptology ePrint Archive, Report 2021/362, 2021.
https://eprint.iacr.org/2021/362.

Radek Hrbacek and Vaclav Dvorak.
Bent function synthesis by means of cartesian genetic programming.
In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII*, pages 414–423, Cham, 2014. Springer International Publishing.

Mohamed Fadl Idris, Je Sen Teh, Jasy Liew Suet Yan, and Wei-Zhu Yeoh.
A deep learning approach for active s-box prediction of generalized feistel lightweight block ciphers.
Cryptology ePrint Archive, Report 2021/066, 2021.
https://eprint.iacr.org/2021/066.

References XI

Georgi Ivanov, Nikolay Nikolov, and Svetla Nikova.
Cryptographically strong s-boxes generated by modified immune algorithm.
In Enes Pasalic and Lars R. Knudsen, editors, *Cryptography and Information Security in the Balkans*, pages 31–42, Cham, 2016. Springer International Publishing.

Georgi Ivanov, Nikolay Nikolov, and Svetla Nikova.
Reversed genetic algorithms for generation of bijective s-boxes with good cryptographic properties.
Cryptography Commun., 8(2):247–276, April 2016.

T. Iwase, Y. Nozaki, M. Yoshikawa, and T. Kumaki.
Detection technique for hardware trojans using machine learning in frequency domain.
In *2015 IEEE 4th Global Conference on Consumer Electronics (GCCE)*, pages 185–186, 2015.

Domagoj Jakobovic, Stjepan Picek, Marcella S. R. Martins, and Markus Wagner.
A characterisation of s-box fitness landscapes in cryptography.
In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, page 285–293, New York, NY, USA, 2019. Association for Computing Machinery.

Domagoj Jakobovic, Stjepan Picek, Marcella S.R. Martins, and Markus Wagner.
Toward more efficient heuristic construction of boolean functions.
Applied Soft Computing, 107:107327, 2021.

N. Karimian, F. Tehranipoor, M. T. Rahman, S. Kelly, and D. Forte.
Genetic algorithm for hardware trojan detection with ring oscillator network (ron).
In *2015 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–6, 2015.

Jonathan Katz and Yehuda Lindell.
Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series).
Chapman & Hall/CRC, 2007.

References XII

Jaehun Kim, Stjepan Picek, Annelie Heuser, Shivam Bhasin, and Alan Hanjalic.
Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2019(3):148–179, May 2019.

Hayato Kimura, Keita Emura, Takanori Isobe, Ryoma Ito, Kazuto Ogawa, and Toshihiro Ohigashi.
Output prediction attacks on spn block ciphers using deep learning.
Cryptology ePrint Archive, Report 2021/401, 2021.
https://eprint.iacr.org/2021/401.

Lars R. Knudsen and Matthew J. B. Robshaw.
The Block Cipher Companion.
Springer Publishing Company, Incorporated, 2011.

John Koza.
Evolving a computer program to generate random numbers using the genetic programming paradigm.
In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 37–44. Morgan Kaufmann, 1991.

John R. Koza.
Genetic Programming: On the Programming of Computers by Means of Natural Selection.
MIT Press, Cambridge, MA, USA, 1992.

W. O. Krawec.
A genetic algorithm to analyze the security of quantum cryptographic protocols.
In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 2098–2105, 2016.

Walter Krawec, Stjepan Picek, and Domagoj Jakobovic.
Evolutionary algorithms for the design of quantum protocols.
In Paul Kaufmann and Pedro A. Castillo, editors, *Applications of Evolutionary Computation*, pages 220–236, Cham, 2019. Springer International Publishing.

References XIII

Walter O. Krawec, Michael G. Nelson, and Eric P. Geiss.
Automatic generation of optimal quantum key distribution protocols.
In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, page 1153–1160, New York, NY, USA, 2017. Association for Computing Machinery.

A. Kulkarni, Y. Pino, and T. Mohsenin.
Adaptive real-time trojan detection framework through machine learning.
In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 120–123, 2016.

Mansoureh Labafniya, Stjepan Picek, Shahram [Etemadi Borujeni], and Nele Mentens.
On the feasibility of using evolvable hardware for hardware trojan detection and prevention.
Applied Soft Computing, 91:106247, 2020.

Linus Lagerhjelm.
Extracting information from encrypted data using deep neural networks.
Master's thesis, Umeå University, Department of Applied Physics and Electronics, 2018.

Carlos Lamenca-Martinez, Julio Cesar Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda.
Lamar: A new pseudorandom number generator evolved by means of genetic programming.
In Thomas Philip Runarsson, Hans-Georg Beyer, Edmund Burke, Juan J. Merelo-Guervós, L. Darrell Whitley, and Xin Yao, editors, *Parallel Problem Solving from Nature - PPSN IX*, pages 850–859, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

E. C. Laskari, G. C. Meletiou, Y. C. Stamatiou, and M. N. Vrahatis.
Cryptography and cryptanalysis through computational intelligence.
In Nadia Nedjah, Ajith Abraham, and Luiza de Macedo Mourelle, editors, *Computational Intelligence in Information Assurance and Security*, pages 1–49, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

Caroline Lemieux and Koushik Sen.
Fairfuzz: A targeted mutation strategy for greybox fuzz testing coverage.
In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 475–485, 2018.

References XIV

Alejandro León-Javier, Nareli Cruz-Cortés, Marco A. Moreno-Armendáriz, and Sandra Orantes-Jiménez.
Finding minimal addition chains with a particle swarm optimization algorithm.
In Arturo Hernández Aguirre, Raúl Monroy Borja, and Carlos Alberto Reyes Garciá, editors, *MICAI 2009: Advances in Artificial Intelligence*, pages 680–691, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

Liran Lerman, Romain Poussier, Gianluca Bontempi, Olivier Markowitch, and François-Xavier Standaert.
Template Attacks vs. Machine Learning Revisited (and the Curse of Dimensionality in Side-Channel Analysis).
In *COSADE 2015, Berlin, Germany, 2015. Revised Selected Papers*, pages 20–33, 2015.

Liran Lerman, Nikita Veshchikov, Stjepan Picek, and Olivier Markowitch.
Higher order side-channel attack resilient s-boxes.
In *Proceedings of the 15th ACM International Conference on Computing Frontiers*, CF '18, page 336–341, New York, NY, USA, 2018. Association for Computing Machinery.

Huimin Li, Marina Krcek, and Guilherme Perin.
A comparison of weight initializers in deep learning-based side-channel analysis.
IACR Cryptol. ePrint Arch., 2020:904, 2020.

Konstantinos G Liakos, Georgios K Georgakilas, Serafeim Moustakidis, Patrik Karlsson, and Fotis C Plessas.
Machine learning for hardware trojan detection: A review.
In *2019 Panhellenic Conference on Electronics Telecommunications (PACET)*, pages 1–6, 2019.

Chenyang Lyu, Shouling Ji, Chao Zhang, Yuwei Li, Wei-Han Lee, Yu Song, and Raheem Beyah.
MOPT: Optimized mutation scheduling for fuzzers.
In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1949–1966, 2019.

Houssem Maghrebi, Thibault Portigliatti, and Emmanuel Prouff.
Breaking cryptographic implementations using deep learning techniques.
In *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE 2016, Hyderabad, India, December 14-18, 2016, Proceedings*, pages 3–26, 2016.

References XV

A. Maldini, N. Samwel, S. Picek, and L. Batina.
Genetic algorithm-based electromagnetic fault injection.
In *2018 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 35–42, 2018.

Antun Maldini, Niels Samwel, Stjepan Picek, and Lejla Batina.
Optimizing electromagnetic fault injection with genetic algorithms.
In Jakub Breier, Xiaolu Hou, and Shivam Bhasin, editors, *Automated Methods in Cryptographic Fault Analysis*, pages 281–300, Cham, 2019. Springer International Publishing.

Stefan Mangard, Elisabeth Oswald, and Thomas Popp.
Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security).
Springer-Verlag, Berlin, Heidelberg, 2007.

Luca Manzoni, Luca Mariot, and Eva Tuba.
Does constraining the search space of ga always help? the case of balanced crossover operators.
In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 151–152, New York, NY, USA, 2019. Association for Computing Machinery.

Luca Manzoni, Luca Mariot, and Eva Tuba.
Balanced crossover operators in genetic algorithms.
Swarm and Evolutionary Computation, 54:100646, 2020.

Luca Mariot, Domagoj Jakobovic, Alberto Leporati, and Stjepan Picek.
Hyper-bent boolean functions and evolutionary algorithms.
In Lukas Sekanina, Ting Hu, Nuno Lourenço, Hendrik Richter, and Pablo García-Sánchez, editors, *Genetic Programming*, pages 262–277, Cham, 2019. Springer International Publishing.

Luca Mariot and Alberto Leporati.
A genetic algorithm for evolving plateaued cryptographic boolean functions.
In Adrian-Horia Dediu, Luis Magdalena, and Carlos Martín-Vide, editors, *Theory and Practice of Natural Computing*, pages 33–45, Cham, 2015. Springer International Publishing.

References XVI

Luca Mariot and Alberto Leporati.
Heuristic search by particle swarm optimization of boolean functions for cryptographic applications.
In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, page 1425–1426, New York, NY, USA, 2015. Association for Computing Machinery.

Luca Mariot, Stjepan Picek, Domagoj Jakobovic, and Alberto Leporati.
Evolutionary search of binary orthogonal arrays.
In Anne Auger, Carlos M. Fonseca, Nuno Lourenço, Penousal Machado, Luís Paquete, and Darrell Whitley, editors, *Parallel Problem Solving from Nature – PPSN XV*, pages 121–133, Cham, 2018. Springer International Publishing.

Luca Mariot, Stjepan Picek, Domagoj Jakobovic, and Alberto Leporati.
An evolutionary view on reversible shift-invariant transformations.
In Ting Hu, Nuno Lourenço, Eric Medvet, and Federico Divina, editors, *Genetic Programming*, pages 118–134, Cham, 2020. Springer International Publishing.

Luca Mariot, Stjepan Picek, Alberto Leporati, and Domagoj Jakobovic.
Cellular automata based s-boxes.
Cryptography and Communications, 11(1):41–62, 2019.

Loïc Masure, Cécile Dumas, and Emmanuel Prouff.
A comprehensive study of deep learning for side-channel analysis.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(1):348–375, Nov. 2019.

Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone.
Handbook of Applied Cryptography.
CRC Press, 2001.

W. Millan, L. Burnett, G. Carter, A. Clark, and E. Dawson.
Evolutionary heuristics for finding cryptographically strong s-boxes.
In Vijay Varadharajan and Yi Mu, editors, *Information and Communication Security*, pages 263–274, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

References XVII

W. Millan, J. Fuller, and E. Dawson.
New concepts in evolutionary search for boolean functions in cryptology.
In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, volume 3, pages 2157–2164 Vol.3, 2003.

William Millan, Andrew Clark, and Ed Dawson.
An effective genetic algorithm for finding highly nonlinear boolean functions.
In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *Information and Communications Security*, pages 149–158, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

William Millan, Andrew Clark, and Ed Dawson.
Heuristic design of cryptographically strong balanced boolean functions.
In Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, pages 489–499, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

William Millan, Andrew Clark, and Ed Dawson.
Boolean function design using hill climbing methods.
In Josef Pieprzyk, Rei Safavi-Naini, and Jennifer Seberry, editors, *Information Security and Privacy*, pages 1–11, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

Julian F. Miller.
An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach.
In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2*, GECCO'99, page 1135–1142, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

Julian F. Miller.
Cartesian genetic programming.
In Julian F. Miller, editor, *Cartesian Genetic Programming*, pages 17–34, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

Thomas M. Mitchell.
Machine Learning.
McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

References XVIII

Nadia Nedjah and Luiza de Macedo Mourelle.
Minimal addition chain for efficient modular exponentiation using genetic algorithms.
In Tim Hendtlass and Moonis Ali, editors, *Developments in Applied Artificial Intelligence*, pages 88–98, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

Nadia Nedjah and Luiza de Macedo Mourelle.
Minimal addition-subtraction chains using genetic algorithms.
In Tatyana Yakhno, editor, *Advances in Information Systems*, pages 303–313, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

Nadia Nedjah and Luiza de Macedo Mourelle.
Minimal Addition-Subtraction Sequences for Efficient Pre-processing in Large Window-Based Modular Exponentiation Using Genetic Algorithms.
In Jiming Liu, Yiu-ming Cheung, and Hujun Yin, editors, *Intelligent Data Engineering and Automated Learning*, volume 2690 of *Lect. Notes in Comp. Science*, pages 329–336. Springer, 2003.

Nadia Nedjah and Luiza de Macedo Mourelle.
Finding minimal addition chains using ant colony.
In Zheng Rong Yang, Hujun Yin, and Richard M. Everson, editors, *Intelligent Data Engineering and Automated Learning – IDEAL 2004*, pages 642–647, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

Nadia Nedjah and Luiza de Macedo Mourelle.
High-performance SoC-based Implementation of Modular Exponentiation Using Evolutionary Addition Chains for Efficient Cryptography.
Applied Soft Computing, 11(7):4302–4311, October 2011.

David Oranchak.
Evolutionary algorithm for decryption of monoalphabetic homophonic substitution ciphers encoded as constraint satisfaction problems.
In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, page 1717–1718, New York, NY, USA, 2008. Association for Computing Machinery.

References XIX

Agustin Orfila, Juan M. Estevez-Tapiador, and Arturo Ribagorda.
Evolving high-speed, easy-to-understand network intrusion detection rules with genetic programming.
In Mario Giacobini, Anthony Brabazon, Stefano Cagnoni, Gianni A. Di Caro, Anikó Ekárt, Anna Isabel Esparcia-Alcázar, Muddassar Farooq, Andreas Fink, and Penousal Machado, editors, *Applications of Evolutionary Computing: EvoWorkshops 2009: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, EvoNUM, EvoSTOC, EvoTRANSLOG, Tübingen, Germany, April 15-17, 2009. Proceedings*, pages 93–98, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

L. G. Osorio-Hernandez, E. Mezura-Montes, N. Cruz-Cortes, and F. Rodriguez-Henriquez.
A genetic algorithm with repair and local search mechanisms able to find minimal length addition chains for small exponents.
In *2009 IEEE Congress on Evolutionary Computation*, pages 1422–1429, 2009.

Artem Pavlenko, Alexander Semenov, and Vladimir Ulyantsev.
Evolutionary computation techniques for constructing sat-based attacks in algebraic cryptanalysis.
In Paul Kaufmann and Pedro A. Castillo, editors, *Applications of Evolutionary Computation*, pages 237–253, Cham, 2019. Springer International Publishing.

Guilherme Perin, Łukasz Chmielewski, Lejla Batina, and Stjepan Picek.
Keep it unsupervised: Horizontal attacks meet deep learning.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(1):343–372, Dec. 2020.

Guilherme Perin, Łukasz Chmielewski, and Stjepan Picek.
Strength in numbers: Improving generalization with ensembles in machine learning-based profiled side-channel analysis.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(4):337–364, Aug. 2020.

Guilherme Perin, Lichao Wu, and Stjepan Picek.
Aisy - deep learning-based framework for side-channel analysis.
Cryptology ePrint Archive, Report 2021/357, 2021.
https://eprint.iacr.org/2021/357.

References XX

Guilherme Perin, Lichao Wu, and Stjepan Picek.
Gambling for success: The lottery ticket hypothesis in deep learning-based sca.
Cryptology ePrint Archive, Report 2021/197, 2021.
https://eprint.iacr.org/2021/197.

Pedro Peris-Lopez, Julio Cesar Hernandez-Castro, Juan M. Estevez-Tapiador, and Arturo Ribagorda.
Lamed - a prng for epc class-1 generation-2 rfid specification.
Comput. Stand. Interfaces, 31(1):88–97, January 2009.

S. Picek, L. Batina, D. Jakobović, and R. B. Carpi.
Evolving genetic algorithms for fault injection attacks.
In *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1106–1111, 2014.

S. Picek, B. Ege, K. Papagiannopoulos, L. Batina, and D. Jakobović.
Optimality and beyond: The case of 4×4 s-boxes.
In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 80–83, 2014.

S. Picek, A. Heuser, A. Jovic, and L. Batina.
A systematic evaluation of profiling through focused feature selection.
IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 27(12):2802–2815, 2019.

S. Picek, K. Knezevic, and D. Jakobovic.
On the evolution of bent (n, m) functions.
In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pages 2137–2144, 2017.

S. Picek, K. Knezevic, D. Jakobovic, and C. Carlet.
A search for differentially-6 uniform (n, n-2) functions.
In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–7, 2018.

References XXI

S. Picek, K. Knezevic, L. Mariot, D. Jakobovic, and A. Leporati.
Evolving bent quaternary functions.
In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2018.

Stjepan Picek.
Applications of evolutionary computation to cryptography.
PhD thesis, Radboud University Nijmegen, The Netherlands, 2015.

Stjepan Picek.
Evolutionary computation and cryptology.
In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, GECCO '16 Companion, page 883–909, New York, NY, USA, 2016. Association for Computing Machinery.

Stjepan Picek.
Applications of soft computing in cryptology.
In Dooho Choi and Sylvain Guilley, editors, *Information Security Applications*, pages 305–317, Cham, 2017. Springer International Publishing.

Stjepan Picek.
Challenges in deep learning-based profiled side-channel analysis.
In Shivam Bhasin, Avi Mendelson, and Mridul Nandi, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 9–12, Cham, 2019. Springer International Publishing.

Stjepan Picek, Lejla Batina, Pieter Buzing, and Domagoj Jakobovic.
Fault injection with a new flavor: Memetic algorithms make a difference.
In Stefan Mangard and Axel Y. Poschmann, editors, *Constructive Side-Channel Analysis and Secure Design*, pages 159–173, Cham, 2015. Springer International Publishing.

References XXII

Stjepan Picek, Lejla Batina, and Domagoj Jakobovic.
Evolving dpa-resistant boolean functions.
In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII*, pages 812–821, Cham, 2014. Springer International Publishing.

Stjepan Picek, Lejla Batina, Domagoj Jaković, Bariş Ege, and Marin Golub.
S-box, set, match: A toolbox for s-box analysis.
In David Naccache and Damien Sauveron, editors, *Information Security Theory and Practice. Securing the Internet of Things*, pages 140–149, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

Stjepan Picek, Claude Carlet, Sylvain Guilley, Julian F. Miller, and Domagoj Jakobovic.
Evolutionary algorithms for boolean functions in diverse domains of cryptography.
Evolutionary Computation, 24(4):667–694, 2016.

Stjepan Picek, Claude Carlet, Domagoj Jakobovic, Julian F. Miller, and Lejla Batina.
Correlation immunity of boolean functions: An evolutionary algorithms perspective.
In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, page 1095–1102, New York, NY, USA, 2015. Association for Computing Machinery.

Stjepan Picek, Carlos A. Coello, Domagoj Jakobovic, and Nele Mentens.
Finding short and implementation-friendly addition chains with evolutionary algorithms.
Journal of Heuristics, 24(3):457–481, June 2018.

Stjepan Picek, Carlos A. Coello Coello, Domagoj Jakobovic, and Nele Mentens.
Evolutionary algorithms for finding short addition chains: Going the distance.
In Francisco Chicano, Bin Hu, and Pablo García-Sánchez, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 121–137, Cham, 2016. Springer International Publishing.

Stjepan Picek, Marko Cupic, and Leon Rotim.
A new cost function for evolution of s-boxes.
Evolutionary Computation, 24(4):695–718, 2016.

References XXIII

Stjepan Picek, Bariş Ege, Lejla Batina, Domagoj Jakobovic, undefinedukasz Chmielewski, and Marin Golub.
On using genetic algorithms for intrinsic side-channel resistance: The case of aes s-box.
In *Proceedings of the First Workshop on Cryptography and Security in Computing Systems*, CS2 '14, page 13–18, New York, NY, USA, 2014. Association for Computing Machinery.

Stjepan Picek, Sylvain Guilley, Claude Carlet, Domagoj Jakobovic, and Julian F. Miller.
Evolutionary approach for finding correlation immune boolean functions of order t with minimal hamming weight.
In Adrian-Horia Dediu, Luis Magdalena, and Carlos Martín-Vide, editors, *Theory and Practice of Natural Computing*, pages 71–82, Cham, 2015. Springer International Publishing.

Stjepan Picek, Annelie Heuser, and Sylvain Guilley.
Template attack versus Bayes classifier.
Journal of Cryptographic Engineering, 7(4):343–351, Nov 2017.

Stjepan Picek, Annelie Heuser, Alan Jovic, Shivam Bhasin, and Francesco Regazzoni.
The curse of class imbalance and conflicting metrics with machine learning for side-channel evaluations.
IACR Trans. Cryptogr. Hardw. Embed. Syst., 2019(1):209–237, 2019.

Stjepan Picek, Annelie Heuser, Alan Jovic, Simone A. Ludwig, Sylvain Guilley, Domagoj Jakobovic, and Nele Mentens.
Side-channel analysis and machine learning: A practical perspective.
In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 4095–4102, 2017.

Stjepan Picek and Domagoj Jakobovic.
Evolving algebraic constructions for designing bent boolean functions.
In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, page 781–788, New York, NY, USA, 2016. Association for Computing Machinery.

References XXIV

Stjepan Picek and Domagoj Jakobovic.
On the design of s-box constructions with genetic programming.
In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 395–396, New York, NY, USA, 2019. Association for Computing Machinery.

Stjepan Picek, Domagoj Jakobovic, and Marin Golub.
Evolving cryptographically sound boolean functions.
In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '13 Companion, page 191–192, New York, NY, USA, 2013. Association for Computing Machinery.

Stjepan Picek, Domagoj Jakobovic, Julian F. Miller, Lejla Batina, and Marko Cupic.
Cryptographic boolean functions: One output, many design criteria.
Applied Soft Computing, 40:635 – 653, 2016.

Stjepan Picek, Domagoj Jakobovic, Julian F. Miller, Elena Marchiori, and Lejla Batina.
Evolutionary methods for the construction of cryptographic boolean functions.
In Penousal Machado, Malcolm I. Heywood, James McDermott, Mauro Castelli, Pablo García-Sánchez, Paolo Burelli, Sebastian Risi, and Kevin Sim, editors, *Genetic Programming*, pages 192–204, Cham, 2015. Springer International Publishing.

Stjepan Picek, Domagoj Jakobovic, and Una-May O'Reilly.
Cryptobench: Benchmarking evolutionary algorithms with cryptographic problems.
In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, page 1597–1604, New York, NY, USA, 2017. Association for Computing Machinery.

Stjepan Picek, Karlo Knezevic, Domagoj Jakobovic, and Ante Derek.
C3po: Cipher construction with cartesian genetic programming.
In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 1625–1633, New York, NY, USA, 2019. Association for Computing Machinery.

References XXV

Stjepan Picek, Elena Marchiori, Lejla Batina, and Domagoj Jakobovic.
Combining evolutionary computation and algebraic constructions to find cryptography-relevant boolean functions.
In Thomas Bartz-Beielstein, Jürgen Branke, Bogdan Filipič, and Jim Smith, editors, *Parallel Problem Solving from Nature – PPSN XIII*, pages 822–831, Cham, 2014. Springer International Publishing.

Stjepan Picek, Luca Mariot, Alberto Leporati, and Domagoj Jakobovic.
Evolving s-boxes based on cellular automata with genetic programming.
In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, page 251–252, New York, NY, USA, 2017. Association for Computing Machinery.

Stjepan Picek, Luca Mariot, Bohan Yang, Domagoj Jakobovic, and Nele Mentens.
Design of s-boxes defined with cellular automata rules.
In *Proceedings of the Computing Frontiers Conference*, CF'17, page 409–414, New York, NY, USA, 2017. Association for Computing Machinery.

Stjepan Picek, Bodhisatwa Mazumdar, Debdeep Mukhopadhyay, and Lejla Batina.
Modified transparency order property: Solution or just another attempt.
In Rajat Subhra Chakraborty, Peter Schwabe, and Jon Solworth, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 210–227, Cham, 2015. Springer International Publishing.

Stjepan Picek, Robert I. McKay, Roberto Santana, and Tom D. Gedeon.
Fighting the symmetries: The structure of cryptographic boolean function spaces.
In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, page 457–464, New York, NY, USA, 2015. Association for Computing Machinery.

Stjepan Picek, Julian F. Miller, Domagoj Jakobovic, and Lejla Batina.
Cartesian genetic programming approach for generating substitution boxes of different sizes.
In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, page 1457–1458, New York, NY, USA, 2015. Association for Computing Machinery.

References XXVI

Stjepan Picek, Kostas Papagiannopoulos, Barış Ege, Lejla Batina, and Domagoj Jakobovic.
Confused by confusion: Systematic evaluation of dpa resistance of various s-boxes.
In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology – INDOCRYPT 2014*, pages 374–390, Cham, 2014. Springer International Publishing.

Stjepan Picek, Ioannis Petros Samiotis, Jaehun Kim, Annelie Heuser, Shivam Bhasin, and Axel Legay.
On the performance of convolutional neural networks for side-channel analysis.
In Anupam Chattopadhyay, Chester Rebeiro, and Yuval Yarom, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 157–176, Cham, 2018. Springer International Publishing.

Stjepan Picek, Dominik Sisejkovic, and Domagoj Jakobovic.
Immunological algorithms paradigm for construction of boolean functions with good cryptographic properties.
Engineering Applications of Artificial Intelligence, 62:320 – 330, 2017.

Stjepan Picek, Dominik Sisejkovic, Domagoj Jakobovic, Lejla Batina, Bohan Yang, Danilo Sijacic, and Nele Mentens.
Extreme pipelining towards the best area-performance trade-off in hardware.
In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *Progress in Cryptology – AFRICACRYPT 2016*, pages 147–166, Cham, 2016. Springer International Publishing.

Stjepan Picek, Dominik Sisejkovic, Vladimir Rozic, Bohan Yang, Domagoj Jakobovic, and Nele Mentens.
Evolving cryptographic pseudorandom number generators.
In Julia Handl, Emma Hart, Peter R. Lewis, Manuel López-Ibáñez, Gabriela Ochoa, and Ben Paechter, editors, *Parallel Problem Solving from Nature – PPSN XIV*, pages 613–622, Cham, 2016. Springer International Publishing.

Stjepan Picek, Bohan Yang, Vladimir Rozic, and Nele Mentens.
On the construction of hardware-friendly 4×4 and 5×5 s-boxes.
In Roberto Avanzi and Howard Heys, editors, *Selected Areas in Cryptography – SAC 2016*, pages 161–179, Cham, 2017. Springer International Publishing.

References XXVII

Stjepan Picek, Bohan Yang, Vladimir Rozic, Jo Vliegen, Jori Winderickx, Thomas De Cnudde, and Nele Mentens.
Prngs for masking applications and their mapping to evolvable hardware.
In Kerstin Lemke-Rust and Michael Tunstall, editors, *Smart Card Research and Advanced Applications*, pages 209–227, Cham, 2017. Springer International Publishing.

A. Poorghanad, A. Sadr, and A. Kashanipour.
Generating high quality pseudo random number using evolutionary methods.
In *2008 International Conference on Computational Intelligence and Security*, volume 1, pages 331–335, 2008.

Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas.
Study of deep learning techniques for side-channel analysis and introduction to ASCAD database.
IACR Cryptology ePrint Archive, 2018:53, 2018.

Sanjay Rawat, Vivek Jain, Ashish Kumar, Lucian Cojocar, Cristiano Giuffrida, and Herbert Bos.
Vuzzer: Application-aware evolutionary fuzzing.
In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2017.

Jorai Rijsdijk, Lichao Wu, Guilherme Perin, and Stjepan Picek.
Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis.
Cryptology ePrint Archive, Report 2021/071, 2021.
https://eprint.iacr.org/2021/071.

Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber.
Modeling attacks on physical unclonable functions.
In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, page 237–249, New York, NY, USA, 2010. Association for Computing Machinery.

U. Rührmair, J. Sölter, F. Sehnke, X. Xu, A. Mahmoud, V. Stoyanova, G. Dror, J. Schmidhuber, W. Burleson, and S. Devadas.
Puf modeling attacks on simulated and silicon data.
IEEE Transactions on Information Forensics and Security, 8(11):1876–1891, 2013.

References XXVIII

U. Rührmair and M. van Dijk.
Pufs in security protocols: Attack models and security evaluations.
In *2013 IEEE Symposium on Security and Privacy*, pages 286–300, 2013.

Sayandeep Saha, Rajat Subhra Chakraborty, Srinivasa Shashank Nuthakki, Anshul, and Debdeep Mukhopadhyay.
Improved test pattern generation for hardware trojan detection using genetic algorithm and boolean satisfiability.
In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 577–596. Springer, 2015.

Sayandeep Saha, Dirmanto Jap, Sikhar Patranabis, Debdeep Mukhopadhyay, Shivam Bhasin, and Pallab Dasgupta.
Automatic characterization of exploitable faults: A machine learning approach.
IEEE Trans. Information Forensics and Security, 14(4):954–968, 2019.

Bruce Schneier.
Applied Cryptography (2nd Ed.): Protocols, Algorithms, and Source Code in C.
John Wiley & Sons, Inc., USA, 1995.

Lukáš Sekanina.
Virtual reconfigurable circuits for real-world applications of evolvable hardware.
In AAndy M. Tyrrell, Pauline C. Haddow, and Jim Torresen, editors, *Evolvable Systems: From Biology to Hardware*, pages 186–197, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

Kamran Shafi and Hussein A Abbass.
Biologically-inspired complex adaptive systems approaches to network intrusion detection.
information security technical report, 12(4):209–217, 2007.

Dong Song, Malcolm I. Heywood, and A. Nur Zincir-Heywood.
Training genetic programming on half a million patterns: an example from anomaly detection.
IEEE Trans. Evolutionary Computation, 9(3):225–239, 2005.

References XXIX

Lee Spector.
Automatic Quantum Computer Programming: A Genetic Programming Approach (Genetic Programming).
Springer-Verlag, Berlin, Heidelberg, 2006.

Emanuele Strieder, Christoph Frisch, and Michael Pehl.
Machine learning of physical unclonable functions using helper data: Revealing a pitfall in the fuzzy commitment scheme.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(2):1–36, Feb. 2021.

Petr Švenda, Martin Ukrop, and Vashek Matyáš.
Determining cryptographic distinguishers for estream and sha-3 candidate functions with evolutionary circuits.
In Mohammad S. Obaidat and Joaquim Filipe, editors, *E-Business and Telecommunications*, pages 290–305, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

Petr Tesař.
A New Method for Generating High Non-linearity S-Boxes.
Radioengineering, 19(1):23–26, April 2010.

Dhruv Thapar, Manaar Alam, and Debdeep Mukhopadhyay.
Transca: Cross-family profiled side-channel attacks using transfer learning on deep neural networks.
Cryptology ePrint Archive, Report 2020/1258, 2020.
https://eprint.iacr.org/2020/1258.

Johannes Tobisch, Anita Aghaie, and Georg T. Becker.
Combining optimization objectives: New modeling attacks on strong pufs.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(2):357–389, Feb. 2021.

Johannes Tobisch and Georg T. Becker.
On the scaling of machine learning attacks on pufs with application to noise bifurcation.
In Stefan Mangard and Patrick Schaumont, editors, *Radio Frequency Identification*, pages 17–31, Cham, 2015. Springer International Publishing.

References XXX

Daan van der Valk, Marina Krcek, Stjepan Picek, and Shivam Bhasin.
Learning from a big brother - mimicking neural networks in profiled side-channel analysis.
In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

Spandan Veggalam, Sanjay Rawat, Istvan Haller, and Herbert Bos.
Ifuzzer: An evolutionary interpreter fuzzer using genetic programming.
In *European Symposium on Research in Computer Security*, pages 581–601. Springer, 2016.

An Wang, Yuan Li, Yaoling Ding, Liehuang Zhu, and Yongjuan Wang.
Efficient framework for genetic-algorithm-based correlation power analysis.
Cryptology ePrint Archive, Report 2021/179, 2021.
https://eprint.iacr.org/2021/179.

Yunchao Wang, Zehui Wu, Qiang Wei, and Qingxian Wang.
Neufuzz: Efficient fuzzing with deep neural network.
IEEE Access, 7:36340–36352, 2019.

Léo Weissbart, Łukasz Chmielewski, Stjepan Picek, and Lejla Batina.
Systematic side-channel analysis of curve25519 with machine learning.
Journal of Hardware and Systems Security, 4(4):314–328, October 2020.

Léo Weissbart, Stjepan Picek, and Lejla Batina.
One trace is all it takes: Machine learning-based side-channel attack on eddsa.
In Shivam Bhasin, Avi Mendelson, and Mridul Nandi, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 86–105, Cham, 2019. Springer International Publishing.

Nils Wisiol, Georg T. Becker, Marian Margraf, Tudor A. A. Soroceanu, Johannes Tobisch, and Benjamin Zengin.
Breaking the lightweight secure puf: Understanding the relation of input transformations and machine learning resistance.
In Sonia Belaïd and Tim Güneysu, editors, *Smart Card Research and Advanced Applications*, pages 40–54, Cham, 2020. Springer International Publishing.

References XXXI

Nils Wisiol, Christopher Mühl, Niklas Pirnay, Phuong Ha Nguyen, Marian Margraf, Jean-Pierre Seifert, Marten van Dijk, and Ulrich Rührmair.
Splitting the interpose puf: A novel modeling attack strategy.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(3):97–120, Jun. 2020.

Stephen Wolfram.
Random sequence generation by cellular automata.
Adv. Appl. Math., 7(2):123–169, June 1986.

Maverick Woo, Sang Kil Cha, Samantha Gottlieb, and David Brumley.
Scheduling black-box mutational fuzzing.
In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 511–522, New York, NY, USA, 2013. ACM.

Lennert Wouters, Victor Arribas, Benedikt Gierlichs, and Bart Preneel.
Revisiting a methodology for efficient cnn architectures in profiling attacks.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(3):147–168, Jun. 2020.

Lichao Wu, Guilherme Perin, and Stjepan Picek.
I choose you: Automated hyperparameter tuning for deep learning-based side-channel analysis.
Cryptology ePrint Archive, Report 2020/1293, 2020.
https://eprint.iacr.org/2020/1293.

Lichao Wu and Stjepan Picek.
Remove some noise: On pre-processing of side-channel measurements with autoencoders.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(4):389–415, Aug. 2020.

Lichao Wu, Gerard Ribera, Noemie Beringuier-Boher, and Stjepan Picek.
A fast characterization method for semi-invasive fault injection attacks.
In Stanislaw Jarecki, editor, *Topics in Cryptology – CT-RSA 2020*, pages 146–170, Cham, 2020. Springer International Publishing.

References XXXII

Shelly Xiaonan Wu and Wolfgang Banzhaf.
Review: The use of computational intelligence in intrusion detection systems: A review.
Appl. Soft Comput., 10(1):1–35, January 2010.

Yang Yu, Michail Moraitis, and Elena Dubrova.
Profiled deep learning side-channel attack on a protected arbiter puf combined with bitstream modification.
Cryptology ePrint Archive, Report 2020/1031, 2020.
https://eprint.iacr.org/2020/1031.

Gabriel Zaid, Lilian Bossuet, François Dassance, Amaury Habrard, and Alexandre Venelli.
Ranking loss: Maximizing the success rate in deep learning side-channel analysis.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2021(1):25–55, Dec. 2020.

Gabriel Zaid, Lilian Bossuet, Amaury Habrard, and Alexandre Venelli.
Methodology for efficient cnn architectures in profiling attacks.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(1):1–36, Nov. 2019.

Jiajia Zhang, Mengce Zheng, Jiehui Nan, Honggang Hu, and Nenghai Yu.
A novel evaluation metric for deep learning-based side channel analysis and its extended application to imbalanced data.
IACR Transactions on Cryptographic Hardware and Embedded Systems, 2020(3):73–96, Jun. 2020.

Zhenbin Zhang, Liji Wu, An Wang, Zhaoli Mu, and Xiangmin Zhang.
A novel bit scalable leakage model based on genetic algorithm.
Security and Communication Networks, 8(18):3896–3905, 2015.