Tutorial on Boolean Functions and Artificial Intelligence

Stjepan Picek

BFA 2024, September 13, 2024

Outline

Outline

Outline of this Lecture

Intelligence

- Intelligence – ability to adapt oneself adequately to relatively new situations in life. (R. Pintner)
- Intelligence – having learned or the ability to learn to adjust oneself to the environment. (Colvin)
- Intelligence – the ability to carry out abstract thinking. (Terman)
- Intelligence – innate general cognitive ability. (Burt)

Artificial Intelligence

Artificial Intelligence

Artificial intelligence is intelligence demonstrated by machines.

Artificial Intelligence

The science and engineering of making intelligent machines.

Computational Intelligence

The ability of a computer to learn a specific task from data or experimental observation.

Artificial Intelligence

- AI is the new electricity. (Andrew Ng)

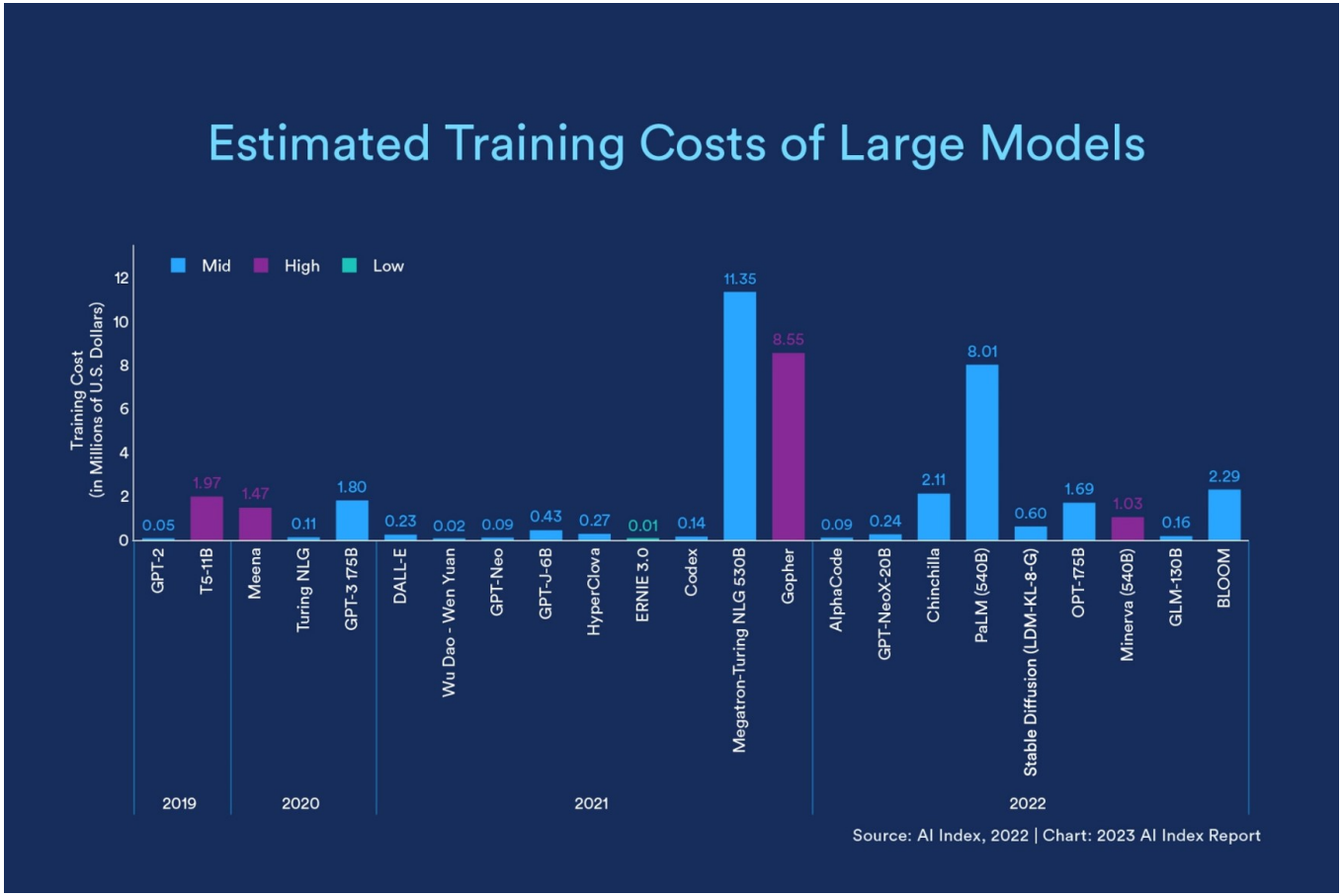
- Computer vision.
- Healthcare.
- Speech recognition.
- Natural Language Processing.
- Robotics.
- **Security**.
- ...

Artificial Intelligence

- Powerful hardware.
- Big data.
- Novel applications.

AI is Becoming Better

AI is Becoming More Expensive



Approaches

- Objective functions driven.
- Data driven.

Optimization as a goal

- What is an optimization?
- Informally speaking, it is a process of finding new, better solutions to various problems.
- People use **optimization methods** in every aspect of their lives.

Evolutionary Computation

- Research area within computer science that draws inspiration from the process of natural evolution.
- Evolutionary computation (EC) are 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.

How EAs work

1: *Input* : *Parameters of the algorithm*
2: *Output* : *Optimal solution set*
3: $t \leftarrow 0$
4: $P(0) \leftarrow$ *CreateInitialPopulation*
5: **while** *TerminationCriterion* **do**
6: $\quad t \leftarrow t + 1$
7: $\quad P'(t) \leftarrow$ *SelectMechanism* $(P(t-1))$
8: $\quad P(t) \leftarrow$ *VariationOperators*$(P'(t))$
9: **end while**
10: *Return OptimalSolutionSet*(P)

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.

Types of Machine Learning

- Supervised learning.
- Unsupervised learning.
- Semi-supervised learning.
- Reinforcement learning.

When to Use Machine Learning

- Difficult problems (computer vision, NLP, pattern recognition).
- Systems that dynamically change (robots, multi-agent systems).
- Data mining.

AI-complete Problem

A problem not possible to solve with a classical algorithmic approach. Implies that the difficulty of these computational problems is equivalent to that of solving the central artificial intelligence problem – making computers as intelligent as people.

Outline

Outline of this Lecture

AI and Cybersecurity

- RNG and PRNG
- Implementation attacks
- Hardware Trojans
- Modeling attacks on PUFs
- Design of cryptographic primitives
- Cryptanalysis
- Intrusion detection
- Malware and spam identification/detection
- Fuzzing
- Privacy-preserving machine learning
- Adversarial machine learning
- ...

History

- Communication between Norbert Wiener and Warren Weaver (1947).
- "A most serious problem, for UNESCO and for the constructive and peaceful future of the planet, is the problem of translation, as it unavoidably affects the communication between peoples..."
- "Also knowing nothing official about, but having guessed and inferred considerable about, powerful new mechanized methods in cryptography - methods which I believe succeed even when one does not know what language has been coded - one naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say "This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.""

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.

Outline

Outline of this Lecture

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.

Physically Unclonable Functions

- 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 concerning the number of challenge-response pairs (CRPs) the attacker can 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 (concerning different challenges).
- Strong PUFs have a virtually unlimited number of challenges c, but their CRPs are highly correlated.
- Given enough (often a 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 an analog physical clone).
- There exists no validated design of a strong PUF that is fully resilient against modeling attacks.

Physically Unclonable Functions

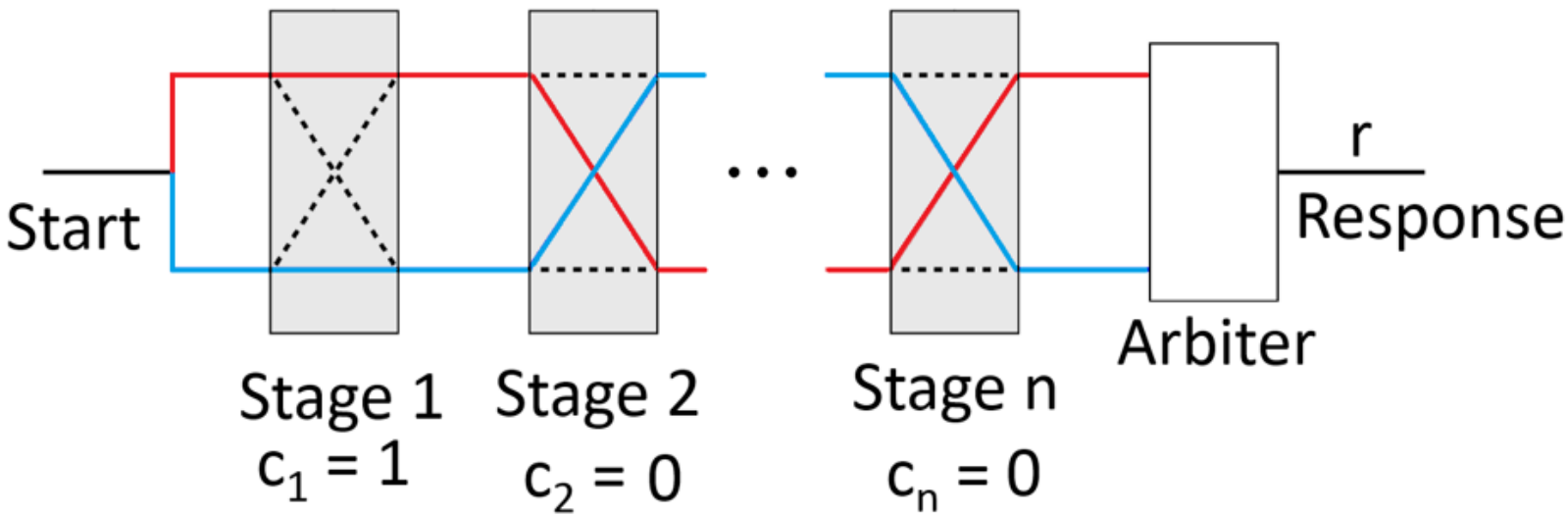


Figure: An example of a strong PUF - Arbiter PUF with n stages.

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 there is no clear benefit of using more complex techniques, e.g., deep learning.

Outline

Outline of this Lecture

Cryptographic Theory vs Physical Reality

- Cryptographic algorithms are (supposed to be) theoretically secure.
- Implementations leak in the physical world.

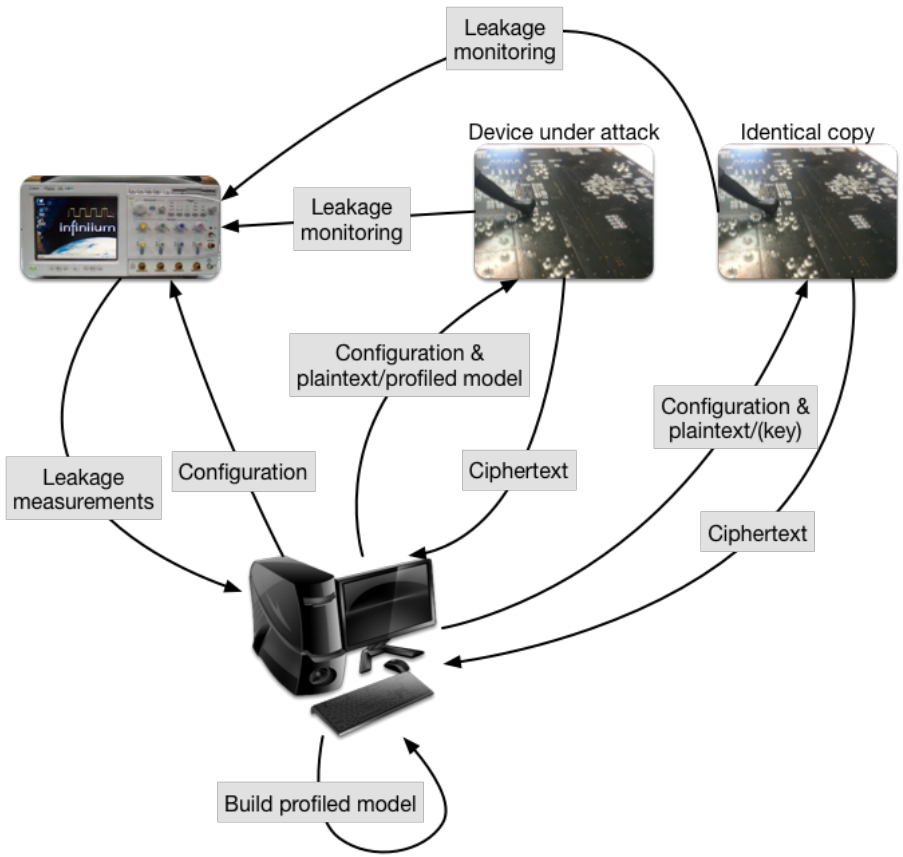
Implementation attacks

Implementation attacks do not aim at the weaknesses of the algorithm but at its implementation.

Profiling Attacks

- Profiling attacks have a prominent place as the most powerful among side-channel attacks.
- Within the profiling phase, the adversary estimates leakage models for targeted intermediate computations, which are then exploited to extract secret information in the actual attack phase.
- Template Attack (TA) is the most powerful attack from the information-theoretic point of view.
- Some **machine learning** (ML) techniques also belong to the profiling attacks.

Profiling Attacks



- Profiling attacks are more complicated than direct attacks.
- The attacker must have a copy of the device to be attacked.

Common "Traditional" Approaches in Profiling SCA

- Multilayer Perceptron.
- Naive Bayes.
- Support Vector Machines.
- Random Forest.

Deep Learning

- Stacked neural networks, i.e., networks consisting of multiple layers.
- Layers are made of nodes.

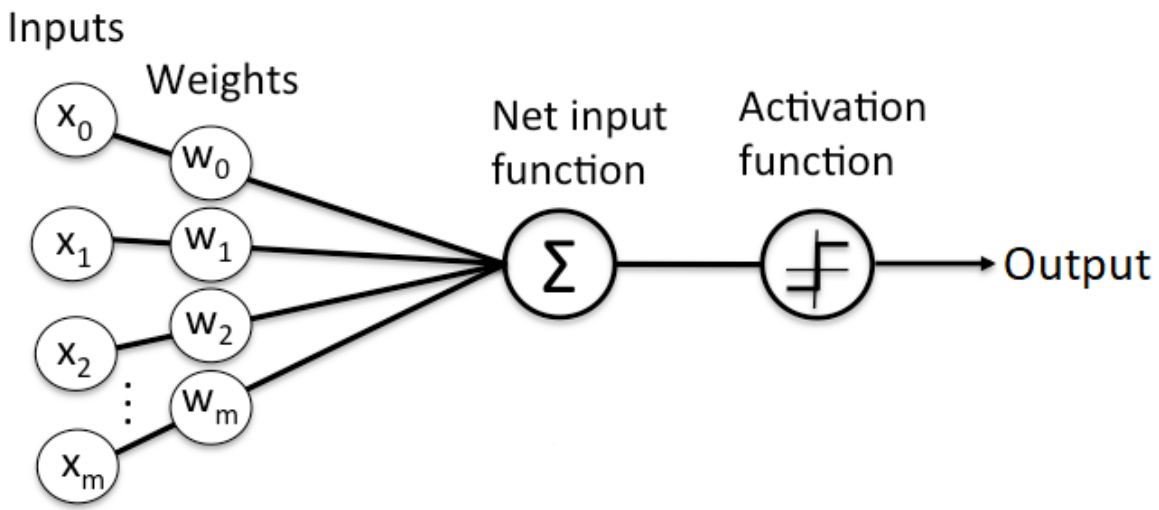


Figure: Perceptron.

Multilayer Perceptron

- One input layer, one output layer, and at least one hidden layer.

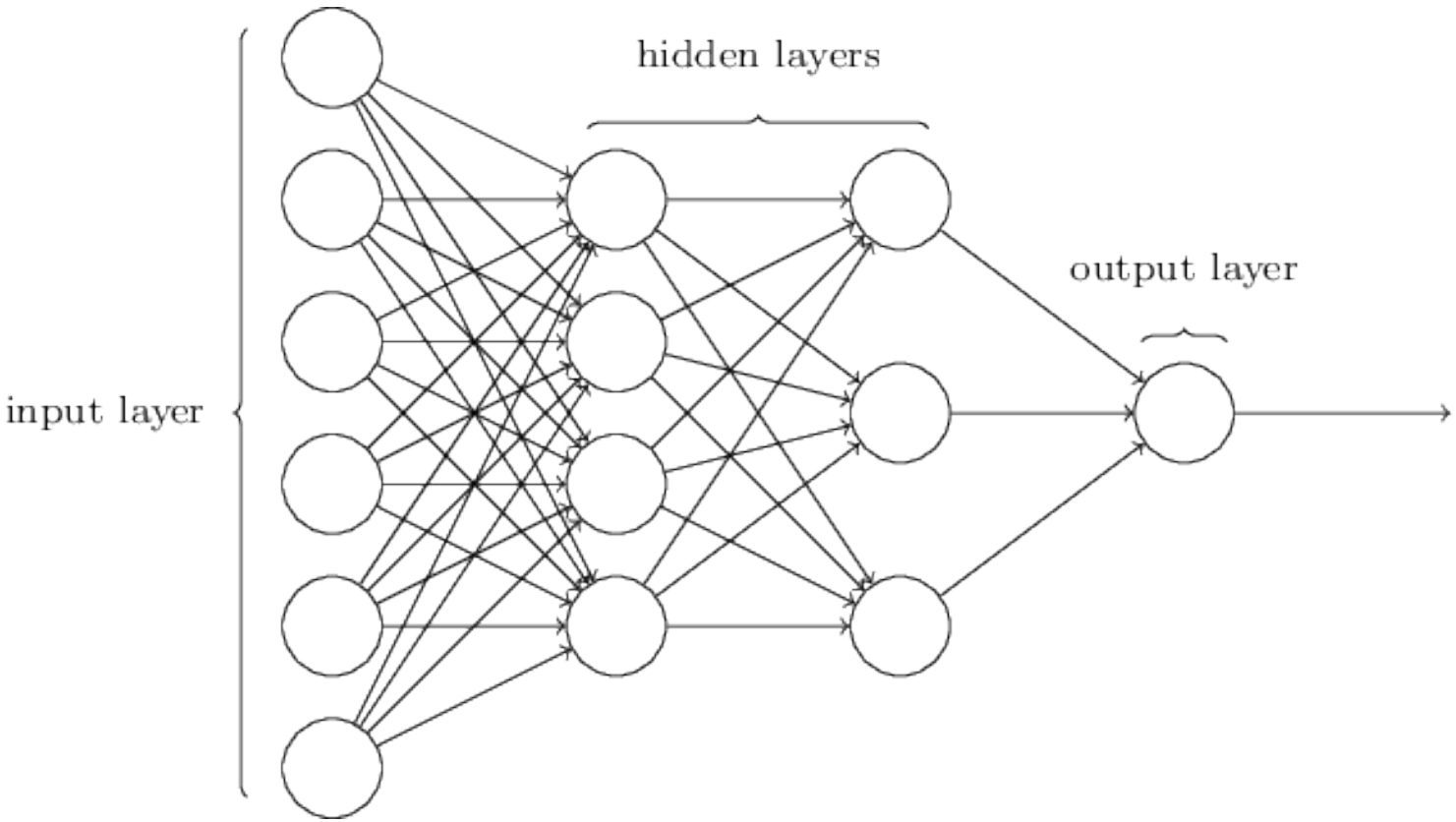


Figure: Multilayer perceptron.

Convolutional Neural Networks

- CNNs represent a type of neural network first designed for 2-dimensional convolutions.
- They are primarily used for image classification, but lately, they have proven to be powerful classifiers in other domains.
- From the operational perspective, CNNs are similar to ordinary neural networks: they consist of a number of layers where each layer is made up of neurons.
- CNNs use three main types of layers: convolutional layers, pooling layers, and fully-connected layers.

State-of-the-art Results with DLSCA

Table: Points of interest, the minimum number of attack traces to get guessing entropy equal to 1, model search success (when GE=1), and number of trainable parameters for all datasets and feature selection scenarios.

Dataset	Neural Network Model	Feature Selection Scenario	Amount of POIs (HW/ID)	Attack Traces (HW/ID)	Search Success (%) (HW/ID)	Trainable Parameters (HW/ID)
ASCADf	MLP	RPOI	200/100	5/**1**	99.22%/96.86%	82 209/429 256
ASCADf	CNN	RPOI	400/200	5/**1**	99.23%/99.08%	499 533/158 108
ASCADf	MLP	OPOI	700/700	480/104	82.80%/68.80%	16 309/10 266
ASCADf	CNN	OPOI	700/700	744/87	55.53%/35.33%	594 305/62 396
ASCADf	MLP	NOPOI	2 500/2 500	7/**1**	74.50%/39.00%	2 203 009/5 379 256
ASCADf	CNN	NOPOI	10 000/10 000	7/ **1**	15.40%/2.45%	545 693/439 348
ASCADf	CNN	NOPOI desync	10 000/10 000	532/36	2.44%/2.64%	268 433/64 002
ASCADr	MLP	RPOI	200/20	3/**1**	99.23%/100%	565 209/639 756
ASCADr	CNN	RPOI	400/30	5/**1**	100%/100%	575 369/636 224
ASCADr	MLP	OPOI	1 400/1 400	328/129	71.40%/37.25%	31 149/34 236
ASCADr	CNN	OPOI	1 400/1 400	538/78	47.92%/23.95%	270 953/87 632
ASCADr	MLP	NOPOI	25 000/25 000	6/ **1**	44.39%/7.02%	5 243 209/12 628 756
ASCADr	CNN	NOPOI	25 000/25 000	7/ **1**	19.17%/4.35%	369 109/721 012
ASCADr	CNN	NOPOI desync	25 000/25 000	305/73	0.71%/1.04%	22 889/90 368

Outline

Outline of this Lecture

Traditional Cryptanalysis

- Aims at recovering the secret key by using a knowledge of (P, C) pairs.
- Looking for patterns to distinguish encrypted data from random.
- Adversary's goal is to distinguish the output of a cipher from random data faster than brute force key search.
- Two common key-recovery attacks:
  1. differential cryptanalysis: exploits difference propagation
  2. linear cryptanalysis: exploits large $P - to - C$ correlations

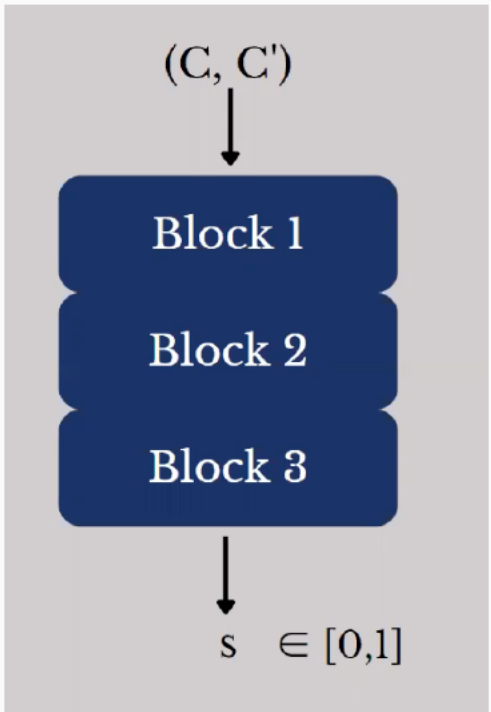
Differential Cryptanalysis

- Invented by Biham and Shamir in 1990 as a way to attack DES.
- Exploits a scenario where a particular ΔC occurs given a particular input difference ΔP with a "high" probability.
- It is a chosen plaintext attack, so the attacker will select pairs of inputs, P and $P^{'}$, to satisfy a particular ΔP.

Neural-aided Cryptanalysis

- Started by Gohr in 2019.
- Trained neural distinguishers of depth-10 and depth-1 for round-reduced versions of Speck32/64.
- The approach proved successful on 5-8 rounds (accuracy above 50%).
- Improved 11-round key recovery attack complexity on Speck32/64 (using Bayesian optimization).
- Up to now, used on more than 20 different cryptographic algorithms.

Neural-aided Cryptanalysis



A three-block **neural network**.

(C, C') is either the **encryption** of:
1. **(P, P')** where **P' = P ⊕ Δ**
2. **(P, P')** where there is **no fixed difference**

If **s ≥ 0.5** then (C, C') is a **real pair**
else (C, C') is a **random pair**.

Outline

Outline of this Lecture

Boolean Functions

- The easiest problem to start for evolutionary computation (EC) and crypto.
- A natural mapping between the truth table representation of Boolean functions and representation of solutions in EC.
- Boolean functions are an important cryptographic primitive often used in stream ciphers as the source of nonlinearity.
- Boolean functions are commonly used in combiner or filter generators (which are not so used anymore).

Boolean Functions

- Three main directions in the evolution of Boolean functions:
  1. Evolution of Boolean functions fulfilling a number of cryptographic properties (balancedness, nonlinearity, algebraic degree, correlation immunity, algebraic immunity), where some of the properties can be conflicting.
  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

- 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 representations give good results.
- Currently, the best results are achieved with GP/CGP.
- Obtained results are comparable with those from algebraic constructions.

Boolean Functions

Bitstring representation

```
<Individual size="1">
        <FitnessMax value="116.727"/>
        <BitString size="256">1111000000010111001101011110100011101100000010110
        0111000000010100011101010110101010010100010011101100001000011110011100
        1000010110000001010101111111111111111011101100001111010111011110001010100
        0010111100000011010010100100011111101001111011101000111010010010001001
        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:
01000100101010111100001100111111011100010111111111010101000000000010001111
00111111110001101010010111110111000110100010011101010010010000010011000010
00101000100001000111001001011111011001111000111001011110100111110001010011
10010010110011001111011000100101011100
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:
01010101111111111011010011100001111111111010101011100001101101001100110010
01100111010010100001111111001100011001101111100000101101010101010111111110
01011011000011000000000010101010011110001101001011001100011001101011010000
01111001100110110011000001111010110 10
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).

Learnability

- Learning as the phenomenon of knowledge acquisition in the absence of explicit programming.
- It is possible to design learning machines that have three of the following properties:
  1. The machines can provably learn whole classes of concepts. Furthermore, these classes can be characterized.
  2. The classes of concepts are appropriate and nontrivial for general-purpose knowledge.
  3. The computational process by which the machines deduce the desired programs requires a feasible (i.e., polynomial) number of steps.

Learnability

- A learning machine consists of a learning protocol together with a deduction procedure. The former specifies the manner in which information is obtained from the outside. The latter is the mechanism by which a correct recognition algorithm for the concept to be learned is deduced.
- There is circumstantial evidence from cryptography, however, that the whole class of functions computable by polynomial size circuits is not learnable.
- The existence of good cryptographic functions that are easy to compute therefore implies that some easy-to-compute functions are not learnable.

Learnability and Machine Learning

- Boolean formulas can be learned by deep neural networks.
- Common problems include model-sampling benchmarks, combinatorial optimization problems (graph coloring, clique), and random k-CNFs.

Heuristic design of cryptographically strong balanced Boolean functions

- Eurocrypt 98.
- Experiments for $n = 8$.
- Genetic algorithm capable of generating highly nonlinear balanced Boolean functions.
- Hill climbing techniques are adapted to locate balanced, highly nonlinear Boolean functions that also almost satisfy correlation immunity.

Evolving Boolean Functions Satisfying Multiple Criteria

- Simulated Annealing.
- Nonlinearity, Autocorrelation, Correlation Immunity, and Algebraic Degree.
- From $n = 5$ to $n = 12$.

Search for Boolean Functions With Excellent Profiles in the Rotation Symmetric Class

- Modified steepest-descent-based iterative heuristic search.
- Boolean functions on 9 variables having nonlinearity 241.
- 10 variable functions having first-order resiliency and nonlinearity 492.

Evolutionary Algorithms for Boolean Functions in Diverse Domains of Cryptography

- We can make more masking more affordable by using Boolean functions with small Hamming weight and high correlation immunity.

t \ n	11	12	13	14	15	16
2	16	16	32	64	128	256
3	32	32	32	64	128	256
4	**128**	256	256	256	2 048	4 096
5	**256**	**256**	512	1 024	2 048	4 096
6	512	**1 024**	1 024	2 048	4 096	4 096
7	1 024	1 024	**2 048**	4 096	8 192	8 192
8	1 024	2 048	4 096	8 192	8 192	16 384
9	1 024	2 048	4 096	8 192	16 384	16 384
10	1 024	2 048	4 096	8 192	16 384	32 768
11		2 048	4 096	8 192	16 384	32 768
12			4 096	8 192	16 384	32 768
13				8 192	16 384	32 768
14					16 384	32 768
15						32 768

Evolutionary Strategies for the Design of Binary Linear Codes

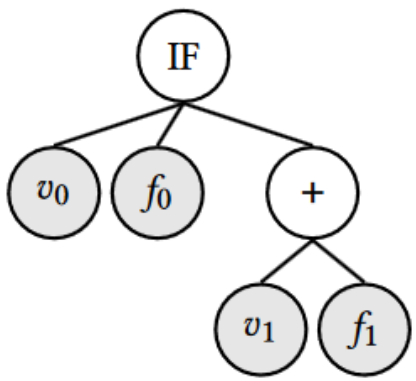
- Optimizing the minimum distance of unrestricted binary codes, i.e., with no constraints on their linearity, which is a desirable property for efficient implementations.
- Exploring only the subset of linear codes of a fixed length and dimension.
- Candidate solutions are binary matrices.
- Up to length $n = 14$, the algorithm always converges to an optimal solution with a full success rate, and the evolved codes are all inequivalent to the Best-Known Linear Code (BKLC) given by MAGMA.

Evolving Algebraic Constructions for Designing Bent Boolean Functions

- A common problem with previously described approaches is that they result in specific examples of Boolean functions.
- The question is can AI be used to obtain something more general.
- It is possible to use GP to evolve constructions of Boolean functions.
- ((((v1 XNOR f0) OR (f3 AND f0)) XOR ((f1 XOR v0) XNOR v1)) AND2 ((v0 AND2 f2) AND2 ((f0 XNOR f3) XOR (f1 AND2 v1))))

Evolving constructions for balanced, highly nonlinear boolean functions

- Finding balanced, highly nonlinear Boolean functions.
- Particular case of indirect sum function.



$$F(v_0, v_1, v) = \begin{cases} f_0(v) \ , & \text{if } v_0 = 1 \ , \\ f_1(v) \oplus v_1 \ , & \text{if } v_0 = 0 \ . \end{cases}$$

A wide class of Boolean functions generalizing the hidden weight bit function

- C. Carlet proposed a generalization of the Hidden Weight Boolean Function that allows a construction of n-variable balanced functions f from $(n-1)$-variable Boolean functions g satisfying some light conditions.
- This generalized HWBF construction allows keeping HWBF quality of being fast to compute if function g is fast enough to compute and has good algebraic immunity while improving its nonlinearity.
- Current results indicate that the best nonlinearity is obtained when g is a monomial function.
- Unfortunately, the values for nonlinearity are still far from the upper bound on nonlinearity.
- Change function g such that it is not monotone anymore but allows for higher nonlinearity.

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).

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.
- Usual 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 in the direction of resilience against side-channel attacks.
  3. To find more efficient implementations of S-boxes (efficient in terms of area and power).

S-boxes

- The best results are obtained with tree representation and cellular automata approach.
- In fact, this 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

```
<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:
100101101001100110010110100110010011110011001100001111001100
100001110111100011010010111010010100001110111100011010010110
110000000011111100111111110000001111001100001100111100110000
111101010000010100001010111110100101111110101111101000001010
101110110111011101000100011101110100010010001000101110111000
100110101001110100110101001101010011001010110010101101010011
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).
- Result is a bijective S-box with six inputs and outputs (ECF).

Outline

Outline of this Lecture

Conclusions

- AI has a prominent role in cryptography (and even more cybersecurity).
- Current results are promising.
- But we need more relevant problems.
- It does not seem the latest trends in AI are much used for research on Boolean functions.