Infocommunications Journal

A PUBLICATION OF THE SCIENTIFIC ASSOCIATION FOR INFOCOMMUNICATIONS (HTE)

March 2016 Volume VIII Number 1 ISSN 2061-2079

Technically Co-Sponsored by

IEEE ComSoc™ IEEE Communications Society

hte

IEEE HUNGARY SECTION

Indexing information

Infocommunications Journal is covered by Inspec, Compendex and Scopus.
Infocommunications Journal is also included in the Thomson Reuters – Web of ScienceTM Core Collection, Emerging Sources Citation Index (ESCI)

Infocommunications Journal

Technically co-sponsored by IEEE Communications Society and IEEE Hungary Section

Supporters

FERENC VÁGUJHELYI – president, National Council for Telecommunications and Information Technology (NHIT)
GÁBOR MAGYAR – president, Scientic Association for Infocommunications (HTE)

Editorial Office (Subscription and Advertisements)**:**
Scientic Association for Infocommunications
H-1051 Budapest, Bajcsy-Zsilinszky str. 12, Room: 502
Phone: +36 1 353 1027, Fax: +36 1 353 0451
E-mail: info@hte.hu • Web: www.hte.hu

Articles can be sent also to the following address:
Budapest University of Technology and Economics
Department of Telecommunications and Media Informatics
Tel.: +36 1 463 1102, Fax: +36 1 463 1763
E-mail: vida@tmit.bme.hu

Subscription rates for foreign subscribers: 4 issues 10.000 HUF + postage

Publisher: PÉTER NAGY

HU ISSN 2061-2079 • Layout: PLAZMA DS • Printed by: FOM Media

Special Issue on Applied Cryptography – Guest Editorial

Václav (Vashek) Matyáš, Zdeněk Říha and Pavol Zajac

***Abstract*—This special issue brings selected papers from the SantaCrypt 2015 workshop, held in Prague, December 3-4, 2015.**

THIS special issue focuses on the area of applied cryptography, bringing up selected papers from Santa's Crypto Get-Together (SantaCrypt), a workshop that runs since 2001 as an annual Czech and Slovak workshop aiming to facilitate closer cooperation of professionals working in the field of applied cryptography and related areas of security. All three papers deal with cryptanalysis, although each of them approaches this area from a completely different perspective.

The first paper "New results on reduced-round Tiny Encryption Algorithm using genetic programming" of Karel Kubíček et al. explores use of evolutionary computing for cryptanalysis of the Tiny Encryption Algorithm (TEA). The authors deploy EACirc, a genetically inspired randomness testing framework based on finding a dynamically constructed test of statistical properties of TEA outputs. This test works as a probabilistic distinguisher separating cipher outputs from truly random data. TEA was chosen as a "benchmark"' algorithm and the paper provides results of EACirc applied to the TEA ciphertext created from differently structured plaintext. A different construction of EACirc tests also allows the authors to determine which part of the cipher's output is relevant to the decision of a well-performing randomness distinguisher.

The second paper "Side Channels in SW Implementation of the McEliece PKC" of Marek Klein deals with the McEliece cryptosystem – that is considered secure in the presence of quantum computers because there is no known quantum algorithm to solve the problem this cryptosystem is built on. The author examines a naïve implementation of the cryptosystem from the point of side channels, which can be used to gather information about the message or the secret key. The paper presents results of chosen timing attacks on straightforward implementation of this cryptosystem, as well as practical countermeasures and evaluation of their effectiveness.

The third paper "Cryptanalysis based on the theory of symmetric group representations" of Romana Linkeová and Pavel Příhoda focuses on an alternative of the famous key exchange protocol of Diffie and Hellman, working over a structure of small matrices over a group ring, as proposed by D. Kahrobaei et al. 2013. Their modification aimed to address an issue of the original proposal of Diffie and Hellman, the issue of performance faced by devices with a limited computational power. Research of alternative algebraic structures lead, among others, to the proposal of D. Kahrobaei et al. Linkeová and Příhoda attack this modification and prove that it is not secure with the help of the theory of symmetric group representations.

Václav (Vashek) Matyáš is a Professor at the Masaryk University, Brno, CZ, and serves as a Vice-Dean for Foreign Affairs and External Relations, Faculty of Informatics. His research interests relate to applied cryptography and security, publishing over a hundred peer-reviewed papers and articles, and co-authoring six books. He was a Fulbright Visiting Scholar with Harvard University, Center for Research on Computation and Society, and also worked with Microsoft Research Cambridge, University College Dublin, Ubilab at UBS AG, and was a Royal Society Postdoctoral Fellow with the Cambridge University Computer Lab. Vashek was one of the Editors-in-Chief of the Identity in the Information Society journal, and he also edited the Computer and Communications Security Reviews, and worked on the development of Common Criteria and with ISO/IEC JTC1 SC27. Vashek is a member of the Editorial Board of the Infocommunications Journal and a Senior Member of the ACM. He received his PhD degree from Masaryk University, Brno and can be contacted at matyas AT fi.muni.cz.

Zdeněk Říha is an Assistant Professor at the Masaryk University, Faculty of Informatics, in Brno, Czech Republic. He received his PhD degree from the Faculty of Informatics, Masaryk University. In 1999 he spent 6 months on an internship at Ubilab, the research lab of the bank UBS, focusing on security and usability aspects of biometric authentication systems. Between 2005 and 2008 he was seconded as a Detached National Expert to the European Commission's Joint Research Centre in Italy, where he worked on various projects related to privacy protection and electronic passports. He was involved in the ePassport interoperability group known as the Brussels Interoperability Group. Zdeněk has been working with the WG 5 (Identity management and privacy technologies) of ISO/IEC JTC 1/SC 27. Zdeněk's research interests include smartcard security, PKI, security of biometric systems and machine readable travel documents. Zdeněk can be contacted at zriha AT fi.muni.cz.

Pavol Zajac is an Associate Professor at the Slovak University of Technology in Bratislava. His main research interests lie in the area of mathematical cryptography. Nowadays he works mostly with post- quantum cryptography and related algebraic problems. Pavol can be contacted at pavol.zajac AT stuba.sk.

New results on reduced-round Tiny Encryption Algorithm using genetic programming

Karel Kubíček, Jiří Novotný, Petr Švenda, Martin Ukrop

***Abstract*— Analysis of cryptoprimitives usually requires extensive work of a skilled cryptanalyst. Some automation is possible, e.g. by using randomness testing batteries such as Statistical Test Suite from NIST (NIST STS) or Dieharder. Such batteries compare the statistical properties of the function's output stream to the theoretical values. A potential drawback is a limitation to predefined tested patterns. However, there is a new approach – EACirc is a genetically inspired randomness testing framework based on finding a dynamically constructed test. This test works as a probabilistic distinguisher separating cipher outputs from truly random data.**

In this work, we use EACirc to analyze the outputs of Tiny Encryption Algorithm (TEA). TEA was selected as a frequently used "benchmark" algorithm for cryptanalytic approaches based on genetic algorithms. In this paper, we provide results of EACirc applied to TEA ciphertext created from differently structured plaintext. We compare the methodology and results with previous approaches for limited-round TEA. A different construction of EACirc tests also allows us to determine which part of cipher's output is relevant to the decision of a well-performing randomness distinguisher.[1]

***Index Terms*—randomness statistical testing, TEA, genetic algorithms, randomness distinguisher, software circuit**

I. INTRODUCTION

Automatized randomness testing is useful for checking one of the expected cipher properties – output ciphertext should be indistinguishable from a stream of truly random data. This property alone is not sufficient for a cipher to be secure, but the ability to distinguish ciphertexts from random data constitutes an important hint on potential cipher weaknesses.

The common way to automate testing of randomness is using statistical batteries. NIST STS [1] is a standard battery of tests commonly used for this purpose, together with other batteries such as Diehard [2], Dieharder [3] or TestU01 [4]. The batteries contain sets of fixed tests (usually parameterized to form multiple different subtests) checking expected statistical properties of tested output stream (TEA ciphertext in our case) in comparison to the expected values for truly random data. Empirical tests of randomness fall under the standard statistical model – statistical hypothesis testing. Tests assume the assessed bitstream is random (the null hypothesis) and try to reject it (to show the bitstream is not random). Each randomness test is defined by the test statistic S, which is a real-valued function of a numeric sequence. Tests are evaluated by comparing the p-value (computed from the test statistic) with a chosen significance level α. For the p-value computation, it is necessary to know an exact distribution of the statistic S under a valid null hypothesis or, at least, its close approximation.

The limitation of the standard batteries for randomness testing is the fact they implement a fixed set of tests and can detect only a limited set of patterns and statistical irregularities. If the used set of tests is fixed and known, a sequence of completely deterministic data can be crafted such that no tests will detect statistically significant deviances from truly random data. However, as cryptographic functions have a deterministic output (dependent only on input data and a key), it is a priori expected that the function output cannot pass all possible tests of randomness and so there exist tests that reveal the output sequence as non-random. However, such a test can be very difficult to find.

In this work we use EACirc [5], a novel framework for constructing empirical tests of randomness that can succeed in finding such a test (at least hypothetically). Our goal is to find an empirical test of randomness that indicates if a given sequence is either non-random (with a high probability) or sufficiently indistinguishable from a truly random data stream. In this framework, randomness tests are created iteratively, adapting to the processed sequence. The construction is stochastic and uses genetic programming [6]. The tests are constructed from a predefined pool of operations. A set of these operations, together with a limit on the total number of operations, allows us to control the complexity of the constructed tests. The framework theoretically enables us to build an arbitrary randomness test over a set of chosen operations (in practice, however, the total number of operations used is limited). Therefore, it can be viewed as a general framework for the test construction and it could (hypothetically) provide a better detection ability than standard tests.

TEA has been intensively analyzed, including randomness testing of cipher output with stochastic genetic algorithms. Capabilities of EACirc are compared with previous results as well as conventional statistical batteries.

This paper is organized as follows: section II introduces TEA as a simple encryption algorithm applied nowadays as a benchmark for randomness tests. Subsequently, section III contains information about EACirc with the definition of used settings. Input data structure is also discussed in this section. Results and their interpretation are presented in section IV with analysis of found distinguishers and performance and data usage of EACirc. In section V, we describe the future work.

Manuscript submitted on September 29, 2015, revised on February 18, 2016.

All authors are from Masaryk University, Brno, Czech Republic. They can be contacted by email address {karel.kubicek, jiri.novotny, xsvenda, mukrop}@mail.muni.cz.

[1]Paper supplementary material available at http://crcs.cz/papers/infocomm2016

II. TINY ENCRYPTION ALGORITHM

Tiny Encryption Algorithm (TEA) is a block cipher designed by David Wheeler and Roger Needham [7]. The algorithm was designed to have a simple structure based on the Feistel network with 32 rounds (we count two steps of Feistel network as 1 TEA round). The cipher uses plaintext blocks of 64 bits and keys of 128 bits.

A. TEA distinguishers – state of the art

Nowadays, TEA is not considered secure for regular use as it suffers from multiple weaknesses, most significantly the related-key attack [8]. However, it is frequently used as a benchmark for randomness testing using genetic algorithms. Starting in 2002 with a paper by Julio C. Hernández, José M. Sierra, Pedro Isasi and Arturo Ribagorda [9], statistically significant deviances were found for TEA limited to 1 and 2 rounds. Fixed bitmask with high Hamming weight evolved by genetic algorithms was applied both to the cipher input data and key. The expected distribution of bit patterns of 10 least significant bits of ciphertexts were then evaluated with a χ^2 test. A similar team published new results with the same approach but improved settings of genetic algorithms [10], which also detected deviances for 3 and 4 rounds. Aaron Garrett, John Hamilton and Gerry Dozier [11] extended this work in 2007 with new optimizations of the fitness function, which helped to create masks with a higher weight for 1 and 2 rounds TEA but failed to surpass previous results for 3 and 4 rounds.

Wei Hu et al. [12] in 2010 used quantum inspired genetic algorithm and a similar approach with bitmasks and succeeded with TEA limited to 4 and 5 rounds. Eddie Yee-Tak Ma and Chalie Obimbo [13] realized an attack on TEA limited to 1 round in 2011 utilizing genetic algorithms and harmony search for the derivation of degenerated keys instead of detection of statistical deviances of output.

III. OUR APPROACH

A. Randomness testing with genetic programming

As stated in the section I, the common way of automating randomness testing is the use of statistical batteries with predefined tests such as NIST STS. The approach based on genetic algorithms is different, as used tests iteratively evolve and adapt to the presented data.

Firstly, a set of individuals is created with each representing a candidate distinguisher function. Secondly, every individual decides if the provided block of input data is random or non-random. Thirdly, as the correctness of the decision is known, better individuals can be selected. Individuals are randomly mutated or cross-bred to create (hopefully) better descendants. The process follows the principles of biological evolution. I.e. if ciphertexts share a common statistical property (e.g. correlation between i^{th} and j^{th} bit), then an individual capable of expressing this property can potentially be evolved and improved in the process of further evolution.

The use of genetic algorithms also induces a couple of disadvantages. We are affected mainly by these:

- As the representation of the distinguisher functions is not straightforward, there are many possible candidate configurations. This induces a search space that may be artificially and unnecessarily large if the representation is not properly selected.
- Genetic modifications of candidate solutions from the previous iteration (mutation, crossover) are done randomly, and configuration space may not be completely searched. A well-working distinguisher can be missed even if it exists.
- The process of fine-tuning the parameters of genetic algorithms can significantly influence the quality of distinguishers found. E.g. [10] found distinguisher for a higher number of rounds then [9] although using same underlying approach and representation.

For more details about possible problems and their solution in EACirc, refer to the thesis of Martin Ukrop [14], section 3.1.

B. EACirc framework

The constructed distinguisher is a small program that simulates a standard hardware circuit. It consists of logic gates (nodes) grouped into layers. Every gate in a layer is connected to several nodes from the layer above using connectors (see fig. 1). It is crucial that the functionality of the circuit (circuit-like software) can be simply changed by replacing operations in gates or by redirection of connectors. This property is used for an iterative construction of distinguishers. The construction is controlled by genetic programming that uses a fitness function (success rate) based on the ability of a distinguisher to correctly indicate a given bitstream to be non-random with a high confidence. A well-performing distinguisher is able to assign non-random inputs, the outputs with a significantly different distribution than outputs assigned to truly random inputs. The output distribution difference is formalized using the Kolmogorov-Smirnov test [15].

The supposed usage of EACirc is similar to the statistical batteries. The process is fully automatized with statistical results that are simple to interpret. Additionally, EACirc can be used as a tool for showing cipher weaknesses for manual crypt-analysis performed later. For example, skilled cryptologist can see from fig. 1 what part of TEA is causing statistical deviations and how the weakness can be exploited as shown in the result interpretation (section IV-C).

The whole framework is being continuously extended and enhanced by our team and is accessible online with full documentation [5].

C. EACirc parametrization

EACirc can be configured on multiple levels: firstly, the representation of software circuit used to express candidate distinguishers and, secondly, the parameters of genetic programming. General settings are described in the thesis of Martin Ukrop [14], chapter 4 and project's documentation [5]. The following settings were relevant for TEA analysis.

Functions in nodes: Circuit nodes can contain an identity function, constant-producing function, basic logical binary operators, shifts and rotations, integer comparison

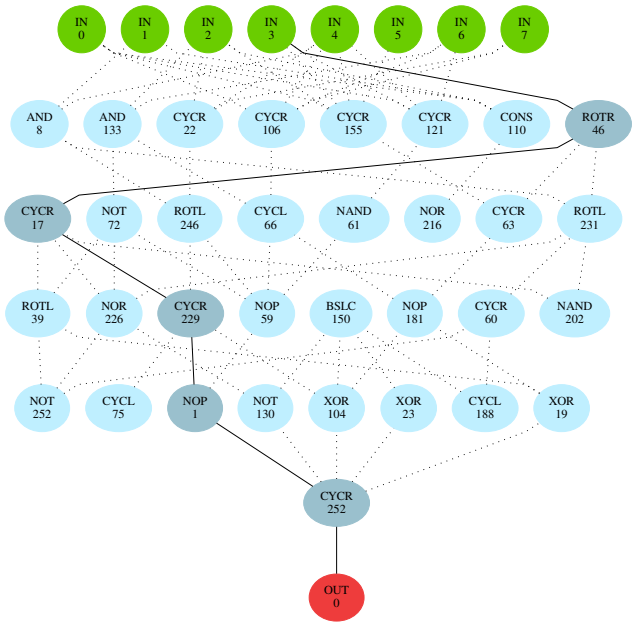


Figure 1. Software circuit with green input nodes, blue inner nodes (operations) and a red output node. Inner nodes and connectors are highlighted if they affect output (dotted edges and lighter nodes are part of evolved circuit, but they are not affecting the output byte in any way). This circuit was evolved in the experiment with 4-round TEA.

functions, masks for bit selection and additional input read operator. The larger diversity of functions means stronger expression capability (within the limited space). However, this vastly increases the space of applicable individuals slowing down the evolution process. Due to this, the set of used function was restricted (integer comparison functions and additional input read operator were not used). All functions are byte-oriented.

Circuit dimensions: In our case, the input layer has the same size (or multiple of) as the TEA block. Other relevant settings include providing more TEA ciphertext blocks as a single input (which would again slow down the evolution considerably). We used 5 internal layers with 8 nodes per layer. The last layer contains a node with 1-byte long output used as the circuit's overall result. See fig. 1 as an example of circuits in our experiments.

Test vectors: Another important setting influencing the success rate of EACirc is the number of test vectors used to evaluate the performance of candidate distinguishers (circuits). In our scenario, the set of test vectors consists of 2 subsets: TEA ciphertexts and data from a quantum random number generator (believed to be completely random), with both subsets of vectors having the same size. More test vectors mean more data for each iteration of evolution to learn from, as well as more precision for the fitness function. On the other hand, more test vectors also need more computation time as every candidate circuit is always evaluated for every separate test vector. In this work, we used two main configurations: for CPU-only version, 1 000 test vectors were used. For nVidia CUDA implementation, 128 000 test vectors were used (see section III-D).

Generations: The number of evolved generations influences the length of searching for the cipher properties. In our case, 30 000 generations were used. The number increased to 300 000 generations provided no observable improvement.

Population size: Number of individuals in a population. We use only one individual for each iteration, which is mutated into two individuals for next generation – an approach similar to hill-climbing heuristics. More individuals may increase the success rate and convergence speed towards a well-performing distinguisher, but may also negatively influence the interpretation of results as different individuals may be correlated. For this reason, the interpretation of results from more individuals is left for future.

D. Accelerated computation using GPGPU

The more test vectors are processed, the more computation time to evaluate a circuit is needed. Since the evaluation on a set of test vectors is parallel in nature, data parallelism techniques can be applied. To reduce the runtime and to fully utilize our hardware, the evaluation is optionally computed on GPGPU accelerators using nVidia's CUDA technology. We are running multiple instances of the evaluated circuit on each test vector in the set in parallel.

On used hardware (Intel Core2Duo E8400 and nVidia GeForce GTX 460) the GPU acceleration gives us 229× speedup for circuit evaluation (see fig. 2 for more detailed benchmark results). The execution of EACirc with 1 000 test vectors running only on CPU takes approximately the same time as the GPU-accelerated version with 128 000 test vectors (about 3.5 minutes).

E. Statistical uniformity testing

During the process of evolution, distinguishers iteratively adapt to the set of test vectors with a p-value computed in each iteration. We use the fact that for independent samples of truly random data the p-values are uniformly distributed on the interval [0,1]. To leverage this, we intentionally use only p-values from iterations just after the test vectors were regenerated to separate data for training and verification (we regenerate the set every 100^{th} generation). We can then test their uniformity using the Kolmogorov-Smirnov (KS) test [15] with the assumption that p-values are expected to be uniformly distributed. KS computes its own p-value which is compared with a significance level $\alpha = 5\%$ to draw the conclusion (p-value below the significance level makes us reject the randomness of the assessed data). Since KS gives a probabilistic answer, we repeat the whole process 1 000 times to avoid statistical anomalies. For the random data, it can be expected that about 5% of all runs fail the testing process (since the significance level is set to 5%).

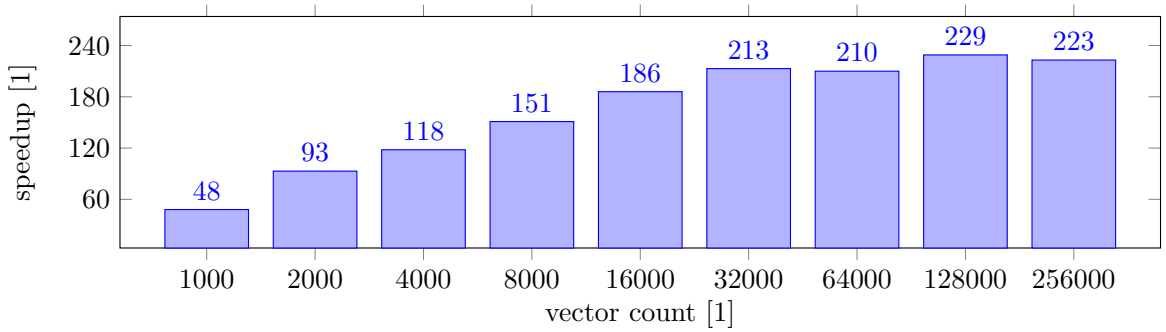


Figure 2. Speedup of GPU-accelerated circuit evaluation against circuit running on CPU, computed as $time_{CPU} / time_{GPU}$. The performance of a GPU accelerator is generally dependent on work size, in our case the number of test vectors. The benchmark used machines equipped with Intel Core2Duo E8400 at 3 GHz and nVidia GeForce GTX 460 with 336 CUDA Cores on 1550 MHz.

F. Oneclick

As EACirc is randomized in nature, we need to run many tests in parallel. To ease the time-consuming monkey-work of running and post-processing experiments, we use a tool called Oneclick [16], which distributes computations using the BOINC infrastructure [17] on the laboratory computers. This tool reduces both the necessary human work and the running time of the experiment.

G. TEA customization

For complete automation, the tested ciphers are included as plugins into EACirc, which then both generates the test vectors and runs the distinguisher evolution. Since we want to test TEA with a variable number of rounds (not only the recommended 32), we use a slightly changed version of the cipher that is shown below.

Algorithm 1 encrypt(uint32_t *data, const uint32_t *key)

```
const uint32_t delta = 0x9e3779b9;
uint32_t sum = 0;
for int j = 0; j < numRounds; j++ do
  sum += delta;
  data[0] += ((data[1]«4) + key[0]) ^ (data[1] + sum)
           ^ ((data[1]»5) + key[1]);
  data[1] += ((data[0]«4) + key[2]) ^ (data[0] + sum)
           ^ ((data[0]»5) + key[3]);
end for
```

The function input is a plaintext block (64-bits long), stored in the array `data`, and the `key` array of length 128 bits. The output is stored in array `data`. Only changed part of the algorithm is limiting the rounds count to `numRounds`.

H. Design of experiments

There are various settings for generation of output data stream from TEA. The first decision is which cipher mode should be used. We used the electronic codebook (ECB) mode, as this was the case of previous papers on the topic since [9]. This also minimizes the influence of the used mode on the output stream of data (ciphertext) produced by TEA.

An important factor is how the plaintext for TEA is generated. Even a weak cipher will usually provide a strong output if completely random input data are supplied as input. Our framework does not mask the input data with specific bitmask (as was the case in [9]) but instead generates input with some redundancy as described below.

The following ways to generate plaintexts for TEA were implemented:

1) The counter incremented by one for each test vector. This solutions is simple and does not suffer from the problem of repeating plaintexts. It also corresponds to potential usage of the cipher (e.g. if used in the counter mode). On the other side, it has a low Hamming weight (first 40 bits of the plaintext consist only of zeroes). Therefore, this type of plaintext is very difficult to compare with the methodology of previous works.
2) The vectors with 5 randomly placed 0 and other bits set to 1. The number 5 was chosen to create enough unique test vectors (over 10 million). This input also has an extreme (and fixed) Hamming weight, but there are no positions with fixed bits.
3) The vectors with two almost identical parts differing only in a single bit. We used this plaintext type for testing the strict avalanche criterion. The first input block of TEA is fully random and the second is the same with only a single changed bit. In this case, the circuit uses test vectors of 128 bits.

A similar reasoning is relevant for the generation of secret keys used. As we already manipulated input data for the cipher, we used a fixed (but completely random) value as a key for the whole test. For more information about the impact of key reinitialization frequency, please refer to the thesis of Martin Ukrop [14].

Used settings were chosen to simulate TEA usage. Users typically do not change encryption keys during a single session. Input data usually contain some redundancy, as long as meaningful text is processed. Other input types were selected to search for unwanted dependencies inside the cipher.

Table I
COMPARISON OF PREVIOUS RESULTS FOR REDUCED ROUNDS TEA.

Rounds	HSIR02 [9]		HI04 [10]		Wei Hu et al. [12]	
	χ^2	(MW)	χ^2	(MW)	χ^2	(MW)
1	8380416	72	522240	153	522.240	153[1]
2	1900	77	736.05	155	602	171
3	(untested)		393.6	116	530.756	117.8
4	(untested)		294.86	50[2]	742.632[1]	67.6[1]
5	(untested)		(untested)		631.74	76

IV. RESULTS

A. Comparison

Results from previous papers can be difficult to compare with ours because the approaches are significantly different (described in section II-A and section III-H). There are no results of statistical batteries from previous works. Therefore, we cannot use them as a common basis for the comparison. Our results can be compared in terms of rounds count, but tested data are different.

All previous works published weights of constructed masks (abbreviated as (MW) in the table), which were used on both the input block and key. This means the mask length is $64 + 128 = 192$ bits, which is a maximum weight for unchanged input. They also presented the average χ^2 statistics of maximal deviation from a random distribution.

Table II, table III and table IV provide a comparison of results from statistical testing batteries NIST STS (version 2.1.1) [1] and Dieharder (version 3.31.1) [3] run in default configuration together with EACirc on the given plaintext type. The result in the cell for Dieharder is the number of passed tests (out of the total 57 tests). From NIST STS, we used all 188 tests. Both batteries used the significance level $\alpha = 1\%$. Results that fail to reject the null hypothesis (are unable to show the non-randomness in data) have gray-colored background. The column for EACirc represents the best results achieved in our experiments. Values from EACirc represent the percentages of runs for which the set of p-values failed the KS test for uniformity with the significance level $\alpha = 5\%$. For the reference random-random distinguishing experiment, the value of 5% is expected (and also measured), so we also mark such results with gray background (data indistinguishable from random). For more detailed explanation of this method, please refer to [18], section 3.2.

We tried different settings of EACirc with the goal of finding the best distinguisher possible. Changes from the default settings (specified in section III-C) are following (EACirc$_{xy}$, where x denotes plaintext type and y stands for EACirc parameters):

- EACirc$_{1a}$ was tested with plaintexts created as a counter incremented by one for each vector (type 1). Besides, this version did not allow shifts and rotations in nodes.

[1]These results are computed as the average of values from tables of the original work [12]. Please note that average value is simplified and for more information refer to the original work.

[2]For this result, a different approach was used. Apart from that, the output mask has very low entropy. For more information, please refer to the original paper [10] (section 2.4).

Table II
COMPARISON OF EACIRC AND STANDARD STATISTICAL BATTERIES WITH PLAINTEXT CREATED AS A COUNTER STARTING FROM ZERO (TYPE 1). GRAY-COLORED CELLS INDICATE THE EXPERIMENTS THAT FAILED TO REJECT THE RANDOMNESS OF TESTED DATA.

Rounds	NIST	Dieharder	EACirc$_{1a}$	EACirc$_{1b}$	EACirc$_{1c}$
	$(x/188)$	$(x/57)$	$(\%)$	$(\%)$	$(\%)$
1	1	0	100	100	100
2	1	1	100	100	100
3	27	3	100	100	100
4	183	31	5.0	99.8	100
5	188	51	3.0	5.6	5.3

Table III
COMPARISON OF EACIRC AND STANDARD STATISTICAL BATTERIES WITH PLAINTEXT WITH 5 RANDOMLY PLACED ZEROES (TYPE 2). GRAY-COLORED CELLS INDICATE THE EXPERIMENTS THAT FAILED TO REJECT THE RANDOMNESS OF TESTED DATA.

Rounds	NIST	Dieharder	EACirc$_2$
	$(x/188)$	$(x/57)$	$(\%)$
1	24	1	100
2	183	8	93.3
3	188	39	5.6
4	187	44	5.6
5	187	48	5.5

- EACirc$_{1b}$ used the same settings as EACirc$_{1a}$, except shifts and rotations in nodes were allowed.
- EACirc$_{1c}$ had the same settings as EACirc$_{1b}$ but used the nVidia CUDA implementation, which allows to use 128 000 test vectors as well as increase the evaluator precision.
- EACirc$_2$ was tested with plaintexts of all ones (64b for TEA), with 5 flipped bits to zero on random positions (type 2).
- EACirc$_3$ was tested with twice the input length. The first block is random, and the second is identical to the first but for one bitflip on a random position (plaintext type 3). The total test vector length is 128 bits.

B. Results interpretation

The direct comparison of success with the previous papers is not straightforward due to the different approaches used. In previous approaches, to determine which bits of plaintexts will cause the output of round-reduced TEA to be non-uniform (tested by χ^2 test), the input was changed by applying a bitmask. In this paper, the goal is to find defects in ciphertexts

Table IV
COMPARISON OF EACIRC AND STANDARD STATISTICAL BATTERIES WITH PLAINTEXT SUITABLE FOR STRICT AVALANCHE CRITERION TESTING (TYPE 3). GRAY-COLORED CELLS INDICATE THE EXPERIMENTS THAT FAILED TO REJECT THE RANDOMNESS OF TESTED DATA.

Rounds	NIST	Dieharder	EACirc$_3$
	$(x/188)$	$(x/57)$	$(\%)$
1	29	6	100
2	67	7	100
3	186	24	100
4	187	39	100
5	188	56	4.5

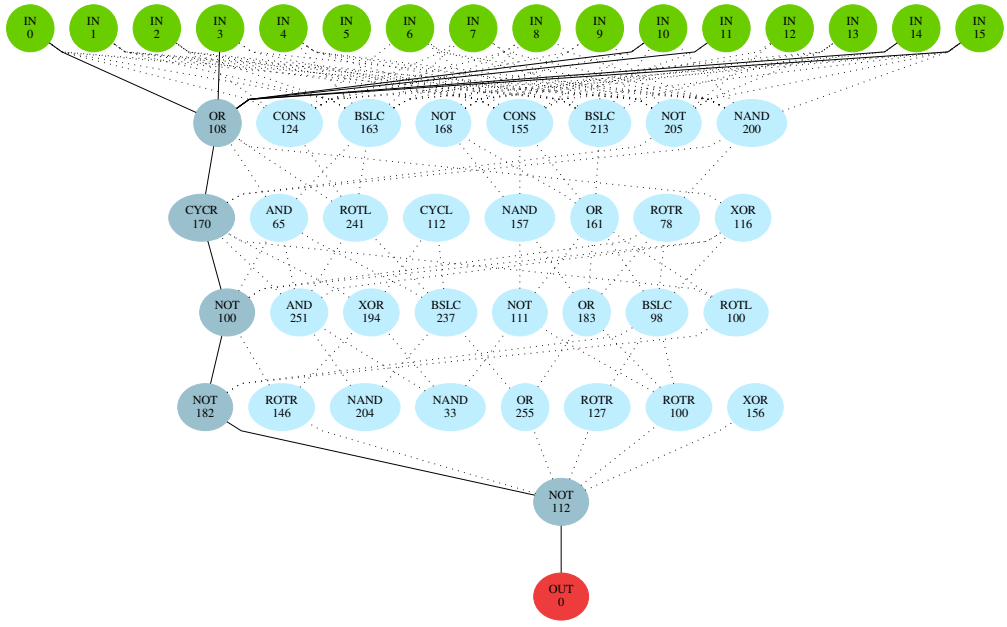


Figure 3. This circuit was evolved in the experiment with 4-round TEA on plaintexts suitable for testing the strict avalanche criterion (type 3). It can be seen that the output byte is mostly dependent on the 4^{th} and the 12^{th} (4^{th} in the second half) byte.

(dependent bits, biased bits, etc.) without directly manipulating plaintexts for the cipher. If we compare only the resulting number of rounds, for which output of the round-reduced TEA can be seen as statistically different from a random bit stream, the best results are provided by [12].

Comparing our approach with the statistical batteries is more straightforward, as we can use the same input data (ciphertext) for EACirc as well for statistical batteries. In all tested combinations (different structures of plaintext), EACirc consistently performs better than NIST STS. Dieharder is able to detect small deviances in one additional round (see table II).

Some information about the cipher can be derived from the results from different plaintext types. For example, statistical batteries perform better than EACirc on plaintexts with just 5 zeroes (type 2). Another observation is based on plaintext types 1 and 3 – EACirc is easily able to detect non-randomness of 4-rounds TEA, but fails to do so for 5 rounds. The same issue may be present for both plaintext types (the 5^{th} round reducing this issue).

The result of each run is single bit fail/pass result (p-value computed by KS uniformity test is smaller/bigger than the significance level). This often leads to the loss of interesting information – what is the quality of the evolved distinguisher and what dependency of output bits was found? Therefore, we decided to perform a deeper analysis of the found distinguishers.

C. Resulting circuits interpretation

The outcomes presented in the previous subsection are aggregated results over 1 000 different EACirc runs providing the single p-value for interpretation. Such an approach can provide superior detection of statistical deviances, but will not signalize which concrete bit(s) and dependencies between them cause these deviances – information valuable for the cipher's designer. By analysis of a single well-performing circuit, such an information can be obtained. We analyzed in detail some successful (fitness over 95%) evolved distinguishers. The weaker the distinguisher is, the more noise is present (circuit functionality is not performing as a correct distinguisher for the increasing number of inputs) and the harder it is to reason about the exact bits on which distinguisher's decision is based.

EACirc circuits used in this paper have 4 layers with 8 nodes in each. As a result, the evolved distinguisher can be rather complex and thus difficult to interpret. To provide a better comprehension, reductions of the target circuit can be performed as the distinguisher usually does not use all available nodes and connectors (they do not contribute to the output byte).

First of all, we can prune circuits taking into account the arity of used operations removing the unnecessary connectors. Manual analysis of pruned circuits is considerably simpler.

In the case of 4-round TEA on counter plaintexts (type 1), we analyzed several distinguishers with the fitness over 98%. In all of these circuits (see for example fig. 1) the distinguisher decision is based on the fourth byte of TEA ciphertext. The fourth byte is usually almost unchanged (operations affect only some bits).

We also analyzed 4-round TEA on plaintexts suitable for strict avalanche criterion testing (type 3). In this case, the input layer had 16 input nodes, capable of processing two blocks of TEA ciphertext at once. Analyzed distinguishers (see for example circuit in fig. 3) commonly combine the fourth byte of the first ciphertext block with the fourth byte of the second ciphertext block.

$$\Sigma = 1\,000\ \frac{\text{runs}}{\text{experiment}} \cdot \left(\frac{30\,000\ \frac{\text{generations}}{\text{run}}}{100\ \frac{\text{generations}}{\text{test set}}} \cdot \frac{1}{2} \cdot 1\,000\ \frac{\text{vectors}}{\text{test set}} \cdot 8\ \frac{\text{bytes}}{\text{vector}} \right) \approx 1,2\,\text{GiB per experiment}$$

Figure 4. The amount of data analyzed by EACirc for a single configuration of randomness testing experiment.

D. Performance

The runtime of EACirc with basic settings (1 000 test vectors and 30 000 generations) is around 3.5 minutes on a single core 3 GHz Intel Core2Duo processor. It includes the creation of test vectors and is not measurably affected by the number of TEA rounds executed. Due to the randomized nature of the framework, we replicate every experiment 1 000 times. This gives us a combined computation time of approximately 3 500 minutes on single CPU core.

Since the individual runs are independent, execution can be parallelized and distributed over multiple computers. We used 12 laboratory computers with the 3 GHz Intel Core2Duo processors mentioned above, which resulted in the execution time of about 5 hours for every single tested scenario. Thus, testing TEA limited to 1-8 rounds can be executed within 2 days of computation. For larger sets of tests, we used the national grid infrastructure provided by MetaCentrum [19].

Tests with 128 000 test vectors were executed on GPUs using nVidia CUDA. The running time for each test was around 3.5 minutes. As more GPU cores are available for parallelization of circuit evaluation, a higher amount of test vectors could be evaluated. The runtime was still linearly dependent on the generation count – tests with 300 thousand generations and 128 000 test vectors had a running time of around 105 minutes.

EACirc needs truly random data as a reference stream for a distinguisher evolution phase. We used a pool of 1920 MiB of data pre-loaded from the High Bit Rate Quantum Random Number Generator Service [20].

Regarding the TEA ciphertext, we generated 500 test vectors (half a set) of 64 bits each in 300 test vector sets in 1 000 runs for statistical interpretation. This amounts to 1.2 GiB of ciphertext data for the whole experiment or 1.2 MiB for a single run. See fig. 4 to understand, how we figured out the data usage of EACirc.

V. Discussion and Future work

The EACirc framework is continually developed and extended with different inner approaches and settings with the goal of improving the distinguisher success rates. At the moment, we work on two alternative circuit representations. One with the possibility of executing more complex Java bytecode sequences in circuit nodes. The sequences would be extracted directly from the Java implementation of the tested ciphers. Another alternative would be based on polynomials, which should provide better possibilities not only for creating distinguishers but also for analyzing the importance of isolated parts of tested function's output the distinguisher is based on.

Different heuristics like simulated annealing can be used for the mutation of a single individual, which may provide a better success rate or faster convergence than the currently used hill-climbing technique with a stable mutation probability. We are also working on the interpretation of multi-individual settings to be able to use the full potential of genetic algorithms for TEA analysis.

Acknowledgment

We acknowledge the support of Czech Science Foundation, project GA16-08565S. The access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated.

References

[1] A. Rukhin *et al.*, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications", *NIST Special Publication 800-22rev1a*, 2010. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf (visited on 2015-12-29).

[2] G. Marsaglia. (1995). Diehard battery of tests of randomness, Floridan State University, [Online]. Available: http://stat.fsu.edu/pub/diehard/ (visited on 2015-12-29).

[3] R. G. Brown. (2004). Dieharder, A Random Number Test Suite. version Version 3.31.1, Duke University Physics Department, [Online]. Available: http://www.phy.duke.edu/~rgb/General/dieharder.php (visited on 2015-12-29).

[4] P. L'Ecuyer and R. Simard, "Testu01: a c library for empirical testing of random number generators", *ACM Trans. Math. Softw.*, vol. 33, no. 4, Aug. 2007, ISSN: 0098-3500. DOI: 10.1145/1268776.1268777. [Online]. Available: http://doi.acm.org/10.1145/1268776.1268777 (visited on 2015-12-29).

[5] P. Švenda, M. Ukrop, M. Sýs, *et al.* (2012). Eacirc, Framework for autmatic search for problem solving circuit via evolutionary algorithms, Centre for Research on Cryptography and Security, Masaryk University, [Online]. Available: https://github.com/crocs-muni/EACirc (visited on 2015-12-29).

[6] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, "Genetic Programming: An Introduction, On the Automatic Evolution of Computer Programs and Its Applications", 1997.

[7] D. J. Wheeler and R. M. Needham, "TEA, a tiny encryption algorithm", in *Fast Software Encryption*, Springer, 1995, pp. 363–366.

[8] J. Kelsey, B. Schneier, and D. Wagner, "Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA", ICICS '97, pp. 233–246, 1997. [Online]. Available: http://dl.acm.org/citation.cfm?id=646277.687180 (visited on 2015-12-29).

[9] J. C. Hernández, J. M. Sierra, P. Isasi, and A. Ribagorda, "Genetic Cryptoanalysis of Two Rounds TEA", in *Computational Science—ICCS 2002*, Springer, 2002, pp. 1024–1031.

[10] J. C. Hernández and P. Isasi, "Finding Efficient Distinguishers for Cryptographic Mappings, with an Application to the Block Cipher TEA", *Computational Intelligence*, vol. 20, no. 3, pp. 517–525, 2004.

[11] A. Garrett, J. Hamilton, and G. Dozier, "A Comparison of Genetic Algorithm Techniques for the Cryptanalysis of TEA", *International journal of intelligent control and systems*, vol. 12, no. 4, pp. 325–330, 2007.

[12] W. Hu *et al.*, "Cryptanalysis of TEA Using Quantum-Inspired Genetic Algorithms", *Journal of Software Engineering and Applications*, vol. 3, no. 01, p. 50, 2010.

[13] E. Y.-T. Ma and C. Obimbo, "An evolutionary computation attack on one-round tea", *Procedia Computer Science*, vol. 6, pp. 171–176, 2011.

[14] M. Ukrop, "Usage of evolvable circuit for statistical testing of randomness", bachelor thesis, Faculty of Informatics Masaryk University, 2013. [Online]. Available: https://is.muni.cz/th/374297/fi_b/ (visited on 2015-12-29).

[15] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*, 3rd ed. CRC Press, 2003, ISBN: 9781420036268.

[16] Ľ. Obrátil, "Automated task management for BOINC infrastructure and EACirc project", bachelor thesis, Faculty of Informatics Masaryk University, 2015. [Online]. Available: https://is.muni.cz/th/410282/fi_b/ (visited on 2015-12-29).

[17] D. P. Anderson *et al.* (2015). BOINC project, [Online]. Available: https://boinc.berkeley.edu/ (visited on 2015-12-29).

[18] M. Sýs, P. Švenda, M. Ukrop, and V. Matyáš, "Constructing empirical tests of randomness", 2014. [Online]. Available: http://dx.doi.org/10.5220/0005023902290237 (visited on 2015-12-29).

[19] Team Czech NGI. (2015). Metacentrum – Virtual Organization of the Czech National Grid Organization, [Online]. Available: https://metavo.metacentrum.cz/ (visited on 2015-12-29).

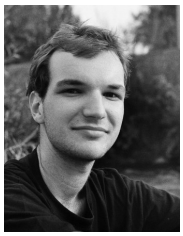
[20] Nano-Optics groups (Department of Physics) and PicoQuant GmbH. (2010). High bit rate quantum random number generator service, Humboldt University of Berlin, [Online]. Available: http://qrng.physik.hu-berlin.de/ (visited on 2015-12-29).

[21] P. Švenda, M. Ukrop, and V. Matyáš, "Towards cryptographic function distinguishers with evolutionary circuits", in *SECRYPT*, Centre for Research on Cryptography and Security, Masaryk University, 2013, pp. 135–146. [Online]. Available: http://dx.doi.org/10.5220/0004524001350146 (visited on 2015-12-29).

Karel Kubíček Master student at Masaryk University, Brno, Czech Republic in field Security of Information Technology. Involved in Center for Research on Cryptography and Security since 2014.

Jiří Novotný Master student at Masaryk University, Brno, Czech Republic in the field of Parallel and Distributed Systems. Involved in Center for Research on Cryptography and Security since 2015.

Petr Švenda Assistant Professor at the Masaryk University, Brno, Czech Republic. His research focuses on the possibilities of using evolution algorithms for an analysis of cryptographic primitives. He also engages in research in the field of authentication and key establishment protocols for distributed architectures with multiple communicating parties or users, e.g. wireless sensor networks. He also analyses the practical security of cryptographic smart cards including the development of secure applications on this platform. He participated in consultations and development for academic, state and industrial organizations in the Czech Republic and abroad.



Martin Ukrop Postgraduate student at Masaryk University, Brno, Czech Republic in the field of information security. Involved in Center for Research on Cryptography and Security since 2012. His research focuses on the usage of evolutionary algorithms in security, particularly in randomness assessment. He also participates in usable security research at the aforementioned lab.

Side Channels in SW Implementation of the McEliece PKC

Marek Klein

***Abstract*—The McEliece cryptosystem is considered secure in the presence of quantum computers because there is no known quantum algorithm to solve the problem this cryptosystem is built on. However, naive implementation of the cryptosystem can open side channels, which can be used to gather information about the message or the secret key. In this paper we present results of chosen timing attacks on straightforward implementation of this cryptosystem. Furthermore, we present practical countermeasures and evaluate their efficacy.**

***Index Terms*—Side-channel attacks, timing attacks, post-quantum cryptography, code based cryptography, countermeasures.**

I. INTRODUCTION

PUBLIC key cryptography, or asymmetric cryptography, is a set of cryptographic algorithms that require two keys. One of the keys, public key, is published and everyone can use it in order to encrypt their secret. Although everybody knows how the message is encrypted, only the legitimate receiver, an owner of the second key, is able to decrypt the message. This property is widely used in the real world to secure financial transactions, to provide authenticity and in many other applications.

Security of currently most used cryptosystems, such as RSA [1], DSA or ECDSA [2], is based on the factorization of large primes or the calculation of the discrete logarithm. However, these cryptosystems are insecure in the case of existence of quantum computers, which are being actively developed these days. Therefore, several solutions have been proposed to be used instead of currently used cryptosystems. One of the candidates for post-quantum cryptography is the McEliece cryptosystem. It is based on the problem of decoding large linear codes without a visible structure. This problem belongs to the category of NP-complete problems and there is no known algorithm, solving this decoding problem in polynomial time.

In section II, we describe the McEliece cryptosystem, key generation, encryption and decryption.

In section III, we describe known attacks against the McEliece cryptosystem, and in section IV, we show that BitPunch implementation [3] is vulnerable to chosen timing side channel attacks and we present results of attacking chosen implementation.

In section V, we present practical countermeasures against chosen attacks and their efficiency.

Support by NATOs Public Diplomacy Division in the framework of "Science for Peace", Project MD.SFPP 984520, is acknowledged.
Manuscript received September 29, 2015; revised January 25, 2016.

II. THE MCELIECE CRYPTOSYSTEM

The McEliece cryptosystem [5] was introduced by Robert J. McEliece in 1978. It is a public key cryptosystem based on linear codes. As one of the first cryptosystems, it used randomization during encryption. This cryptosystem uses error-correcting codes for which there exist fast decoding algorithms, for example Goppa codes.

In the following text, algorithms for generating keys, encryption and decryption are described.

A. Key Generation

Generation of the private and public key, according to [6], is described in 1. First, it is necessary to choose domain parameters m and t, where m defines the size of the finite field $\mathbb{F}(2^m)$ and t is the number of errors that can be corrected by the Patterson algorithm 4. Then monic irreducible polynomial $g(X) \in \mathbb{F}(2^m)$ of degree t is generated. Based on elements $\alpha_0, \ldots, \alpha_{n-1}$, where $\alpha_i \in \mathbb{F}(2^m)$, and the polynomial $g(X)$, the control matrix $\mathbf{H}$ is created. The matrix $\mathbf{H}$ is computed as multiplication of matrices $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{Z}$, which are as follows:

$$\mathbf{X} = \begin{pmatrix} g_t & 0 & \cdots & 0 \\ g_{t-1} & g_t & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_1 & g_2 & \cdots & g_t \end{pmatrix}$$

$$\mathbf{Y} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_0^{t-1} & \alpha_1^{t-1} & \cdots & \alpha_{n-1}^{t-1} \end{pmatrix}$$

$$\mathbf{Z} = diag(g(\alpha_0)^{-1}, \ldots, g(\alpha_{n-1})^{-1})$$

Afterward, a random permutation $\mathbf{P}$ is generated and the control matrix $\mathbf{H}$ is permuted by the inverse permutation $\mathbf{P}^T$. This permuted matrix is then transformed from the matrix over $\mathbb{F}(2^m)$ into the matrix $\mathbf{H}_2$ over $\mathbb{F}(2)$ where elements from $\mathbb{F}(2^m)$ are transformed into column vectors from $\mathbb{F}(2)$ of length m. From matrix $\mathbf{H}_2$, a generator matrix is created for a linear code and part of this matrix is published as a public key. Private key consists of permutation $\mathbf{P}$ and polynomial $g(X)$.

Algorithm 1 McEliece-PKC Key Generation.

Require: McEliece domain parameters m and t.
Ensure: Public key $\mathbf{R}^T$ and private key $(\mathbf{P}, g(X))$.
1: Construct $\mathbb{F}(2^m) = \{\alpha_0, \ldots, \alpha_{n-1}\}$, where $n = 2^m$.
2: Generate a random monic, irreducible polynomial $g(X)$ of degree t, having coefficients in $\mathbb{F}(2^m)$ and $X \in \mathbb{F}(2^m)$.
3: Calculate the $t \times n$ control matrix $\mathbf{H}$ for the Goppa code generated by the polynomial $g(X)$.
4: Create a random $n \times n$ permutation matrix $\mathbf{P}$.
5: Calculate the permuted control matrix $\widehat{\mathbf{H}} = \mathbf{H}\mathbf{P}^T$.
6: Transform the $t \times n$ matrix $\widehat{\mathbf{H}}$ over $\mathbb{F}(2^m)$ into the $mt \times n$ matrix $\mathbf{H}_2$ over $\mathbb{F}(2)$.
7: Bring $\mathbf{H}_2$ into the systematic form $\widehat{\mathbf{G}} = [\mathbb{I}_{mt}|\mathbf{R}]$.
8: The expanded public key is the $k \times n$ matrix over $\mathbb{F}(2)$, denoted as $\mathbf{G} = \left[\mathbf{R}^T|\mathbb{I}_k\right]$.
9: **return** $\mathbf{R}^T$ and $(\mathbf{P}, g(X))$

B. Encryption

Algorithm 2 describes the encryption process. The corresponding codeword $\mathbf{c}'$ from linear code, generated by the matrix $\mathbf{G}$, is computed for the message $\mathbf{m}$. This codeword is then encrypted by adding the error vector with t nonzero entries.

Algorithm 2 McEliece-PKC Encryption.

Require: k-bit plain text $\mathbf{m}$, public key $(\mathbf{G}, t)$.
Ensure: n-bit cipher text $\mathbf{c}$.
1: $\mathbf{c}' = \mathbf{mG}$
2: Generate the n-bit error vector $\mathbf{e}$ such that $hwt(\mathbf{e}) = t$.
3: $\mathbf{c} = \mathbf{c}' \oplus \mathbf{e}$
4: **return** $\mathbf{c}$

C. Decryption

Algorithm 3 describes decryption of the received message $\mathbf{c}$. The received message is permuted by the private permutation $\mathbf{P}$. Afterward, Patterson algorithm [4] is used to remove the error vector from the message and then the plain text is reconstructed.

Algorithm 3 McEliece-PKC Decryption.

Require: n-bit cipher text $\mathbf{c}$, private key $(\mathbf{P}, g(X))$.
Ensure: k-bit plain text $\mathbf{m}$.
1: Permute $\mathbf{c}$: $\mathbf{c}' = \mathbf{cP}$.
2: Use Patterson algorithm 4 to reconstruct the error vector $\mathbf{e}'$.
3: Permute the error vector $\mathbf{e} = \mathbf{e}'\mathbf{P}^T$.
4: Remove the error vector from the received message $\mathbf{c}' = \mathbf{c} \oplus \mathbf{e}$.
5: Reconstruct the plain text $\mathbf{m}$ from $\mathbf{c}'$.
6: **return** $\mathbf{m}$

III. TIMING SIDE-CHANNEL ATTACKS

There exist numerous different side-channel attacks on the McEliece cryptosystem. In this section, we describe such attacks, which we realized against the BitPunch implementation.

A. Attack against the Degree of the Error Locator Polynomial

Timing attack described in [7] can be executed during decryption of the received message. The attack is aimed at determining the error vector $\mathbf{e}$.

Let us assume we have a cipher text $\mathbf{c}$ and that we are looking for the corresponding plain text $\mathbf{m}$. The aim is to remove the error vector $\mathbf{e}$ from the received cipher text $\mathbf{c}$.

We try to decode the message $\mathbf{c}_i$ for all $\mathbf{e}_i$, where $i = 0, \ldots, n-1$.

During decryption it is necessary to determine the error $\mathbf{e}$ that was added to the message $\mathbf{c}'$. This error is determined by the error locator polynomial $\sigma_{\mathbf{c}}(X)$, whose degree is $deg(\sigma_{\mathbf{c}}) = hwt(\mathbf{e})$, if $hwt(\mathbf{e}) \leq t$. If $hwt(e) > t$, then $deg(\sigma_{\mathbf{c}}(X)) = t$ with probability $1 - 2^{-m}$. Therefore, evaluation time of the polynomial $\sigma(X)$ depends on the degree of this polynomial.

a) Attack description:

We only need to measure the time of evaluation of the error vector $\mathbf{e} = (\sigma_{\mathbf{c}_i}(\alpha_0), \ldots, \sigma_{\mathbf{c}_i}(\alpha_{n-1})) \oplus (1, \ldots, 1)$. As we can see, the polynomial $\sigma_{\mathbf{c}_i}(X)$ is evaluated n-times and if n is large enough then even a small difference in the degree of $\sigma_{\mathbf{c}_i}(X)$ might cause considerable time difference and therefore we can determine $\mathbf{e}$.

Let $\tau_i = T((\sigma_{\mathbf{c}_i}(\alpha_0), \ldots, \sigma_{\mathbf{c}_i}(\alpha_{n-1})) \oplus (1, \ldots, 1))$ be the time of decoding message $\mathbf{c}_i$. Put the t smallest τ_i into the set I. Then the error vector can be created as: $\mathbf{e} = \bigoplus_i \mathbf{e}_i$, for i such that $\tau_i \in I$.

b) Countermeasure:

To avoid this attack, we can artificially raise the degree of the polynomial $\sigma_{\mathbf{c}}(X)$ to t in case $deg(\sigma_{\mathbf{c}}(X)) < t$.

B. Timing Attack against Secret Permutation

Timing attack described in [8] can be used to determine the secret permutation $\mathbf{P}$. The attacker violates encryption schema by sending specific ciphertexts with only 4 errors instead of t errors.

To understand this attack, it is necessary to realize that the error locator polynomial $\sigma(X)$, determined during decryption, can be written in the following forms:

$$\sigma(X) = \sigma_t \prod_{j \in \varepsilon'} (X - \alpha_j) = \sum_{i=0}^{t} \sigma_i X^i \qquad (1)$$

where ε' is set of indexes i, for which $e'_i = 1$, i.e. those elements of $\mathbb{F}_{2^m}$ that correspond to error positions in the permuted error vector.

Authors use the ability of constructing their own cipher texts; therefore they can control the number of errors and positions of errors in the error vector $\mathbf{e}$. They decided to create an error vector $\mathbf{e}$ with the hamming weight $w < t$. Specifically, they used $w = 4$, since it is the only one offering a plain timing attack.

If $w = 4$, then $deg(\sigma(X)) = 4$. Since $deg(\sigma(X))$ is even, $deg(a(X))$ is 2. Hence, $a(X)$ provides a leading coefficient of $\sigma(X)$ and $deg(b(X)) \leq 1$. This freedom in the degree of $b(X)$ leads to two possible control flows in the decryption algorithm. One iteration in the Extended Euclidean algorithm (XGCD) (5) or zero iterations in the XGCD. These cases lead

to two different forms of $\sigma(X)$. In case of one iteration, we find $\sigma_3 \neq 0$, because $b(X) = q_1(X)$. In case of zero iterations, we find $\sigma_3 = 0$, because $b(X) = 1$.

c) Attack description:

Let $\varepsilon = \{f_1, f_2, f_3, f_4\}$ be the set of indexes of four positions of errors in the non-permuted error vector $\mathbf{e}$.

We can rewrite the equation for the error locator polynomial (Equation 1) as

$$\sigma(X) = \sigma_4 \prod_{j \in \varepsilon} \left(X - \alpha_{\mathbf{P}_j}\right), \quad (2)$$

where $\mathbf{P}_j$ is the vector notation of permutation $\mathbf{P}$. While $\mathbf{e} = \mathbf{e}'\mathbf{P}$, we can write $e_i = e'_{\mathbf{P}_i}$ for entries of vector $\mathbf{e}$.

From Equation 2, we can write the coefficient σ_3 as a function of error positions:

$$\sigma_3(f_1, f_2, f_3, f_4) = \sigma_4\left(\alpha_{\mathbf{P}_{f_1}} + \alpha_{\mathbf{P}_{f_2}} + \alpha_{\mathbf{P}_{f_3}} + \alpha_{\mathbf{P}_{f_4}}\right). \quad (3)$$

The aim is to build a set of linear equations describing the secret permutation $\mathbf{P}$. Since the attacker can construct his own cipher texts with the hamming weight $hwt(\mathbf{e}) = 4$, he can ask the decryption device to decrypt his messages and measures the timing of step 4 of Patterson algorithm 4. If the attacker concludes from the timing that the number of iterations in XGCD algorithm is zero, the attacker adds equation $\sigma_3(f_1, f_2, f_3, f_4) = 0$ into the set of equations.

The equation system can be represented as an $l \times n$ matrix, where l is the number of equations and n is the length of the code used in the McEliece cryptosystem.

Depending on the rank of the matrix, a number of entries of the permutation have to be guessed.

d) Countermeasure:

As described in [8], to avoid this attack, it is necessary to check, and if needed, manipulate the degree of $\tau(X)$, because if the number of iterations in the XGCD algorithm 5 is zero, then $deg(\tau(X)) \leq d = \lfloor t/2 \rfloor$ before the first iteration.

It is necessary to perform the test whether $deg(\tau(X)) < d$ after determining $\tau(X)$ in Patterson algorithm 4. In case $deg(\tau(X)) < d$, then $\tau(X)$ must be manipulated in such a way that $deg(\tau(X)) = t - 1$.

It is recommended to use pseudo-random values derived from the cipher text to manipulate coefficients of $\tau(X)$. In case of using truly random values, the attacker might determine that the decryption operation is not deterministic.

e) Note:

Similar attack is described in [9]. The same as above, authors construct their own ciphertexts of low hamming weights and exploit the XGCD algorithm. Moreover, they also use error vectors of higher hamming weights; therefore, they can gather more linear equations.

IV. Attacks on The BitPunch Library

In this section we present results of attacking BitPunch implementation of the McEliece cryptosystem.

Attacks were realized on the following platform:

- CPU - Intel Core i5-3230M CPU @ 2.60GHz × 4
- Architecture - 64-bit
- Operating system - Ubuntu 14.04

In order to achieve the most accurate results, Intel Hyper-Threading, turbo mode, and frequency scaling were turned off. To measure execution time, we used the RDTSC instruction [10].

A. Attack against The Degree of ELP

In this section, we present results of attacks based on the degree of the error locator polynomial, described in subsection III-A.

First, we attacked the cryptosystem as in a real-life situation. We asked for decryption of manipulated ciphertext. For each ciphertext $\mathbf{c}_i = \mathbf{c} \oplus \mathbf{e}_i$, we measured ten times of the whole decryption process and averaged these iterations. With this simple attack, we were able to reveal from 45 to 50 errors. Results of the attack are presented in Figure 1.

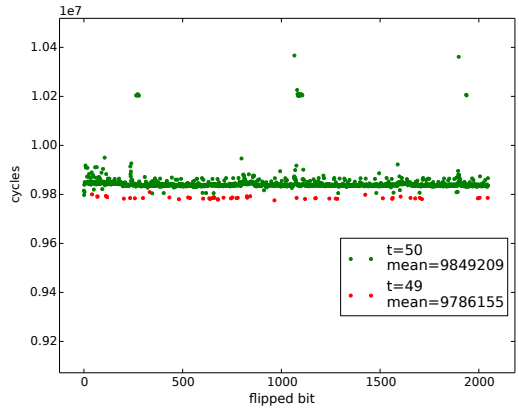


Fig. 1: Decryption times for ciphertext containing 50 error bits compared to decryption times for ciphertext containing 49 error bits.

In Figure 2, we can see decryption times corresponding to the ciphertext containing 50 error bits compared to decryption times corresponding to the ciphertext containing only 49 error bits. According to mean values and standard deviations, we can claim that the instance when the attacker revealed the bit in the error vector is easily distinguishable from when the attacker added the error bit to the ciphertext.

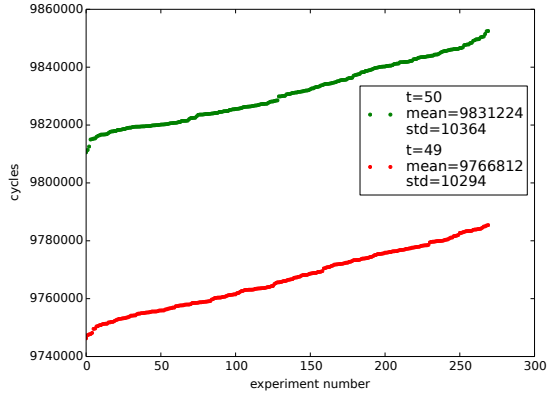


Fig. 2: Decryption times for ciphertext containing 50 error bits compared to decryption times for ciphertext containing 49 error bits.

B. Attack against Secret Permutation and Syndrome Inversion

In this section, we present results of attacks described in subsection III-B. However, to point out these time differences, we attacked the system with applied countermeasures described in section V.

This attack and the attack described in [9] are strongly tied, since they exploit the same vulnerability and time differences caused by revealing that $\sigma_3 = 0$ are added one to another. We can see this addition in Table I. This attack is required on the chosen platform for approximately 25 minutes to gain 102 equations like Equation 3, where $\sigma_3(f_1, f_2, f_3, f_4) = 0$.

	Permutation	Inversion	Decryption
$\sigma_3 \neq 0$	141169	160546	17637240
$\sigma_3 = 0$	60	95770	17416713
difference	141109	64776	220527

TABLE I: Attacks against permutation and inversion.

V. COUNTERMEASURES AGAINST ELP BASED ATTACK

In this section, we present countermeasures against attacks based on manipulation of error locator polynomial $\sigma(X)$ and their efficiency. In the following, we show the code causing timing differences in case of $t = 50$ compared to case of $t = 49$.

A. Naive implementation

In Code 1, we can see the naive implementation of determining error vector. Determination is done by evaluation of polynomial $\sigma(X)$ for each element from the support Γ. The critical operation is on line 3, which represents the evaluation of the element α_l, where $l = 0, \ldots, n-1$.

Code 1: Naive implementation.

```
1 ...
2 for (l = 0; l <
      ctx->code_spec->goppa->support_len;
      l++) {
3   tmp_eval = BPU_gf2xPolyEval(&sigma,
        ctx->math_ctx->exp_table[l],
        ctx->math_ctx);
4   if (tmp_eval == 0) {
5     BPU_gf2VecSetBit(error, l, 1);
6   }
7 }
8 ...
```

Since the aim of following countermeasures is to ensure constant execution time of Code 1, we provide graph of times needed to execute mentioned block of code in Figure 3.

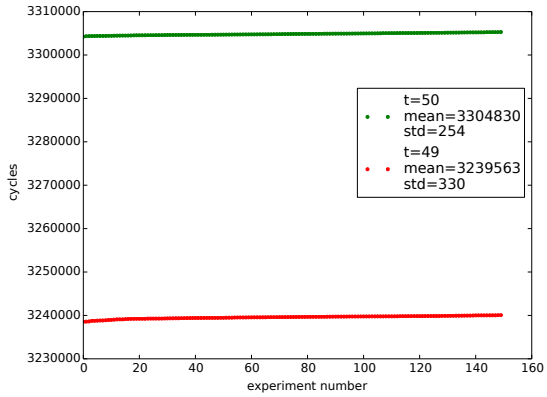


Fig. 3: Evaluation of $\sigma(X)$ without countermeasures.

B. Practical countermeasures

In Code 2, we can see the implementation of polynomial evaluation where running time directly depends on a degree of evaluated polynomial.

Code 2: Polynomial evaluation.

```
1  BPU_T_GF2_16x BPU_gf2xPolyEval(const
       BPU_T_GF2_16x_Poly *poly, const
       BPU_T_GF2_16x x, const BPU_T_Math_Ctx
       *math_ctx) {
2    int i;
3    BPU_T_GF2_16x ret = 0;
4    ret = poly->coef[0];
5
6    for (i = 1; i <= poly->deg; i++) {
7      ret = ret ^
           BPU_gf2xMulModT(poly->coef[i],
           BPU_gf2xPowerModT(x, i, math_ctx),
           math_ctx);
8    }
9    return ret;
10 }
```

Since the number of iterations in Code 2 depends on the degree of polynomial $\sigma(X)$, it is necessary to artificially increase its degree to the expected value. In this case, it is the number of errors that the decoding algorithm is capable of correcting. The degree of polynomial $\sigma(X)$ is increased at line 2 of Code 3.

However, just raising the degree of polynomial $\sigma(X)$ is insufficient. This significant time difference is caused by the *"If"* statement in line 4 of Code 1.

Code 3: Countermeasure 1.

```
1 ...
2 sigma.deg = ctx->t;
3 for (l = 0; l <
      ctx->code_spec->goppa->support_len;
      l++) {
4     tmp_eval = BPU_gf2xPolyEval(&sigma,
          ctx->math_ctx->exp_table[l],
          ctx->math_ctx);
5     BPU_gf2VecSetBit(error, l, !tmp_eval);
6   }
7 ...
```

Essential part of polynomial evaluation is multiplication of elements in the finite field. This is realized by using look-up tables, but as we can see in Code 4, this operation can differ according to its inputs. More specifically, if one of the elements a or b is 0, then the time needed to compute their product is shorter, because look-up tables are not used. This is caused by the *"if"* statement in line 3; when the condition is evaluated as true then 0 is returned immediately.

Code 4: Naive implementation of multiplication in finite field.

```
1  ...
2  BPU_T_GF2_16x BPU_gf2xMulModT(BPU_T_GF2_16x
       a, BPU_T_GF2_16x b, const
       BPU_T_Math_Ctx *math_ctx) {
3    int power;
4    if (a == 0 || b == 0)
5      return 0;
6    power = (math_ctx->log_table[a] +
         math_ctx->log_table[b]) %
         math_ctx->ord;
7    return math_ctx->exp_table[power];
8  }
9  ...
```

To avoid this difference, we decided to compute the product by look-up tables for every case and then find if one of the inputs is 0. This can be done by integer multiplication as is shown in Code Code 5, line 5. The result of the multiplication is saved in a new variable, which is, if needed, returned instead of the value computed by look-up tables.

Code 5: Multiplication in finite field with simple countermeasure.

```
1  ...
2  BPU_T_GF2_16x BPU_gf2xMulModT(BPU_T_GF2_16x
       a, BPU_T_GF2_16x b, const
       BPU_T_Math_Ctx *math_ctx) {
3    BPU_T_GF2_32x condition;
4    BPU_T_GF2_16x candidate;
5    int power;
6    power = (math_ctx->log_table[a] +
         math_ctx->log_table[b]) %
         math_ctx->ord
7    candidate = math_ctx->exp_table[power];
8    if (condition = (a * b))
9      return candidate;
10   return condition;
11 }
12 ...
```

After these modifications, time differences between evaluation times for polynomials of degree 50 and 49 are decreased, but it is still easy to distinguish between polynomials with 50 roots and polynomials with significantly less roots. It means that if attacker adds one more error to the ciphertext, then the polynomial $\sigma(X)$ is of degree 50, but it does not have 50 roots in a chosen finite field.

After the polynomial is evaluated, the appropriate bit is set to 1 if the result of evaluation is 0; otherwise, it is set to 0. This operation is executed by a macro shown in Code 6. We can see that setting the bit to 0 needs one more operation compared to setting the bit to 1.

Code 6: Naive implementation of bit setting macro.

```
1  #define BPU_gf2VecSetBit(v_pointer, i, bit)
2  if (bit) { \
3    (v_pointer)->elements[(i) /
         (v_pointer)->element_bit_size] |=
         ((BPU_T_GF2) 1) << ((i) %
         (v_pointer)->element_bit_size);\
4    } \
5    else { \
6      (v_pointer)->elements[(i) /
           (v_pointer)->element_bit_size] &=
           ((BPU_T_GF2) (0xFFFFFFFFu)) ^
           (((BPU_T_GF2) 1) << ((i) %
           (v_pointer)->element_bit_size));\
7    }
```

Countermeasure shown in Code 7 not only provides the same number of operations, but also removes branching that can be used to attack the system by power analysis.

Code 7: Bit setting macro with countermeasure.

```
1  #define BPU_gf2VecSetBit(v_pointer, i, bit)
2    (v_pointer)->elements[(i) /
         (v_pointer)->element_bit_size] &=
         ((BPU_T_GF2) (0xFFFFFFFFu)) ^
         (((BPU_T_GF2) 1) << ((i) %
         (v_pointer)->element_bit_size));\
3    (v_pointer)->elements[(i) /
         (v_pointer)->element_bit_size] |=
         ((BPU_T_GF2) bit) << ((i) %
         (v_pointer)->element_bit_size);
```

Another operation used during the evaluation of polynomial $\sigma(X)$ is BPU_gf2xPowerModT(x, i, math_ctx). This operation is used to compute the ith power of the element $x \in \mathbb{F}(2^m)$. Execution time of this operation depends on parameters x and i. If one of these parameters is 0, then execution time is shorter. To avoid using this operation, we implemented polynomial evaluation by Horner's method, described by Equation 4, which uses only multiplication:

$$\sigma(X) = \sum_{i=0}^{n} a_i X^i. \tag{4}$$

After applying the previous countermeasures, the only more complex, thus the most vulnerable operation, is multiplication of elements $X, Y \in \mathbb{F}(2^m)$. When we look at its implementation in Code 5, we can see that the same number of instructions should be used during its execution. Nevertheless, different inputs a and b cause different execution times for this block of code, more specifically, modulo operation in line 5.

Logarithmic and exponential tables are implemented in the following way:
$E[i] = \alpha^i$, where $i = 0, \ldots, 2^m - 2$ and $E[2^m - 1] = 0$.
$L[E[i]] = i$, where $i = 0, \ldots, 2^m - 2$ and $L[0] = 2^m - 1$.
Since $D = 2^m - 1$ is used as a divisor, modulo operation needs more time if a dividend $L[a] + L[b] >= D$ than in case $L[a] + L[b] < D$. If $a = 0$ or $b = 0$, then $L[a] + L[b] >= D$; therefore, zero coefficients of $\sigma(X)$ cause this time difference.

In Code 8, modulo operation is replaced by the code at lines $6-10$. This replacement of modulo operation is not only a time constant, but also faster than the previous version. Unfortunately,it is not possible to apply this countermeasure on cryptosystem where parameter $n \neq 2^m$.

Code 8: With countermeasure 6.

```
1  ...
2  BPU_T_GF2_16x BPU_gf2xMulModT(BPU_T_GF2_16x
       a, BPU_T_GF2_16x b, const
       BPU_T_Math_Ctx *math_ctx) {
3    BPU_T_GF2_16x candidate;
4    BPU_T_GF2_16x exp, bit, carry_mask = 1 <<
         math_ctx->mod_deg;
5    BPU_T_GF2_32x condition;
6    exp = math_ctx->log_table[a] +
         math_ctx->log_table[b];
7    exp = exp + 1;
8    bit = (exp & carry_mask);
9    exp = (exp & math_ctx->ord);
10   exp = (exp & math_ctx->ord) - !bit;
11   candidate = math_ctx->exp_table[exp];
12   if (condition = (a * b))
13     return candidate;
14   return condition;
15 }
16 ...
```

In Figure 4, we can see that measured times for polynomials $\sigma(X)$ of degrees 50 and 49 are approximately the same.

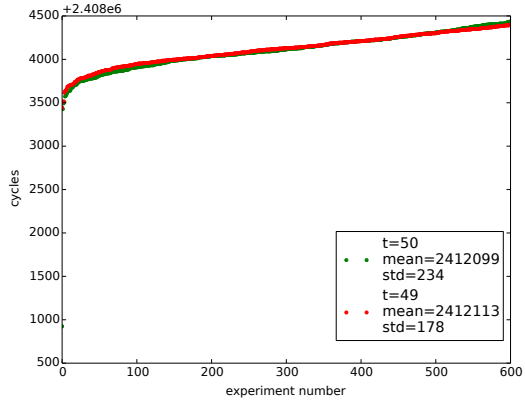


Fig. 4: Evaluation of $\sigma(X)$ - $deg(\sigma(X)) = 50$ compared to $deg(\sigma(X)) = 49$.

However, in Figure 5, it is shown that time differences between evaluation times for polynomials $\sigma(X)$ of degree 50 and 1 are still significant enough to say that algorithms are not time constant.

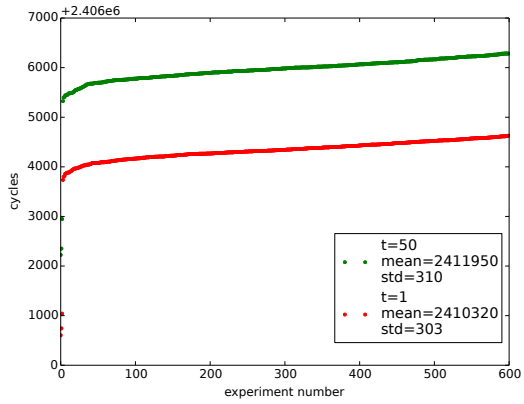


Fig. 5: Evaluation of $\sigma(X)$ - $deg(\sigma(X)) = 50$ compared to $deg(\sigma(X)) = 1$.

Since frequently used data can be stored in the CPU cache for faster access of processor to data, time differences pointed out in Figure 5 could be caused by this "caching". We decided to replace multiplication done by logarithmic and exponential tables by time constant implementation of modular arithmetic as listed in Code 9. Unfortunately, this multiplication is approximately 2.5 times slower.

Code 9: Countermeasure 7.

```
1  BPU_T_GF2_16x BPU_gf2xMulModC(BPU_T_GF2_16x
       a, BPU_T_GF2_16x b, BPU_T_GF2_16x mod,
       BPU_T_GF2_16x mod_deg) {
2    BPU_T_GF2_16x ret=0, i;
3    for(i = 0; i < mod_deg; i++) {
4      b ^= ((b >> mod_deg) & 1) * mod;
5      ret ^= ((a >> i) & 1) * b;
6      b = b << 1;
7    }
8    return ret;
9  }
```

In Figure 6, we can see that evaluation times for polynomials $\sigma(X)$ of degrees 50 and 49 are approximately the same. However, they are not exactly the same, but oscillate around the same values; see Table II.

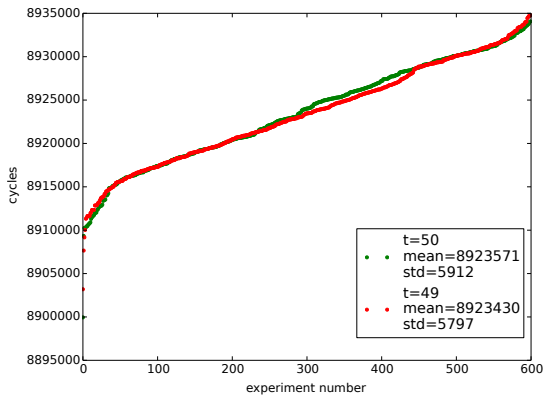


Fig. 6: Evaluation of $\sigma(X)$ - countermeasure no. 7.

	T_1	T_2	T_3	T_4
$deg(\sigma(X)) = 50$	8916516	8912341	8912524	8913037
$deg(\sigma(X)) = 1$	8917032	8911855	8912581	8913073
Difference	-516	486	-57	-36

TABLE III: Evaluation times of $\sigma(X)$ of degree 50 and 1.

VI. CONCLUSION

Proposed countermeasures should avoid the attack described in subsection III-A in a way in which it is not possible to distinguish if attacker guessed the correct position of bit in the error vector or not. On the other side, these countermeasures slow down the evaluation of polynomial $\sigma(X)$. This secured code needs 3 times longer time than naive implementation, where the biggest difference is caused by multiplication in finite field. This operation can be easily implemented in hardware; therefore, we suggest to construct a hybrid implementation of the McEliece cryptosystem. Hybrid implementation could use hardware implementation of time critical operations and software implementation of higher logic.

REFERENCES

[1] Rivest R. L., Shamir A., and Adleman L., "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, 1978, pp. 120-126.
[2] National Institute of Standards and Technology, "FIPS PUB 186-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Digital Signature Standard (DSS)," 2013.
[3] Gulyás A., Klein M., Kudláč J., Machovec F., and Uhrecký F., "Reálna implementácia code-based cryptography," Unpublished master's project, Slovak University of Technology in Bratislava, Slovakia, 2014.
[4] Patterson N., "The algebraic decoding of Goppa codes," IEEE Transactions on Information Theory 21, 2, 1975, pp. 203-207.
[5] McEliece R. J., "A public-key cryptosystem based on algebraic coding theory," DSN progress report, Vol. 42-44., 1978, pp. 114-116.
[6] Shoufan A., et al., "A novel processor architecture for McEliece cryptosystem and FPGA platforms," In Proceedings of the 2009 20th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP '09), IEEE Computer Society, 2009, pp. 98-105.
[7] Strenzke F., Tews E., Molter H. G., Overbeck R., and Shoufan A., "Side channels in the mceliece PKC," In Proceedings of the Second International Workshop, Post-Quantum Cryptography, 2008, pp. 216-229.
[8] Strenzke F., "A timing attack against the secret permutation in the mceliece PKC," In Proceedings of the Third international conference on Post-Quantum Cryptography (PQCrypto'10), Springer-Verlag, Berlin, Heidelberg, 2010, pp. 95-107.
[9] Strenzke F., "Timing attacks against the syndrome inversion in code-based cryptosystems," In Proceedings of the Fifth International Conference on Post-Quantum Cryptography - PQCrypto 2013, pp. 217-230.
[10] Paoloni G., "How to Benchmark Code Execution Times on Intel IA-32 and IA-64 Instruction Set Architectures," White Paper, 2010.

APPENDIX A
ALGORITHMS

Algorithm 4 Patterson Algorithm.

Require: n-bit word $\mathbf{c}$, Goppa polynomial $g(X)$.
Ensure: n-bit error vector $\mathbf{e}$.
1: Compute syndrome polynomial $S_{\mathbf{c}}(X) = \mathbf{cH}^T \left(X^{t-1}, \ldots, X, 1\right)^T$, where $\mathbf{H}$ is control matrix for Goppa code generated by polynomial $g(X)$.
2: Invert $S_{\mathbf{c}}^{-1}(X)$.
3: Let $\tau(X) = \sqrt{S_{\mathbf{c}}^{-1}(X) + X}$.
4: Find polynomials $a(X)$ and $b(X)$, so that $b(X)\tau(X) = a(X) \mod g(X)$, and $deg(a) \leq \lfloor \frac{t}{2} \rfloor$.
5: Determine error locator polynomial $\sigma(X) = a^2(X) + xb^2(X)$, where $deg(\sigma) \leq t$.
6: Reconstruct the error vector $\mathbf{e} = (\sigma(\alpha_0), \ldots, \sigma(\alpha_{n-1})) \oplus (1, \ldots, 1)$.
7: **return e**

Algorithm 5 Extended Euclidean Algorithm.

Require: $\tau(X), g(X), d_{break}$
Ensure: $a(X), b(X)$ such that $b(X)\tau(X) = a(X) \mod g(X)$ and $deg(a) \leq d_{break}$
$r_{-1}(X) = g(X)$
$r_0(X) = \tau(X)$
$b_{-1}(X) = 0$
$b_0(X) = 1$
$i = 0$
while $deg(r_i) > d_{break}$ **do**
    $i = i + 1$
    $q_i(X) = r_{i-2}(X) / r_{i-1}(X)$
    $r_i(X) = r_{i-2}(X) \mod r_{i-1}(X)$
    $b_i(X) = b_{i-2}(X) + q_i(X) b_{i-1}(X)$
$a(X) = r_i(X)$
$b(X) = b_i(X)$
return $a(X)$ and $b(X)$

Marek Klein received his Bc. degree in Modeling and Simulation of Event Systems and Ing. degree in Security of Information Technologies from Slovak University of Technology in Bratislava in 2013 and 2015 respectively. He currently works as developer at Disig, a.s. in the Department of Experimental Development.

Cryptanalysis based on the theory of symmetric group representations

Romana Linkeová, Pavel Příhoda

***Abstract*—The key exchange Diffie-Hellman protocol originally works over the group $\mathbb{Z}_p^*$ where p is at least a 300-digit number. Even though this implementation is simple and secure, it makes the protocol unsuitable for devices with limited computational power. This fact led to a research of other algebraic structures which could be used as a platform for this protocol in order to decrease the computational and storage costs. Such attempt can be found in the work of D. Kahrobaei et al. posted in 2013. D. Kahrobaei et al. proposed a structure of small matrices over a group ring as a platform and claimed that this modification will not affect the security of the Diffie-Hellman protocol. We will attack this modification and prove that it is not secure with the help of the theory of symmetric group representations.**

***Index Terms*—Diffie-Hellman protocol, public key cryptography, symmetric group representations.**

I. Introduction

ONE of the requirements of symmetric cryptography is that two communicating parties are able to establish a secret shared key over a public channel without anyone else being able to retrieve their shared key from the communication as well. One of the cryptographic tools that solves this problem is the Diffie-Hellman protocol which was introduced by Witfield Diffie and Martin Hellman in 1976 in [2].

One of the drawbacks of this protocol is that it does not ensure the authentication of both parties. This fact makes the protocol vulnerable against the man-in-the-middle attack.

Another drawback is that using $\mathbb{Z}_p^*$ (the multiplicative group of integers modulo prime p, where p is suggested to have at least 300 digits), makes the protocol being unsuitable for devices with limited computational power. In order to decrease computational costs, we can exchange $\mathbb{Z}_p^*$ for another algebraic structure. One of the approaches that is trying to do so can be found in [6], where D. Kahrobaei et al. proposed a semigroup of $n \times n$ matrices over the group ring $\mathbb{F}_q[\boldsymbol{S}_m]$ as a platform. They proposed semigroups $M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$ and $M_3(\mathbb{Z}_2[\boldsymbol{S}_5])$ specifically.

Romana Linkeová is with the Department of Algebra, Faculty of Mathematics and Physics, Charles University in Prague, Sokolovská 83, 186 75 Prague, Czech Republic e-mail: linkeovaromana@gmail.com.

Pavel Příhoda is with the Department of Algebra, Faculty of Mathematics and Physics, Charles University in Prague, Sokolovská 83, 186 75 Prague, Czech Republic, e-mail: prihoda@karlin.mff.cuni.cz.

Manuscript received September 29, 2015; revised February 17, 2016.

The main advantage of this algebraic structure is that one can precompute a multiplicative table for elements from $\boldsymbol{S}_5$, which makes the computations in the semigroup very time-efficient. Another advantage is that this modification of the original protocol will not, according to D. Kahrobaei et al., decrease its security. In this paper, we will show that such modification will make the protocol insecure and it will be possible to retrieve the secret shared key within few hours using a common computer.

The security of the Diffie-Hellman key exchange protocol is based on the absence of an algorithm capable of solving the discrete logarithm problem in polynomial time. Nowadays, we are aware of multiple algorithms that are solving the discrete logarithm problem in non-polynomial time, such as baby-step giant-step, Pohlig-Hellman and Pollard's Rho (for more details see [5]). The authors of [6] claimed that those algorithms (together with the Shor's quantum algorithm) will not work for their modified protocol. In this paper, we will concentrate on the baby-step giant-step algorithm and show that it will be more effective than D. Kahrobaei et al. claimed.

Firstly, we will describe the necessary algebraic theory. Secondly, we will focus on the description of the original and the modified Diffie-Hellman protocol. Then, we will present the attack itself. After that, we will show that the baby-step giant-step algorithm will work on the modified protocol. The next section is focused on implementation of our attack. Lastly, we will present a list of papers that also proposed an attack on the modified protocol.

II. Definitions and notations

Definition 1 (Group ring)**.** *Let $G = (G, *, ^{-1}, e)$ be a finite group and let $R = (R, +, \cdot, -, 0_R, 1_R)$ be a ring with unity. Then a group ring $R[G]$ is the set of all formal sums*

$$\sum_{g \in G} r_g g,$$

where $r_g \in R$.

For $u = \sum_{g \in G} a_g g$, $v = \sum_{h \in G} b_h h$, $u, v \in R[G]$, $a_g, b_h \in R$. The addition $u \oplus v$ and multiplication $u \otimes v$ is defined as follows:

$$u \oplus v = \sum_{q \in G} (a_q + b_q) q,$$

$$u \otimes v = \sum_{q \in G} \left(\sum_{gh = q} a_g b_h \right) q.$$

Definition 2 (Period). *Let M be a square matrix. The least $k \in \mathbb{N}$ such that $M^i = M^{i+k}$, for some $i \in \mathbb{N}$ is called the period of M.*

Definition 3 (Pre-period). *Let M be a square matrix. The least $r \in \mathbb{N}$ such that there exists $k \in \mathbb{N}$ that $\forall i \geq r$, $M^{k+i} = M^i$ is called the pre-period of M.*

Definition 4 (Representation). *a representation of a group G of degree n is a homomorphism $\varphi : G \to GL(n,T)$; $\varphi(g) = \varphi_g$ for $g \in G$.*

Definition 5 (Equivalent representations). *Two representations $\varphi : G \to GL(n,T)$ and $\psi : G \to GL(m,T)$ are equivalent if $m = n$ and if $F \in GL(n,T)$ such that $\psi_g = F\varphi_g F^{-1}$, $\forall g \in \boldsymbol{G}$ exists.*

Definition 6 (φ-invariant subspace). *For a representation $\varphi : G \to GL(n,T)$ a subspace $S \leq T^n$ is φ-invariant if $\varphi_g s \in S$, $\forall g \in G$, $s \in S$.*

Definition 7 (Irreducible representation). *a representation $\varphi : G \to GL(n,T)$ is irreducible if and only if φ-invariant subspaces of T^n are $\{0\}$ and T^n.*

Definition 8 (Partition of number n). *Let $n \in \mathbb{N}$, then the partition λ of number n is defined as a non-increasing sequence of m positive integers $\lambda = (\lambda_1, \ldots, \lambda_m)$ such that $\lambda_1 + \ldots + \lambda_m = n$. We denote $\lambda \vdash n$.*

Theorem 1. *For $n \in \mathbb{N}$ and a field T of characteristics 0 or p, where p is a prime and $p > n$:*

- *each $\lambda \vdash n$ gives representation $\varphi^\lambda : S_n \to GL(n_\lambda, T)$ (for more details see [3, Theorem 4.12]),*
- *φ^λ is irreducible representation for all $\lambda \vdash n$,*
- *$\lambda \vdash n$, $\eta \vdash n$, $\eta \neq \lambda$, then φ^λ and φ^η are not equivalent,*
- *each irreducible representation of S_n over T is equivalent to some representation φ^λ,*
- *$TS_n \simeq \prod_{\lambda \vdash n} M_{n_\lambda}(T)$.*

III. Diffie-Hellman protocol

A. Discrete logarithm

Let $G = \langle g \rangle$ be a finite cyclic group of order n. Then for all elements $b \in G$ exists one and only one x in interval $(0, \ldots, n-1)$ such that $b = g^x$. The number x is called the *discrete logarithm* of element b in G. The task to compute x when G, g and b are given is called the *discrete logarithm problem*. We are not aware of any general method that could solve the discrete logarithm problem on a common computer in sub-exponential time.

B. Original Diffie-Hellman protocol

The requirement that two parties should be able to construct a secret shared key over a public channel resulted in the introduction of the Diffie-Hellman protocol in 1976. This protocol describes an exchange between two parties A and B leading to establishment of a secret shared key. Only A and B possess the key and it can not be retrieved by anyone who is listening to their conversation. The security of this protocol is based on the difficulty of the discrete logarithm problem.

The protocol works as follows:

- A and B decide on a finite cyclic group G and its generating element g,
- A picks a secret number $a \in (0, \ldots, |\, G \,| - 1)$ and sends $u = g^a$ to B,
- B picks a secret number $b \in (0, \ldots, |\, G \,| - 1)$ and sends $v = g^b$ to A,
- A computes $v^a = (g^b)^a = g^{ab}$,
- B computes $u^a = (g^a)^b = g^{ba}$,
- both A and B are in possession of the secret shared key g^{ab}.

Both A and B are using the algorithm *square and multiply* when computing g^a, g^b and g^{ab}.

An eavesdropper E, who is trying to retrieve the secret shared key g^{ab} from the knowledge of G, g, g^a, g^b, g^{ab}, is trying to solve the so called *Diffie-Hellman problem*.

The simplest and original implementation of the Diffie-Hellman protocol uses the class of groups $\mathbb{Z}_p^*$ as a platform. Working over groups from this class is convenient since we can easily calculate powers of its elements. Moreover, no fast algorithm that could solve the discrete logarithm problem in those groups is known. Nowadays, the protocol is considered secure, if p is at least a 300-digit number and a and b are at least 100-digit numbers. Unfortunately, these sizes of the parameters do not make this protocol suitable for devices with limited computational power. D. Kahrobaei et al. proposed a semigroup of small matrices as a platform for the protocol. Multiplication is fast in this structure hence the protocol is not that time consuming.

C. Modified Diffie-Hellman protocol

The structure proposed to work with is a semigroup of small matrices over the group ring $\mathbb{Z}_p[S_m]$ where $\mathbb{Z}_p$ is the ring of integers modulo p and S_m is the symmetric group of order $m!$. Parameters proposed in [6] are 3×3 matrices over $\mathbb{Z}_7[S_5]$ or over $\mathbb{Z}_2[S_5]$.

The main advantage of this structure is that we can precompute a multiplicative table for elements from S_5; hence multiplying two elements from $\mathbb{Z}_p[S_5]$ requires only multiplying elements from $\mathbb{Z}_p$ and searching in the multiplicative table.

The modified protocol works in the case $M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$ as follows:

- A and B decides on a matrix $M \in M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$,
- A picks a secret number $a \in \mathbb{N}$ and sends M^a to B,
- B picks a secret number $b \in \mathbb{N}$ and sends M^b to A,
- A computes $(M^b)^a = M^{ba}$,
- B computes $(M^a)^b = M^{ab}$,
- both A and B are in possession of the secret shared key M^{ab}.

It is important to note that M has to be chosen properly, i.e. that it has period larger than 10^{10}. Otherwise the attacker E could retrieve the secret shared key M^{ab} by

means of exhaustive search. The method to construct a suitable matrix $M \in M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$ can be found in [6].

IV. Attack

The goal of our method is to retrieve the secret shared key M^{ab} with only the knowledge of $M_n(\mathbb{F}_q[\boldsymbol{S}_m])$, M, M^a and M^b. To do so, we do not have to find both a and b. We can find a' such that $M^{a'} = M^a$. Then, the secret shared key will be $(M^{a'})^b = (M^a)^b = M^{ab}$. Let us denote $N = M^a$.

The core idea of our method is that we used the representation theory of symmetric groups, which allows us to reduce the work an attacker has to do. We know that in order to break the Diffie-Hellman problem in $M_n(\mathbb{Z}_p[\boldsymbol{S}_m])$ we would have to solve the discrete logarithm problem in this semigroup in a reasonable amount of time. Since the order of $M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$ is approximately 10^{913}, we see it is not possible. However, the representation theory of the symmetric groups allows us to transform the problem onto a structure in which we are able to calculate the discrete logarithms in feasible time.

Firstly, we will describe our method for the case of $M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$. Secondly, we will introduce the approach that solved the challenge given in [6] when using $M_3(\mathbb{Z}_2[\boldsymbol{S}_5])$ as a platform.

A. Case $M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$

The characteristics of $\mathbb{Z}_7$ does not divide the order of S_5, so the theory of symmetric group representations gives us 7 irreducible representations φ_i, $i \in \{1, \ldots, 7\}$, i.e. homomorphisms

$$\varphi_i : \boldsymbol{S}_5 \to \mathrm{GL}(d_i, \mathbb{Z}_7),$$

where $d_i \in \{1, 4, 5, 6, 5, 4, 1\}$.

We can extend the homomorphisms in two steps as follows:

$$\varphi_i' : \mathbb{Z}_7[\boldsymbol{S}_5] \to M_{d_i}(\mathbb{Z}_7)$$

and

$$\psi_i : M_3(\mathbb{Z}_7[\boldsymbol{S}_5]) \to M_{3d_i}(\mathbb{Z}_7).$$

Then, according to [11, Theorem 3.9] and [4, Theorem 2.1.12], we obtain an algebra isomorphism $\psi = (\psi_1, \psi_2, \psi_3, \psi_4, \psi_5, \psi_6, \psi_7)$:

$$\begin{aligned}\psi : M_3(\mathbb{Z}_7[S_5]) \to \\ M_3(\mathbb{Z}_7) \times M_{12}(\mathbb{Z}_7) \times M_{15}(\mathbb{Z}_7) \times M_{18}(\mathbb{Z}_7) \\ \times M_{15}(\mathbb{Z}_7) \times M_{12}(\mathbb{Z}_7) \times M_3(\mathbb{Z}_7).\end{aligned} \tag{1}$$

We can see that the maximum order of matrices we will work with is 18 which is very small.

Note that the homomorphisms φ_i', $i \in \{1, \ldots, 7\}$ can be efficiently computed (for more details see [3, Chapter 8]). Now, we can map matrices M and N using the isomorphism ψ. We get two 7-tuples

$$\psi(M) = (M_{(1)}, \ldots, M_{(7)})$$
$$\psi(N) = (N_{(1)}, \ldots, N_{(7)}).$$

To construct a', we need to find numbers $a_i \in \mathbb{Z}$ such that $M_{(i)}^{a_i} = N_{(i)}$ for all $i \in \{1, \ldots, 7\}$.

To obtain a_i, $i \in \{1, \ldots, 7\}$ we will use the Menezes-Wu algorithm which can be found in [8].

To simplify the situation, we assume that 0 is not an eigen value of any of matrices $M_{(i)}$, $i \in \{1, \ldots, 7\}$. In fact, large powers of Jordan blocks with eigenvalue 0 are zero matrices, so the simplification has no essential effect regarding the attack.

In the following part, we will demonstrate the Menezes-Wu algorithm. We fix $i \in \{1, \ldots, 7\}$ and find one a_i, since all a_i, $i \in \{1, \ldots, 7\}$ can be obtained in the same manner. Also, we will show this method for both the cases of diagonal matrices $M_{(i)}$ and $N_{(i)}$ and non-diagonal matrices $M_{(i)}$ and $N_{(i)}$.

a) Diagonalizable matrices: Matrices $M_{(i)}$ and $N_{(i)}$ are diagonalizable if and only if their characteristic polynomials decompose into the product of linear factors and algebraic multiplicity of each eigen value is equal to its geometric multiplicity. In order to ensure that the characteristic polynomial will decompose to the product of linear factors we will work over the splitting field $\mathbb{F}$ of that polynomial. Let us denote k the rank of matrices $M_{(i)}$ and $N_{(i)}$, $\lambda_1, \ldots, \lambda_k$ eigen values of $M_{(i)}$ and $u_1, \ldots, u_k$ a basis of $\mathbb{F}^k$ composed of eigen vectors of $M_{(i)}$.

Let U be an invertible matrix that has eigen vectors $u_1, \ldots, u_k$ as columns. It holds that

$$U^{-1} M_{(i)} U = \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_k \end{pmatrix}.$$

Then

$$U^{-1} M_{(i)}^a U = \begin{pmatrix} \lambda_1^a & 0 & \ldots & 0 \\ 0 & \lambda_2^a & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_k^a \end{pmatrix} = U^{-1} N_{(i)} U.$$

If we find $c \in \mathbb{N}_0$ such that $\lambda_j^c = \lambda_j^a$ for all $j \in \{1, \ldots, k\}$, it will hold that $M_{(i)}^c = N_{(i)}$ and c will be the required a_i for $M_{(i)}$ and $N_{(i)}$.

To obtain c we have to:

- find the characteristic polynomial q, eigen values $\lambda_1, \ldots, \lambda_k$ and eigen vectors $u_1, \ldots, u_k$ of $M_{(i)}$,
- equations

  $$N_{(i)} u_j = \lambda_j^a u_j, \forall j \in \{1, \ldots, k\}$$

  lead to finding $c_j \in \mathbb{N}_0$ such that $\lambda_j^{c_j} = \lambda_j^a$ for all λ_j, $j \in \{1, \ldots, k\}$. In order to find c_j, $j \in \{1, \ldots, k\}$, we have to solve the discrete logarithm problem in groups of order $\mathrm{ord}(\lambda_j)$, $j \in \{1, \ldots, 7\}$ (note that for every irreducible factor q_j of q we work in $\mathbb{Z}_7[x]/(q_j)$ where $x + (q_j)$ represents the eigen value λ_j),

- the fact that $\text{ord}(\lambda_j) \mid (c_j - c)$,$\forall j \in \{1,\dots,k\}$ allows us to put together a system of diophantine equations

$$c = c_j - \text{ord}(\lambda_j) \cdot h_j, \forall j \in \{1,\dots,k\},$$

where $h_j \in \mathbb{Z}$. By solving this system of equations (see [1, Algorithm 2.4.10]) we will get c such that $\lambda_j^c = \lambda_j^a$, for all $j \in \{1,\dots,k\}$.

So, in order to find c, we need to know the orders of eigen values λ_j, $j \in \{1,\dots,k\}$ and we have to be able to solve the discrete logarithm problem in groups of orders $\text{ord}(\lambda_j)$, $j \in \{1,\dots,k\}$.
For a fixed $j \in \{1,\dots,7\}$ we find the order of eigen value λ_j using the fact that $\text{ord}(\lambda_j)$ divides $\mid T_r^* \mid$, where T_r denotes field $\mathbb{Z}_7(\lambda_j) \simeq \mathbb{Z}_7[x]/(q_j)$, where q_j is the minimal polynomial of λ_j in $\mathbb{Z}_7$. Computing $\text{ord}(x)$ in $(\mathbb{Z}_7[x]/(q_j))^*$ gives us orders of all roots of polynomial q_j.
When computing the discrete logarithm in groups of orders $\text{ord}(\lambda_j)$, consider $\text{ord}(\lambda_j) =\mid T_r^* \mid$ represents the worst case. Denote

$$|T_r^*| = s_1^{l_1} \cdot s_2^{l_2} \cdot \ldots \cdot s_n^{l_n}$$

the factorization of the order of T_r^*. Now we can use the Pohlig-Hellman reduction and reduce the computations into group orders of which will be at most $s_m^{l_m}$ for some $m \in \{1,\dots,n\}$. Also, for a polynomial q of degree d we have $\mid T_r^* \mid = 7^d - 1$. Since $M_{(i)}, i \in \{1,\dots,7\}$ have degrees at most 18, then $d \leq 18$ and prime factors of $7^d - 1$ for $d \leq 18$ are small enough for us to be able to solve the discrete logarithm problems in a reasonable amount of time when using a common computer.

b) Non-diagonalizable matrices: We will outline the method for non-diagonalizable matrices in this section.
Let us fix $i \in \{1,\dots,7\}$. Assume that we have a basis B such that $[H]_B$ is a matrix H expressed in terms of the basis B which has a Jordan canonical form.
Let us have an invertible matrix U which has vectors of basis B as columns. Then it holds that

$$U^{-1}M_{(i)}U = \begin{pmatrix} J_1 & 0 & \dots & 0 \\ 0 & J_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & J_k \end{pmatrix}$$

is a block diagonal matrix with Jordan blocks J_j, $j \in \{1,\dots,k\}$ on diagonal and

$$U^{-1}N_{(i)}U = \begin{pmatrix} J_1^a & 0 & \dots & 0 \\ 0 & J_2^a & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & J_k^a \end{pmatrix}.$$

Now, we can find $c_j \in \mathbb{N}_0$ for each Jordan block J_j, $j \in \{1,\dots,k\}$ such that $J_j^{c_j} = J_j^a$.

Since for Jordan block

$$J_j = \begin{pmatrix} \lambda & 1 & 0 & & 0 & 0 \\ 0 & \lambda & 1 & \dots & 0 & 0 \\ 0 & 0 & \lambda & & 0 & 0 \\ & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \lambda & 1 \\ 0 & 0 & 0 & \dots & 0 & \lambda \end{pmatrix}$$

we have

$$J_j^a = \begin{pmatrix} \lambda^a & \binom{a}{1}\lambda^{a-1} & \binom{a}{2}\lambda^{a-2} & \dots & \binom{a}{k-1}\lambda^{a-k+1} \\ 0 & \lambda^a & \binom{a}{1}\lambda^{a-1} & \dots & \binom{a}{k-2}\lambda^{a-k+2} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda^a & \binom{a}{1}\lambda^{a-1} \\ 0 & 0 & \dots & 0 & \lambda^a \end{pmatrix},$$

and c_j has to ensure equality of all elements in upper triangular matrices J_j^a and $J_j^{c_j}$. Note that if J_j^a and $J_j^{c_j'}$ have same values on the diagonal, we may find c_j of the form $c_j' + z \cdot \text{ord}(\lambda)$ for $z \in (0,1,...,6)$.
Then, we can compute $c \in \mathbb{N}_0$ such that $J_j^c = J_j^a$, $j \in \{1,\dots,k\}$ and c will be the required a_i for $M_{(i)}$ and $N_{(i)}$.
Before we proceed to the construction of a', we will show how to compute the period of $M_{(i)}$ for some $i \in \{1,\dots,7\}$, first.

c) Matrix period: Assuming that $M_{(i)}$ is diagonalizable we have its Jordan canonical form

$$C = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_k \end{pmatrix}.$$

We can see that all Jordan blocks are of degree 1 and the Jordan canonical form of an m^{th} power of $M_{(i)}$ is

$$C^m = \begin{pmatrix} \lambda_1^m & 0 & \dots & 0 \\ 0 & \lambda_2^m & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_k^m \end{pmatrix}.$$

This means that for $M_{(i)}$ it holds that

$$\text{per}(M_{(i)}) = \text{LCM}(\text{ord}(\lambda_1),\dots,\text{ord}(\lambda_k)).$$

Assuming that $M_{(i)}$ is non-diagonalizable, it holds that

$$\text{per}(M_{(i)}) = \text{LCM}(\text{per}(J_1),\dots,\text{per}(J_k)),$$

where J_j, $j \in \{1,\dots,k\}$ are Jordan blocks of non-zero eigen values of $M_{(i)}$. The periods of the Jordan blocks can be found using the following method.
Fixing $j \in \{1,\dots,k\}$, we denote p_j the period of the Jordan block J_j and λ_j the element on its diagonal. Then, it has to hold that

$$p_j = k \cdot \text{ord}(\lambda_j)$$

for $k \in \mathbb{N}$.
Having p_j as a multiple of $\text{ord}(\lambda_j)$ will ensure that the elements on the diagonal of J_j will be the same as the

elements on the diagonal of $J_j^{p_j+1}$. Finding a fitting $k \in \mathbb{N}$ will ensure that all elements above the diagonal of $J_j^{p_j}$ will be zero and we will get $J_j = J_j^{p_j+1}$.
Initialize $p_j = \mathrm{ord}(\lambda_j)$ and split the calculation of k in a few cases. It is important to keep in mind that we are computing over a field with characteristic 7, so that all operations with integers are modulo 7.

1) 7 divides p_j and the rank of J_j is at most 7:
   in this case 7 divides all binomial coefficients in $J_j^{p_j}$, hence $k = 1$ and $\mathrm{per}(J_j) = p_j$;
2) 7 divides p_j and the rank of J_j is greater than 7:
   denote $c_j = p_j/7$. A problem can appear when working with binomial coefficient $\binom{p_j}{7}$. We know that 7 divides p_j, hence

$$\binom{p_j}{7} = \frac{p_j \cdot (p_j-1)(p_j-2) \cdot \ldots \cdot (p_j-6)}{7 \cdot 6 \cdot 5 \cdot \ldots \cdot 1} = \frac{c_j \cdot (p_j-1)(p_j-2) \cdot \ldots \cdot (p_j-6)}{6 \cdot 5 \cdot \ldots \cdot 1}.$$

   For $\binom{p_j}{7} = 0$, we need $7 \mid c_j$. If 7 does not divide c_j, we set $k = 7$ which will lead to $\mathrm{per}(J_j)$ being a multiple of $7p_j$.
   It is easy to see that $7 \mid \binom{7p_j}{14}$ therefore $\mathrm{per}(J_j) = kp_j$.
3) 7 does not divide $\mathrm{ord}(\lambda_j)$ and the rank of J_j is at least 2:
   initialize $p_j = 7 \cdot \mathrm{ord}(\lambda_j)$. This case is described in cases 1 and 2.

The maximum rank of Jordan blocks is 18, so this method includes all possibilities.

d) Finding a': At this point, we have obtained $a_1, \ldots, a_7$ such that

$$(M_{(1)}^{a_1}, \ldots, M_{(7)}^{a_7}) = (N_{(1)}, \ldots, N_{(7)}).$$

We denote $p_i = \mathrm{per}(M_{(i)})$, $i \in \{1, \ldots, 7\}$. We may assume $a_1, \ldots, a_7 \geq$ pre-period(M). Then there exist constants $l_i \in \mathbb{N}_0$, $i \in \{1, \ldots, 7\}$, such that

$$a' = a_1 + l_1 p_1 = a_2 + l_2 p_2 = \ldots = a_7 + l_7 p_7. \quad (2)$$

Equation (2) can be written as a system of diophantine equations

$$a_1 + l_1 p_1 = a_2 + l_2 p_2 = \ldots = a_7 + l_7 p_7,$$

where $l_1, \ldots, l_7$ are calculated. After substituting any l_i, $i \in \{1, \ldots, 7\}$ in (2), we get a' as

$$a' = x + my,$$

where $x, y, m \in \mathbb{Z}$, and x and y are computed and m is a parameter. Choosing m such that $a' \geq a_1 \ldots, a_7$ we find the secret shared key $(M^b)^{a'} = (M^b)^a = M^{ab}$ as was required.

B. Case $M_3(\mathbb{Z}_2[\boldsymbol{S}_5])$

In this case, we can again find homomorphisms φ_i, $i \in \{1, \ldots, 7\}$

$$\varphi_i : \boldsymbol{S}_5 \to \mathrm{GL}(d_i, \mathbb{Z}_2),$$

where $d_i \in \{1, 4, 5, 6, 5, 4, 1\}$
and extend them as before

$$\varphi_i' : \mathbb{Z}_2[\boldsymbol{S}_5] \to M_{d_i}(\mathbb{Z}_2)$$

and

$$\psi_i : M_3(\mathbb{Z}_2[\boldsymbol{S}_5]) \to M_{3d_i}(\mathbb{Z}_2).$$

We get

$$\psi : M_3(\mathbb{Z}_2[S_5]) \to M_3(\mathbb{Z}_2) \times M_{12}(\mathbb{Z}_2) \times M_{15}(\mathbb{Z}_2) \times M_{18}(\mathbb{Z}_2) \times M_{15}(\mathbb{Z}_2) \times M_{12}(\mathbb{Z}_2) \times M_3(\mathbb{Z}_2).$$

However, since $\mathrm{char}(\mathbb{Z}_2) \mid \mathrm{ord}(S_5)$, the representations φ_i, $i \in \{1, \ldots, 7\}$ will not be irreducible and ψ will not be an isomorphism. Because of that we can not use the method mentioned above.
We will present a method of how to solve the challenge given in appendix of [6]. In this challenge, authors presented matrices M, M^a, M^b and asked a reader to find the secret shared key M^{ab}. To do so, we will again search a' such that $M^{a'} = M^a$.
The method works as follows:

- we calculate $\dim(\mathrm{Ker}(\psi)) = 78$ and denote $l = 128$, the smallest power or 2 greater than 78,
- using the method described in [10] we embed $M_3(\mathbb{Z}_2[\boldsymbol{S}_5])$ into $M_{360}(\mathbb{Z}_2)$ in order to calculate the pre-period y of M; note that we do not need to compute the period of M,
- we find b such that $\psi(M)^y = \psi(M)^{y+b}$ using the method mentioned in the previous section,
- this gives us a nilpotent matrix $C = M^y - M^{y+b}$,
- equation

$$0 = C^{128} = M^{128y} - M^{128y+128b}$$

  leads to finding $\mathrm{per}(M) \mid 128b$ of M,
- since $b = 75565$ then $\mathrm{per}(M) = 9672320$ which is period small enough for us to be able to find $a' = 217183$ by means of exhaustive search.

C. Implementation

To support our result, we implemented the attack in both cases $M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$ and $M_3(\mathbb{Z}_2[\boldsymbol{S}_5])$. We used Microsoft Visual C++ 2012 with NTL and MPIR libraries and Wolfram Mathematica 8.
We followed the method presented in [6] and constructed $M \in M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$ as a product $M = M_1 \cdot S$, where $M_1 \in M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$ is an invertible matrix and S is a scalar matrix with an element $s = (3+g_1)(3+g_2)(3+g_3)(3+g_4)(3+g_5)(3+g_6)(5+h)$ on its diagonal. Elements $g_i \in S_5$, $i \in \{1, \ldots, 6\}$ generate different subgroups of order 5 in S_5 and the element

h is a product of two independent cycles of lengths 2 and 3. For our particular choice of the parameters see [7, Page 15].
We picked $a = 387060858948298925004416 5641$ and obtained matrices $M_{(i)}$ and $N_{(i)}$ diagonalizable for $i \in \{1, \ldots, 7\}$ therefore we followed the method presented in IV-A0a and we got

$$a' = 350310065731473567845307248788 2159$$
$$935195562645858532491241278585 04$$
$$+414872873390037779882720801600m$$

as a result.

V. Baby-step giant-step

Nowadays, we are aware of multiple algorithms that speed up solving the discrete logarithm problem. One of them is the baby-step giant-step algorithm.
In this method, for a cyclic group $G = \langle g \rangle$ of order n and an element $b \in \langle g \rangle$, we try to find $x \in \mathbb{N}$ such that $b = g^x$ using the fact that x can be expressed as

$$x = im + j, \tag{3}$$

where $m = \lceil \sqrt{n} \rceil$, $i, j \in \{0, \ldots, m-1\}$.
Equation 3 leads to

$$b = g^{im+j} \Leftrightarrow bg^{-im} = g^j.$$

The algorithm baby-step giant-step then proceeds to so called baby-steps where it computes and stores values (j,g^j) for $j \in \{0, \ldots, m-1\}$. Baby-steps are followed by so called giant-steps which calculate values bg^{-im} for $i \in \{0, \ldots, m-1\}$ and also compare those values to stored g^j. When we hit equality

$$bx^{-im} = g^j$$

for some $i, j \in \{0, \ldots, m-1\}$, we have found $x = im + j$ such that $g^x = b$.
We can transform the situation according to [6] and work in $M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$. In this case, we have $M, N \in M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$, $n = |M_3(\mathbb{Z}_7[\boldsymbol{S}_5])|$ such that $M^x = N$ and $x \in \mathbb{N}$ can again be expressed as in (3).
In [6], the authors used an analogy of the baby-step giant-step algorithm that is based on equation

$$N = M^{im+j} \Leftrightarrow NM^{-j} = M^{im},$$

where M is a regular matrix.
Baby-steps then calculate and store values (J,NM^j) for $j \in \{1, \ldots, m-1\}$ and giant-steps calculate M^{im} for $i \in \{1, \ldots, \lceil n/m \rceil\}$ and compare them to values NM^j. When we hit equality

$$NM^j = M^{im}$$

for some $i \in \{1, \ldots, \lceil n/m \rceil\}$, $j \in \{1, \ldots, m-1\}$, we get $x = im - j$ for which $M^x = N$ holds.
Algorithm 1 shows how the baby-step giant-step works for the modified Diffie-Hellman protocol according to [6].
The algorithm requires that

Algorithm 1: Baby-step giant-step

Input: M, $N \in M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$, $n = |M_3(\mathbb{Z}_7[\boldsymbol{S}_5])|$
Output: $x \in \mathbb{N}$, such that $M^x = N$

$m = \lceil \sqrt{n} \rceil$;
$t = \lceil n/m \rceil$;
for $j = 1, \ldots, m-1$ **do**
  Compute NM^j;
  Store (j, NM^j);
for $i = 0, \ldots, t$ **do**
  Compute $M_i = M^{im}$;
  if there exists j such that $M_i = NM^j$ **then**
    return $im - j$.

$$M^{im} = NM^j \Rightarrow N = M^{im-j}$$

holds.
Since M does not have to be regular, hence invertible, it seems that this implication is not obvious. However, if we consider Jordan canonical forms, we get

$$(U^{-1}MU)^{im} = U^{-1}NU\left(U^{-1}MU\right)^j \tag{4}$$

for basis U. This can be illustrated as

$$\begin{pmatrix} \boxed{\textbf{sing}} & 0 \\ 0 & \boxed{\textbf{reg}} \end{pmatrix}^{im} =$$
$$\begin{pmatrix} \boxed{\textbf{sing}} & 0 \\ 0 & \boxed{\textbf{reg}} \end{pmatrix}^{x} \begin{pmatrix} \boxed{\textbf{sing}} & 0 \\ 0 & \boxed{\textbf{reg}} \end{pmatrix}^{j},$$

where **reg** denotes a section that appertains to nonzero eigen values and **sing** denotes a section that appertains to zero eigen values. We can see that if $N = M^x$ is large enough power of M and if m is large enough, we can illustrate (4) as follows:

$$\begin{pmatrix} \boxed{0} & \\ & \boxed{\textbf{reg}^{im}} \end{pmatrix} = \begin{pmatrix} \boxed{0} & \\ & \boxed{\textbf{reg}^{x}} \end{pmatrix} \begin{pmatrix} \boxed{?} & \\ & \boxed{\textbf{reg}^{j}} \end{pmatrix}.$$

Then

$$\begin{array}{c} M^{im} = NM^j \\ \Updownarrow \\ \textbf{reg}^{im} = \textbf{reg}^x\textbf{reg}^j \\ \Updownarrow \\ \textbf{reg}^{im-j} = \textbf{reg}^x. \end{array}$$

This means that if $im - j$ is large enough and $\textbf{sing}^{im-j} = 0$, then $M^{im-j} = N$.
According to [6], the baby-step giant-step algorithm is not usable for the modified Diffie-Hellman protocol. The main reason is that this algorithm has huge memory requirements whilst working over $M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$.
It is obvious that the knowledge of a period of M would

significantly simplify the situation. Instead of searching in the whole semigroup $M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$, we could just search in space of a size $\mathrm{per}(M)$. We have shown a method for computing the period of M in IV-A0c therefore the baby-step giant-step will be more effective than the authors of [6] claimed.

VI. Related work

The security of the modified Diffie-Hellman protocol proposed in [6] was also analysed in [10] and [9].

In [10], A. Myasnikov and A. Ushakov proposed an embedding of $M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$ into $M_{360}(\mathbb{Z}_7)$. This embedding, together with the Menezes-Wu algorithm, allowed the authors to find the secret shared key using a quantum computer in polynomial time. This paper proves that the modified Diffie-Hellman protocol does not belong to the realm of post-quantum cryptography.

In [9] can be found a method for attacking the modified protocol which is based on the same core idea as our method. The authors first constructed an embedding ψ of $M_3(\mathbb{Z}_7[\boldsymbol{S}_5])$ into $M_{360}(\mathbb{Z}_7)$ as proposed in [10] and then they constructed an isomorphism between $\mathrm{Im}(\psi)$ and $M_3(\mathbb{Z}_7) \times M_{12}(\mathbb{Z}_7) \times M_{15}(\mathbb{Z}_7) \times M_{18}(\mathbb{Z}_7) \times M_{15}(\mathbb{Z}_7) \times M_{12}(\mathbb{Z}_7) \times M_3(\mathbb{Z}_7)$. Having this isomorphism they were able to retrieve the secret shared by computing the minimal polynomial of $A \in \mathrm{Im}(\psi)$. The authors also worked with $M_3(\mathbb{Z}_2[\boldsymbol{S}_5])$ and solved the challenge given in appendix of [6]. However, our work is in scope of [7] and we worked independently from [9].

VII. Conclusion

We have recalled the modified Diffie-Hellman protocol proposed in [6] which is trying to make the original Diffie-Hellman protocol suitable for devices with limited computational power. To do so, the authors of [6] proposed $M_n(\mathbb{Z}_p[\boldsymbol{S}_m])$ as a platform. However, this modification met the computational costs requirements, it decreased the security level of the key exchange itself. We have shown that with help of the theory of symmetric group representations we can exploit the algebraic properties of $M_n(\mathbb{Z}_p[\boldsymbol{S}_m])$ and construct the secret shared key on a common computer in feasible time. The same result was presented in [9]. Consequently, the modified protocol is not as secure as is claimed in [6] when $p > m$. Any improvement of this modification to resist this attack is not clear. Our brief calculation for $m = 5$ and $p = 2$ indicates that choosing the parameters $p < m$ is probably not sufficient to make the protocol secure.

References

[1] H. Cohen, "A Course in Computational Algebraic Number Theory", 1st ed., Springer, Berlin, 1996.

[2] W. Diffie, M. E. Hellman, "New directions in cryptography", in *IEEE Transaction on Information Theory*, vol. IT-22, no. 6, pp 644-654, Nov. 1976.

[3] G. D. James, "The Representation Theory of the Symmetric Group", in *Lecture Notes in Mathematics 682*, Springer, 1978.

[4] G. D. James, A. Kerber, "The Representation Theory of Symmetric group", Cambridge University Press, 2009.

[5] A. Joux, A. Odlyzko, C. Pierrot, "The past, evolving present, and future of the discrete logarithm", in *Open Problems in Mathematics and Computational Science*, pp 5-36, 2014.

[6] D. Kahrobaei, C. Kouppari, V. Shpilrain, "Public key exchange using matrices over group rings", in *Groups, Complexity and Cryptology*, vol. 5, pp 97-115, 2013.

[7] R. Linkeová (2014, May), "Diffie a Hellman si vyměňují matice nad grupovým okruhem". [Online]. Available: https://is.cuni.cz/webapps/zzp/detail/141169/

[8] A. J. Menezes, Y. Wu, "The discrete logarithm problem in $\mathrm{Gl}_n(\mathbb{F}_q)$" in *Ars Combinatoria*, vol. 47, pp 23-32, 1997.

[9] Ch. Monico, M. D. Neusel, "Cryptanalysis of a system using matrices over group rings", in *Groups, Complexity, Cryptology*, vol. 7, pp 175-182, 2015.

[10] A. Myasnikov, A. Ushakov (2012, Oct.), "Quantum algorithm for discrete logarithm problem for matrices over finite group rings". [Online]. Available: http://eprint.iacr.org/2012/574

[11] S. H. Weintraub, "Representation Theory of Finite Groups: Algebra and Arithmetic (Graduate Studies in Mathematics)", in *Amer Mathematical Society*, vol. 59, 2003.

Romana Linkeová is a master student at Charles University in Prague. She graduated her bachelor studies in 2014. Her study field is Mathematical Methods in Information Security. You can contact her: linkeovaromana@gmail.com

Pavel Příhoda is an associate professor at Department of Algebra, Charles University in Prague where he also got his PhD in mathematics. In 2006 - 2007 he was a PostDoc researcher at Centre de Reserca Matematica, Barcelona. His research field is algebra, in particular module theory. You can contact him: prihoda@karlin.mff.cuni.cz

Competitive Programming: a Case Study for Developing a Simulation-based Decision Support System

Norbert Bátfai, *Member, IEEE,* Péter Jeszenszky, *Member, IEEE,* András Mamenyák, Béla Halász, Renátó Besenczi, János Komzsik, Balázs Kóti, Gergely Kövér, Máté Smajda, Csaba Székelyhídi, Tamás Takács, Géza Róka and Márton Ispány, *Member, IEEE*

***Abstract*—FootballAvatar is an experimental industrial research and development subproject of the project *'SziMe3D–3D technological innovation in tourism, education and sport'*. FootballAvatar aims to produce a novel decision support information system based on simulations for professional football clubs. This paper establishes the notion of football avatar in the sense of information technology, though it has a strong mathematical background. However, we would like to apply it in several analytic and simulation software tools developed in our project. The main question is that how this notion could be implemented and used in several software environments including C++, Java, and R, or from an architectural viewpoint, on desktops, smart phones, and tablets, while the kinds of uses and the base definitions have often changed during the R&D phases. This changing of the precise interpretation of the notion of "football avatar" has a direct impact on selecting the software process model. For this reason, we have developed an own software methodology called Competitive Programming (CP), which will be presented in detail, as the main result of the present paper. Our main goal with CP was to create a methodology that allows us to work effectively even when the objectives to achieve are changing rapidly. As an example of the application of the methodology, the paper discusses the aforementioned FootballAvatar project.**

***Index Terms*—Competitive Programming, Agile Programming, Software Process Improvement, OSS Policy, Football Avatars**

I. INTRODUCTION

FootballAvatar is an experimental research and development project aimed at producing the next generation of soccer analysis programs. A major innovation of this project is the simulation-based decision-making, where simulations are organized around the notion of "football avatar". Basically, it is a mathematical concept introduced in Section III-I. However, we would like to apply it in various software environments including C++, Java, and R, or, from an architectural viewpoint, on various front-end platforms, i.e., desktops, smart phones, and tablets. One major challenge was that the precise definition of "football avatar" had been changed during the R&D phases. Since refined definitions had to be used in many software components, we had to develop our own software process methodology practice called Competitive Programming (or CP for short). CP is a competition-based methodology that extends the agile development processes and is based on a combination of eXtreme Programming and Rapid Application Development (RAD). At the heart of the methodology are the creation of an initial rapid prototype and the formation of small (typically, one or two member) developer teams that work on forks of the initial prototype in competition. CP incorporates gamification elements to motivate the competition among teams. The use of free and open source software is also an important element of CP, thus it provides support to implement an open source software policy.

The paper presents the FootbalAvatar project to demonstrate an application scenario of Competitive Programming. CP will be introduced in the first part of the paper, then, in the second part the FootballAvatar system is presented in detail.

A. A Brief History of the FootballAvatar Project and Earlier Work

The idea of the FootballAvatar project was born in July 2009. Then, with the help of the Silicon Field Regional IT Cluster we had found investors and won a tender for the project. Some related works (e.g., [1], [2], and [3]) were created before the FootballAvatar project started. We do not use any of the data or software components created from these works, the FootballAvatar software system has been made from scratch.

The article [1] introduced the notion of "football avatar" using an XML-based approach. [2] and [3] investigated the RoboCup soccer simulation. Since it is an existing and well-known soccer simulation model, it was necessary to examine it before the developement process was started.

B. Technological and Methodological Background

In the scientific literature, it is not uncommon that a company customizes a well-known methodology. For example, [4] presents a study of Toyota's software development process called Toyota Production System. In general, numerous scientific publications can be found about the relationship between Software Process Improvement (SPI) and agile methodologies.

N. Bátfai is with the Faculty of Informatics, University of Debrecen, P.O. Box 12, 4010 Debrecen, Hungary, and also with SziMe3D Ltd., Debrecen, Hungary (e-mail: batfai.norbert@inf.unideb.hu).

P. Jeszenszky, M. Ispány, A. Mamenyák, R. Besenczi, J. Komzsik, B. Kóti, G. Kövér, M. Smajda, Cs. Székelyhídi and T. Takács are with the Faculty of Informatics, University of Debrecen, Hungary, and also with SziMe3D Ltd., Debrecen, Hungary.

B. Halász is with SziMe3D Ltd., Debrecen, Hungary.

G. Róka is with DVSC Futball Szervező Zrt., Debrecen, Hungary.

Manuscript received May 12, 2015. Revised: December 8, 2015.

For example, [5] introduces a mobile development oriented SPI customization.

We are committed to the Agile Manifesto [6] and we are familiar with agile software development methodologies, such as Scrum ([7], [8]) or eXtreme Programming [9]. There are many examples of customized agile methods, for example, see [10]. As another customization example, Solo Scrum [11] may be interesting for us because our SPI uses one-member development teams in many competitions (competition plays an essential role in our SPI). Our methodology also incorporates gamification [12] elements.

In recent years it has become more and more common to write programs that run parallel on the GPU, thus outperforming equivalent CPU solutions ([13], [14]). This would not have been possible without the appearance of easy-to-use APIs, such as the NVIDIA CUDA toolkit [15]. CUDA is widely used for research and simulations, such as simulating artificial neural networks [16]. We believe in this new approach to programming, and to make use of the computational power of GPUs, we also implemented our simulations using CUDA.

C. A Review of Existing Soccer Simulation Models

The whole FootballAvatar system is organized around soccer simulations. Thus, soccer simulations with their theoretical and technical background are the most characteristic feature of the system to be developed.

In our approach, a soccer simulation is (1) realistic, if it has the appearance of a real soccer match, that is, for example, there are players, who have inertia and acceleration; (2) quasi-realistic, if it can be considered to be similar to a real soccer match in some way; (3) non-realistic, where the aim of the simulation is not to reproduce the course of the game itself. For example, a match between two sophisticated 2D RoboCup Soccer Simulation (RCSS) teams, such as HELIOS [17] and WrightEagle [18], is a realistic simulation. On the other hand, FerSML [1] simulations are quasi-realistic models, because FerSML does not use any realistic kinematic models for the motions of the players. In the following, quasi-realistic simulations are also referred to as FerSML-like simulations. The Quantum Consciousness Soccer Simulation (QCSS) [19] is another example for the quasi-realistic model, but it typically can work as a non-realistic model too. Finally, soccer betting prediction is typically based on non-realistic models, for example, statistical forecasting models. Some of these models focus on predicting tournament outcomes [20] or league positions [21] while other ones are concerned with predicting outcomes of individual matches [22].

In the case of realistic and quasi-realistic simulations, we usually use the "TV criterion" to characterize the appearance of simulated matches. It is a subjective criterion introduced in [23] that checks whether the flow of play looks like a real soccer match. It should be noted that the TV criterion is entirely based on subjective opinions of human observers and therefore it cannot be quantitatively described.

At the beginning of the research process we have investigated and understood [2] the 2D RCSS model. However, the possibility of using of the 2D RCSS in FootballAvatar was rejected which was the conclusion of our previously cited work. One of the reasons of the rejection was that the project management has chosen to apply a closed-source license. The other reason was that RCSS [24] very strongly focuses on Artificial Intelligence, which is not surprising since robot soccer is a standard AI task. In this paper, we are interested only in sport science simulations.

Contrarily, FerSML is already a sport science model but the usage of this simulation model was abandoned also from the same license cause. For the same reason, we cannot use QCSS too. However, in the case of QCSS it is essential to emphasize that soccer simulation is not a determining factor, because QCSS is a cognitive model for trying to investigate the emerging human consciousness.

Finally, it should be noticed that our simulators cannot be compared with popular products of the game industry because FootballAvatar as a product is intended for experts of professional football clubs rather than lay audiences.

II. SOFTWARE PROCESS IMPROVEMENT IN FOOTBALLAVATAR

A. Project Organization

SziMe3D Ltd. is located in Debrecen, Hungary in Central Europe. It is the project company of FootballAvatar. "Nagyerdei Gerundium" is a SziMe3D working group which mainly consists of contracted researchers from the University of Debrecen, including 4 PhD doctors, 2 postgraduate researchers, 1 PhD student, and 8 BSc students. The initial idea and the essential part of the design were developed by this group. In addition, the modeling, analyzing, and simulation parts of FootballAvatar are also developed by the "Nagyerdei Gerundium".

There is a representative of SziMe3D Ltd. in the working group "Nagyerdei Gerundium" who corresponds with the Product Owner in Scrum terminology, see [7] and [8]. We are familiar with Scrum, but do not follow it, for example, "Nagyerdei Gerundium" is not a Scrum team, it is divided into smaller subgroups that overlap each other during the development. Our Software Process Improvement (SPI) will be introduced in Sect. II-B.

B. Competitive Programming

In research projects it is natural that clearly defined development targets cannot be established until we have enough experience in the area to be researched. In our case, this area has a strong mathematical flavour, simply because football avatar is a mathematical statistics-based concept (see Section III-I).

It is clear that choosing the software development methodology is an essential element for industrial projects. Taking into consideration that our project is a research project as well, where the software requirements are not clearly understood, it therefore seems appropriate to choose an agile methodology for the development of the FootballAvatar project.

In addition, it is important to note that the quintessence of the development of FootballAvatar is that it takes place in a university environment. This gives a unique opportunity for introducing some innovations in the software development

process. Since FootballAvatar is an industrial project one of the most fundamental issues is the interest of investors. However, the project is embedded in the University of Debrecen, since members of the core developer team (including all but one of the authors) who are contracted employees or apprentices of the project company are also with the university as a researcher or a student. In this environment we deeply believe in the Agile Manifesto [6], and what is more, the management of the project company also support it. The choice of this environment brings several benefits to the development. The main strength of our project is that we have the possibility to involve the best students in the software development. It is a very important aspect from the point of view of cost-efficiency, and it enables us to introduce a competition-based method for extending the agile development processes.

Fig. 1 shows the general model of our own software development process based on a combination of eXtreme Programming (XP) ([9], [25]) and Rapid Application Development (RAD) [26], where the competition among the forked rapid prototypes of XP programming pairs is focused. The first step of our approach is the creation of an initial rapid prototype to identify the major features of the application to be developed. This proto is based on user's and developer's stories presented in weekly project meetings and is typically created by a guru programmer (see Subsection II-B3). It is important to emphasize the role of the developers because they have no soccer-specific preconceptions. Even the absence of these preconceptions can allow us to create entirely new software products and services. Naturally, this has to be done in close cooperation with soccer and business experts. This iteration is a usual agile iteration which is shown by a dotted line in Fig. 1. Our competition-based agile software development practice presented in this paper is referred to as *Competitive Programming*.[1] The formal documents of this method can be found at http://footballavatar.hu/CP. Two of the most significant developer documents are the competition form and the OSS policy form that are shown in Table I and II. We maintain these forms in DocBook XML as part of our documentation process that is presented in more detail in Sect. III-J.

Table I shows the layout of our competition form. The lower part of the form supports the evaluation process that can be iterated many times depending on the result of the previous evaluation.

Table II shows the layout of our OSS submission form. The detailed description of our OSS policy process is presented in Sect. II-C.

Below we briefly survey the main competing areas, such as MABSA and FANM. The former acronym stands for MultiAgent-Based Server Architecture, the latter stands for FANM is Not MABSA. These will be detailed in sections III-E and III-F.

[1]It should be noted that the term *competitive programming* is also used broadly in the context of programming contests, where it is used to describe competitions in which participants compete with each other in solving various programming tasks [27]. There is also a Wikipedia article with the title *Competitive programming* [28] devoted to the topic. In our terminology, this term is used to denote a software development methodology.

TABLE I: Competition form

TO BE FILLED AT TEAM FORMATION	
Date of team formation	
Team name	
Member #1	
Member #2	
Supervisor of the team	
Short description of the task	
For which part of the system is the task related to	
Deadline of first iteration	
Repository location	
Names of competing teams	
Comment	
TO BE FILLED AT EVALUATION	
Date	
OSS policy verdict	approve/approve with limitations (see comment)/cancel
Meeting verdict	approve/cancel/suspend
Comment	

TABLE II: Open source software submission form

TO BE FILLED AT SUBMISSION	
Name of submitter	
Team of submitter	
Date of submission	
Name of software	
URL to obtain the software	
Type of intended usage	use as library/internal developer tool/runtime environment/content
For which parts of the system is the software intended to be used?	
Will the software be distributed with the final product?	yes/no
Name of license(s)	
Location where the license is indicated (e.g., LICENSE file included in package, web page)	
TO BE FILLED AT EVALUATION	
Person responsible for the verdict	
Verdict (approve or reject intended use)	approve/approve with limitations (see comment)/reject
Comment	

- Competing MABSA Implementations: MABSA is our internal research simulation platform which will be discussed in detail in Sect. III-E. For the development of this platform three development teams were allocated, namely: FBA One C++ (FBA is an acronym for

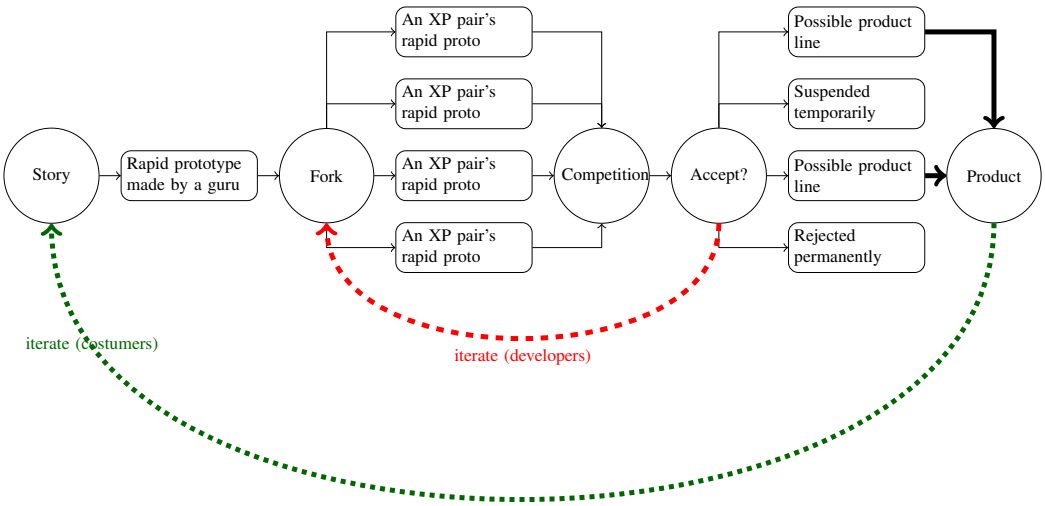


Fig. 1: A general model of the competition based agile software development process. The redundant paths (marked by bold line) are integrated into the product line.

"FootballAvatar"), Tunneled Footballers, and Hungarian Phoenix FC (see Fig. 2). At first, the development team Hungarian Phoenix (MABSA-HPFC) was suspended, and later it was cancelled. For a possible reason for this, see Sect. III-F1.

- Competing FANM Implementations: In our terminology, the term FANM stands for soccer simulation implementations that are easy to reuse across multiple applications and platforms. It will be introduced in Sect. III-F. The competing FANM development teams are shown in Fig. 3.
- Competing CUDA ports of FANM Implementations: A part of the FootballAvatar simulations can be computed on a Linux PC equipped with NVIDIA GPU. Porting FANM models to CUDA is a very challenging task but the result can be very effective. The object of the contest is to maximize the number of parallel threads of soccer matches in a CUDA block.

Finally, we note that similar competition-based methodological approaches are used in our other projects. These approaches have grown out of the first author's competition based teaching techniques. However, the idea of competing rapid prototypes is unequivocally rooted in the FootballAvatar project, because the success of the developed soccer simulation teams can be naturally measured by the results of the matches between them. In addition, we knew and understood very well from the beginning that it will be hard to find successful soccer algorithms that can reproduce the distributions observed in reality. The reason for introducing competitions into our SPI was that we wanted to support this search.

1) Incorporating Gamification Elements: Competitions can be interpreted as games between developers, where the winning itself is the direct reward of competitions. In this sense, stating the most challenging research problems and development tasks as competitions can be regarded as gamification, or rather, as ludification ([12], [29]).

As a classic gamification element, we have developed a point system to indicate the difficulty and also the monetary value of competition tasks. We represent values with quaternary (base 4) numbers, where digits are symbolized by balls. In our notation, a "classic soccer ball" denotes 0, a "silver ball" denotes 1, a "gold ball" denotes 2, and finally, a "fire ball" denotes 3.

2) Selection of the Winner: A question that must be answered is how a winner is selected from a set of competing rapid prototypes. In the case of soccer team simulation algorithms (ie., MABSA and FANM soccer teams) the winner can be naturally determined by simulating a tournament among the competing teams. In general, selecting the winner is straightforward in the case of competing simulation algorithms, ie., the winner is the one that produces results closest to reality. For this and other reasons our intuition suggests that developing simulation algorithms is such an area where the CP methodology can applied very effectively.

In other cases (eg. porting FANM soccer teams to CUDA) the goal of the competition is to minimize or maximize a predefined objective function (ie. the number of parallel threads of soccer matches in a CUDA block). If that is the case, the selection of the winner is straightforward.

On the other hand, there are cases where the winner is determined by some other mechanism. For example, the logo of the FootballAvatar project was also selected in a competition by a voting procedure (see Section III-H).

3) Our Best Practices to Set Up Competing Development Teams: As shown in Table III, we have organized competitions in ten different research and development fields that are named in the first column and will be detailed in Sections III-E, III-F, III-G and III-H. Starting from the second column, the

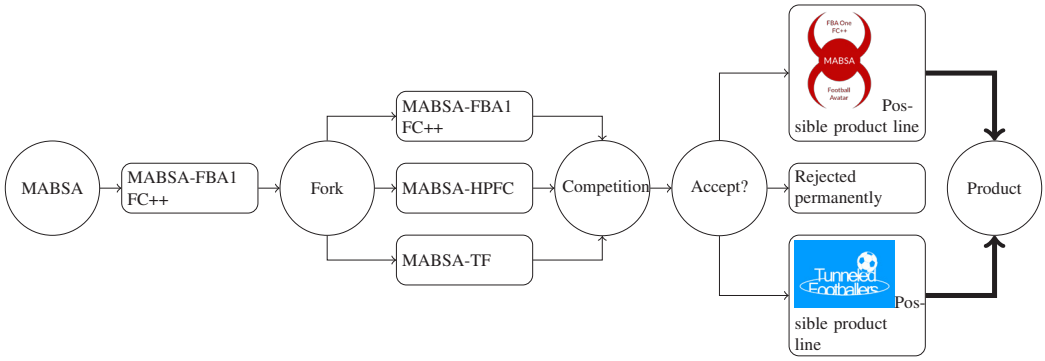


Fig. 2: An arrangement of MABSA rapid prototypes in our competition programming flow chart. MABSA acronyms are explained in detail in Sect. III-E.

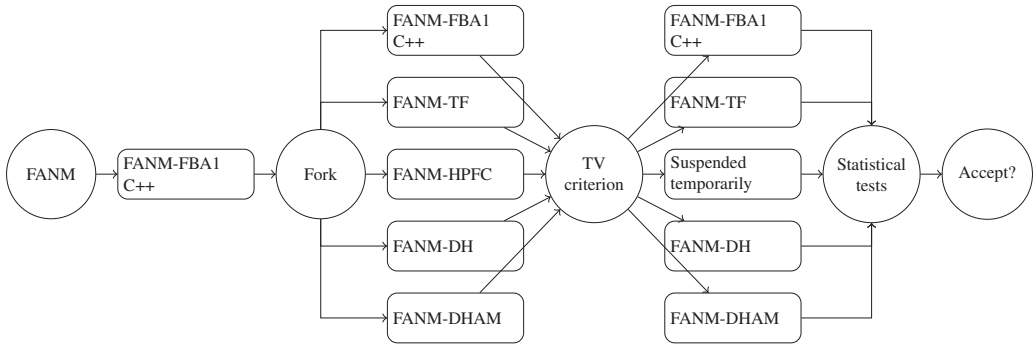


Fig. 3: An arrangement of FANM rapid prototypes in our competition programming flow chart. FANM acronyms are explained in detail in Sect. III-F.

activities of our researchers and developers are shown. In the table, an "1" denotes that the person represented by the column successfully took part in at least one competition in the field. "0" denotes non-participation, while an underlined "0" stands for unsuccessful participation.

Our experience shows that the most successful development teams consist of one or two members. All competing teams with more than two members were unsuccessful, and their activities had to be reorganized or, in several cases, had to be suspended temporarily or cancelled permanently.

In addition, our experience suggests that there is a strong correlation between taking part in the work of the successful teams and the number of commits (shown in the last row of Table III). It is not surprising, as we maintain all source code and documentation under version control. We also noticed that the activity of our researchers and developers follows a Pareto-like distribution in that sense that 80 percent of the commits were made by 20 percent of the members (see Fig. 4).

We recommend that the whole project team should include two guru programmers, two mathematicians, about 10 software engineering students, and a project owner together with further technical experts from the target area of the system to be developed (in our case, they are soccer experts). The development must be managed by the project leader who should be one of two programming gurus. The project leader should continuously monitor the activity of the others. This monitoring process is based on the number of commits per month and until it follows the early mentioned Pareto-like distribution, the project leader makes proposals to include new members in order to help, reorganize or replace members who are in the tail of the Pareto-like distribution.

C. Using Open Source Software and Licensing Issues

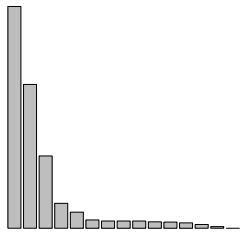
Open source software have become a key factor in the IT industry. According to a Gartner survey conducted in 2008 [30], 85% of companies already used open source software, while the remaining 15% percent expected to do so in the next 12 months. (A total of 274 companies took part in the survey from around the world.) In their more recent report [31] they expect open source software to continue to broaden its presence.

Open source has also become a business, a number of companies are specifically set up to develop and distribute open source software. Even terms such as *professional open source (POS)* [32], *OSS 2.0* [33], and *second-generation open source (OSSg2)* [34] were coined to refer to commercially developed open source software. Enterprises can also benefit a lot from using open source software to develop their own software products (for example, see [35]).

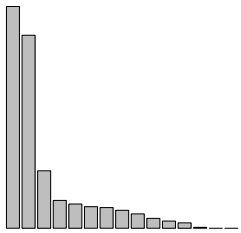
Therefore, to exploit inherent advantages we use a lot of open source software in the development of the FootballAvatar system.

TABLE III: Competition in the FootballAvatar project. The rows show different research and development fields, and the columns represent the activities of our researchers and developers, see text for detailed explanation.

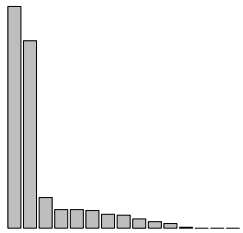
				*		*	*	*		*					
MABSA	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
FANM	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0
FANM+	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0
FANM (CUDA)	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0
2D	1	1	1	0	0	1	0	0	0	0	0	0	0	1	0
3D	1	0	0	0	0	1	0	1	0	1	0	0	1	1	0
MOBILE	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0
ANALY "0"	1	1	1	0	0	1	0	0	0	0	1	0	0	1	0
ANALY "−1"	1	1	1	0	1	1	0	0	1	0	1	0	0	0	0
BRAND	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0
# commits in $[T_1, T_2]$	835	538	270	31	14	27	23	24	16	<10	20	60	93	27	<10
# commits in $[T_3, T_4]$	250	216	64	31	14	27	23	24	16	<10	20	<10	11	<10	<10



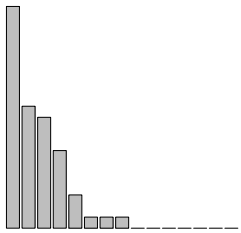
(a) The total number of commits in our repositories in a given time interval $[T_1, T_2]$.



(b) The total number of commits in our repositories in a time interval $[T_3, T_4]$, soon after when new developers joined to the project, where $[T_3, T_4] \subset [T_1, T_2]$.



(c) The number of commits in our source code repository in a given time interval $[T_1, T_2]$.



(d) The number of commits in our source code repository in a time interval $[T_3, T_4]$, soon after when new developers joined to the project, where $[T_3, T_4] \subset [T_1, T_2]$.

Fig. 4: Anonymized bar charts showing developer activity, where each bar represents the number of commits by a developer. Here the shapes of the distributions are significant, rather than the exact number of commits. In figures 4(b) and 4(d), it is shown that performance shifts slightly towards a distribution with a heavier tail after new members had been included. Our project management also uses these charts for the assessment of the work done by the developers.

1) OSS Policy: It is a common practice among enterprises to develop and maintain an *open source software policy* (or OSS policy in short), in which they specify the accepted uses of open source components and their license requirements for third-party open source software (for a comprehensive treatment of this topic, see [36]). It should be noted that, according to the Gartner survey conducted in 2008 [30], 69 percent of companies examined had no such a formal policy. The study also concluded that companies should have an OSS policy.

To assure that their licensing terms will not interfere with our closed source business model we maintain a list of all third-party software used for the development of the system. Developers have to name each third-party software they would like to use and our licensing experts decide on a case-by-case basis if their licensing terms comply with our licensing policies. If the licensing experts reject the use of a third-party software they try to suggest appropriate alternatives too.

We call the organizational process described above as *License Approval Process* (LAP). The process also includes legal consultation with third-party software vendors (see later).

2) An insight into our OSS policy: Since the majority of the system will be proprietary (non-free) software it limits the use of free and open source components. For example, any code released under a copyleft license like the GNU GPL cannot be incorporated into non-free software. Therefore, we prefer using free and open source libraries distributed under commercial friendly open source licenses, such as the Apache License, the X11 License, or the various BSD licenses. (For a comprehensive overview of the popular free and open source licenses see [36].) We use GPL'd code only as a last resort for standalone components that we make available as free and open source. The project company allows open source developments only in certain areas of FootballAvatar, but these components are independent from the simulation core. Note that the GNU GPL does not limit the use of the output of a program distributed under its terms. (See the question "In what cases is the output of a GPL program covered by the GPL too?" in [37].) Thus, we can use GPL'd software in the development of FootballAvatar. For example, we use Blender and GIMP to create content.

Some free and open source licenses require special attention

in non-free projects, one such license is the GNU LGPL. Although it is a so-called weak copyleft license that is intended to be appropriate for non-free applications, it is not without problems (for example, see [36]). Developers must permit the modification of any LGPL'd libraries they use and also the reverse engineering of their own code for debugging such modifications. Since reverse engineering is undesirable in closed source projects, we try to avoid GNU LGPL and use LGPL'd software only as a last resort when there is no appropriate alternative (an example is the Qt framework). We use LGPL'd code for free and open source components. If our non-free code must be linked with LGPL'd code we permit reverse engineering for parts of our system.

Licensing also requires special care because in one of our sales model we will distribute the FootballAvatar system in binary form bundled together with all required third-party software, even with a complete Linux operating system. Fedora has been chosen as our primary free and open source Linux distro. We consulted with the legal team of the Fedora Project and they have authorized us to distribute our system together with Fedora under the name Fedora Remix.

a) Example of Rejection: For example, the use of Ubuntu Touch as a target platform for our mobile interface was rejected. Ubuntu Touch is an Ubuntu Linux distribution for touchscreen mobile devices developed by Canonical Ltd. At present, it is experimental software for evaluation purposes only that is available free for non-commercial use. Although it is a promising mobile platform its current licensing terms are not appropriate for us since they prohibit any commercial use.

3) Types of Third-party Software: It is worth noting that we distinguish the following four category of software in our list of third-party software maintained within the LAP: (1) libraries (such as Mesa and Qt), (2) runtime environments (such as R and WordPress), (3) developer tools (such as Apache Maven and Blender), (4) and other content (such as fonts and artwork). These categories represent slightly different uses of third-party software. For example, developer tools include build automation software used by our developers to build the FootballAvatar system from sources and graphics software used by our artists to create original artwork, such as product logos. These tools don't have to be bundled with the final product.

III. An Overview of the FootballAvatar System

The results of the research and development activities of FootballAvatar can be grouped into following three basic categories:

(1) Actually, software components in the *FBA Core* category constitute the FootballAvatar system. They are distributed under a closed source proprietary license. (2) The elements of the category *FBA Add-Ons* are additional components whose terms of use are different from than those of the *FBA Core*. For example, *FBA Add-Ons* contains open source software. (3) The elements classified in category *FBA Exper* shall not be included in the FootballAvatar product, for example, because of licensing problems.

In the following, we review the FootballAvatar Soccer Simulator Collection that we also refer to as the Multi-speed Simulator.

A. A Quick Glance at the Multi-speed Simulator

The main functionality of the FootballAvatar system is organized into successive simulation levels. We refer to this layered architecture as *Simulation Oriented Architecture*, or SimOA for short. The following three main levels (or speeds) can be distinguished in the FootballAvatar Soccer Simulation Collection: (1) On the level labeled "-1" simulation algorithms use only publicly available or estimated data based on objective and/or subjective observations. (2) Level "0" uses dedicated equipment such as video cameras and sensors to gather data (this infrastructure is provided and operated by our partners). (3) Higher ("$+$") levels can be built on the lower ones.

The software elements on each layer can typically operate in the following three additional modes: standalone mode, analyzer mode, and avatar simulation mode. (1) *Standalone* simulators and analyzers operate independently on both the data provided professionally and available publicly. One important use of the standalone elements is to generate test data. (2) Software in the *analyzer* mode can be used to examine test data or real soccer data. This mode uses advanced data mining and statistical techniques such as bivariate Poisson regression models, see also [38]. (3) The *avatar simulation* mode is the basis of the comparison of real and simulated soccer matches.

The main difference between the standalone and the avatar simulations is that the former has no input at all or its input is not decisive. However, in the case of avatar simulations the representation and behaviour of the players and teams depend very heavily on the input.

In addition, from an architectural viewpoint, we distinguish the following three environments: (1) *MABSA* (Multi-Agent-Based Server Architecture) is a multi-agent system to simulate and analyze soccer matches. This environment is for research purposes only. (2) *FANM* (as opposed to MABSA) software elements neither use networking nor agent technology. These models purely focus on soccer simulation algorithms. (3) *FANM+* elements supplement FANM with additional features. For example, FANM+ contains FANM algorithms ported to Nvidia's CUDA platform.

Finally, from the aspect of implementations, we have several reference implementations, such as FBA One, Tunneled Footballers, and Hungarian Phoenix, that represent potential product lines.

In the next paragraphs, we give a detailed insight into the multi-speed simulator. It is important to emphasize that all presented models and components have been developed in the developer competition stage.

B. Speed "-1"

The level "-1" of the FootballAvatar system consists of simulation algorithms that use only publicly available or estimated data based on objective and/or subjective observations. Among others, the algorithms on this level make prediction of

the outcomes of various events that occur during matches for the coaches of a football team. These events are, for example, the result (win, draw, or lose) of a match, the number of goals, faults, yellow and red cards. It should be emphasized that we do not want to compete against betting offices by developing winning strategies for the gambling market. The main goal of the level "−1" is to help a football team providing useful information such as the comparison of the strengths of the teams who will be playing at the weekend or whether the next opponent has a taste for more faults than usual.

C. Speed "0"

Several performance metrics are used in professional football, of which the best known is the EA SPORTS Player Performance Index (PPI). According to [39], PPI is used in the Barclays Premier League to analyze the performance of players. PPI's strength is its ability to measure the performance of players independently of their playing position. It is also interesting to notice that PPI is based on a published research paper [40].

In the FootballAvatar project we are working on a conceptually similar index but its development is in a very early stage. We are experimenting with different kinds of indices. For instance, we have some promising results with the use of the Similarity Metric [41]. However, the software components in question are experimental and not part of *FBA Core*, they are assigned to *FBA Exper*. "Decision Processor" is a standard *FBA Core* coaching staff's tool in FootballAvatar that can help to classify the decisions of players in a discrete time scale. The output from this tool can be used as a basis for building performance indices. We experimented with several *FBA Exper* solutions based on the data produced by "Decision Processor".

D. Higher Speeds

Higher speeds are based on FANM simulations. Typical use cases for the higher speeds are the following: (1) Simulating full championships, such as national cups, the Champions League, or the World Cup, including the creation of match calendars (match scheduling). (2) Extending simulations by incorporating additional models, for example, physiological ones. Currently, the system uses two such extensions, a player stamina model and a foul model. The above mentioned stamina and foul models affect simulations by modifying the properties of players (also referred to as avatar properties) as a function of time. These models will be discussed in detail in a further paper that is currently under development. The modifications of the properties are implemented via AOP (Aspect-Oriented Programming) [42] aspects.

E. MABSA: MultiAgent-Based Server Architecture

MABSA is an acronym for MultiAgent-Based Server Architecture. It is a TCP/IP-based client-server framework developed in C++ from scratch by the first author. MABSA is based on the Berkeley Socket API and uses IO multiplexing to control both the server and the clients. The main feature of this architecture is that simulation algorithms can be connected to the server in the same way as client agents (see Fig. 5).

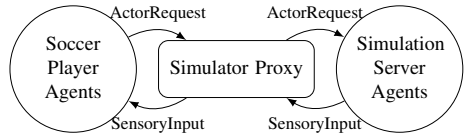


Fig. 5: The MABSA simulation architecture. Simulation algorithms are also implemented as agents, therefore they can be easily plugged in or replaced. This flexibility makes the search for good simulation algorithms easier.

MABSA soccer teams and simulation algorithms have been developed in competition. In the following, we will refer to a MABSA soccer team and a simulation algorithm together as a *MABSA implementation*. Initially, we had three competing implementations that we refer to as MABSA reference implementations.

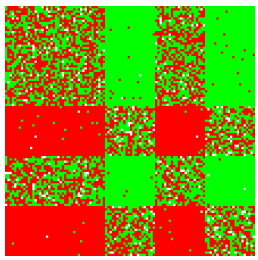
1) The MABSA Communication Protocol, Simulation Algorithms and Soccer Teams: The implementation of the communication protocol between simulation and analyzer algorithms and also simulated players is based on Google's Protocol Buffers [43]. The protocol was developed in accordance with our competition based methodology. Originally, MABSA development teams used their slightly modified versions of an initial protocol. Currently, developers use a unified protocol that represents a consensus among them and is a result of an iterative development. The protocol is organized around two communication classes, namely, *SensoryInput* and *ActorRequest*: the former encapsulates all inputs available to clients (i.e., players), while the latter their responses to their input. The exchange of *SensoryInput* and *ActorRequest* objects between agents (i.e., players and simulation algorithms) is via a simulator proxy.

a) MABSA-FBA1 FC++: It is the first author's C++-based reference implementation which has been developed as a rapid prototype. Directly or indirectly, the other development teams used it as a basis for their own MABSA implementation. MABSA-FBA1 FC++ uses a discrete-time simulation algorithm that will be detailed in a further paper.

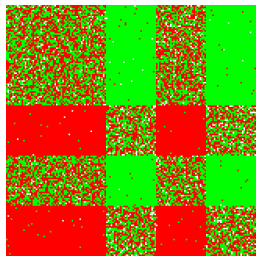
b) Hungarian Phoenix FC (HPFC): Because this development team failed to comply with the time-limits for the development of a working MABSA soccer team, it was suspended temporarily. Then, after an unsuccessful reorganization it was cancelled. The lack of success of HPFC may be explained by the "guru problem" in the sense of [44], since members had not enough experience with network programming and soccer modeling.

c) Tunneled Footballers: It is the third author's C++-based MABSA implementation. It uses a discrete-time algorithm that calculates the motion of the ball based on the Runge-Kutta method. The players can interact with the ball and with their environment with the following commands: *stand*, *move*, *kick*, *catch* and *tackle*. They also have attributes that determine the effectiveness of their actions, namely, speed, stamina, power, ball-control, dribbling, tackling.

The developer has created an own display program called



(a) This figure shows the result of 24,200 simulated matches on a GeForce GTX 560 Ti card, organized into 484 (22 × 22) threads per CUDA block.



(b) This figure shows the result of 51,200 simulated matches on a GeForce GTX 660 Ti card, organized into 1024 (32 × 32) threads per CUDA block.

Fig. 6: The Hungarian flag notation for visualizing CUDA simulation results, where a red pixel denotes a win, a white one a draw, and a green one a defeat (for the home team).

FootballEye, which can be seen in action in Fig. 10. This program serves as a testing and debugging tool for the developer.

F. FANM: FANM is Not MABSA

FANM is a recursive acronym for FANM is Not MABSA. The term denotes the second of our two architectures for soccer simulation. While the MABSA architecture primarily serves as an internal research platform, FANM is intended to be used in the end product. It supports easy embedding of simulations algorithms to the system. FANM algorithms can be ported to run in extremely parallel computing environments, such as CUDA GPUs.

1) FANM Simulation Algorithms and Teams: Similarly to MABSA, FANM implementations have also been developed in competition. We have five different competing FANM development teams, namely, FANM-FBA1 FC++, FANM-TF, FANM-HPFC, FANM-Debrecen Handsomes, and FANM-Debrecen HardAsMuscle.

a) FANM-FBA1 FC++: The name stands for a FANM reference implementation written in C++ by the first author. It is a rapid prototype that is easily portable to CUDA. Its simulation algorithm is developed from scratch and uses a FerSML-like control in that sense that it has been organized around the motion of the ball. A skeleton of the CUDA ported version of this implementation is shown in Listing 1. Fig. 6 shows our visualization technique called Hungarian flag notation, where the columns correspond to the line-ups of the away team and rows correspond to the line-ups of the home team. In this experiment both teams used the same five line-ups, but the 3rd and 5th line-ups were detuned to get better results for the home team. In this figure each pixel represents the result of a match. The figure itself reflects that the results are correct.

b) FANM-TF: It is a FANM implementation written in C++ by the third author. FANM-TF is easily portable to CUDA. Its simulation algorithm was taken from the MABSA implementation called Tunneled Footballers, but it uses neither sockets nor agents.

Listing 1: CUDA skeleton code for the FANM reference implementation FANM-FBA1 FC++.

```
__device__ void
fanmsimu ( /* blockIdx.x, blockIdx.y, dcurandinit,
 dfavatars, dlineups, dpasses, dresults */ )
{
  // GPU
  // A SOCCER SIMULATION
}
__global__ void
setupfanmkernel ( /* dcurandinit */ )
{
  // GPU
  ...
  curand_init ( ... );
}

__global__ void
fanmkernel ( ... )
{
  // GPU
  ...
  fanmsimu ( /* blockIdx.x, blockIdx.y, dcurandinit,
   dfavatars, dlineups, dpasses, dresults */ );
}

int
main ( int argc, char *argv[] )
{
  // CPU
  ...
  dim3 grid ( 5, 10 );
  dim3 tgrid ( 32, 32 );
  ...
  setupfanmkernel <<< grid, tgrid >>> ( /* inic. */ );

  fanmkernel <<< grid, tgrid >>> ( /* drndinic, dfavatars,
   dlineups, dpasses, dresults */ );
  ...
}
```

c) FANM-HPFC: Although it was intended to be a complete FANM implementation, this software works in the analyzer mode only. The development of the standalone and avatar simulation modes was cancelled due to problems inherited from MABSA-HPFC. Another reason for the failure was that, unlike other FANM implementations (e.g., FANM-Debrecen Handsomes) FANM-HPFC does not reuse code from other implementations, it was intended to be written in Java from scratch.

d) FANM-Debrecen Handsomes: This FANM implementation is currently under development by the fifth and ninth authors. It is written in C++ and extends FANM-FBA1 FC++ with the use of a stamina model mentioned in Sect. III-D. Several factors affect the stamina of a player, these factors can be classified into two groups. First, any activity (e.g., movement, passing, dribbling) by the player has a direct impact on the stamina. Second, physiological properties (e.g., blood pressure, blood chemistry characteristics) also modify it.

e) FANM-Debrecen HardAsMuscle: This FANM implementation is developed by the eighth author. It is written in C++ and is based on FANM-FBA1 FC++. There are a few, but significant differences compared to the other FANM implementations. For example, the simulation algorithm uses a foul model that assigns the probability of committing a foul to each player.

G. User Interface

Two main types of users are distinguished in the FootballAvatar system, end users and a special user called FootballAvatar Operator, or FAO for short. Every software product has end users. In our case, they are coaches, managers, and executives of a football club. The FAO is an expert user with a deep knowledge and understading of the FootballAvatar system, whose main task is to provide end users with all information they require, such as simulation results.

Consistent with the above, we distinguish the following two user interface levels: (1) The first UI level is specifically for the FAO. It is a PC-only interface, and its main goal is to provide functionality to run simulations and to provide information for end users. (2) The second UI level is specifically for end users. It is built on the information provided by the FAO. UI implementations must be portable and easy to use, so this interface is only available on portable devices (i.e., on tablets and smartphones).

The organization of the user interface reflects the SimOA introduced in Sect. III-A, i.e., the FAO must choose one of the simulation levels before use.

2D and 3D display applications, mobile solutions, and data analysis applications are typical elements of the user interface. Similarly to MABSA and FANM implementations, they are also being developed in competition with each other. We have five 3D, four 2D, and four mobile-based competing display applications, and eight competing data analysis applications. Only some of them will be presented here.

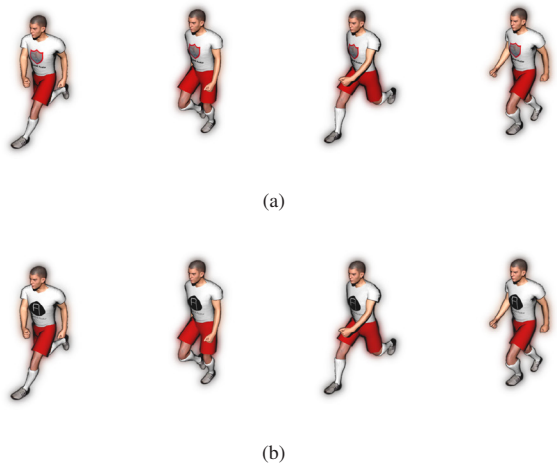
f) Competing 3D Display Applications: Basically, two kinds of 3D display applications are being developed to be used in the FootballAvatar system: an anthropomorphic that uses 3D animated human models, and a schematic that uses buttons to represent players. The latter solution is also referred to as "button soccer display". For example, *3D Model Animation (3DMA)* is an anthropomorphic display application developed by the eleventh author. Fig. 7 shows a 3D model of this application, that was created by Blender [45] and MakeHuman [46].

We have experimented with several "button soccer" type displays. For example, an OpenGL-based display application made by the development team FBA1 is shown in Fig. 8.

g) Competing 2D Display Applications: We have four different competing solutions for 2D display and match analysis. For example, *Multi-Display Player*, or MDP for short, was designed to display multiple visualizations at the same time using screen splitting. It is intended to be used in tactic rooms equipped with a big screen or a projector. MDP provides a flexible layout architecture to which additional visualizations (e.g., new charts) can be added easily.

As another example, Fig. 10 shows a Qt-based display and analytic application that works with both the MABSA and the FANM architecture too and was very successful in the developer competition phase.

h) Competing Mobile Solutions: Fig. 11 shows a screenshot from our four competing mobile solutions. For example, *View 2D Entity (V2DE)* is one of the MDP modules designed to display a football pitch in 2D and to perform basic analytics.



(a)

(b)

Fig. 7: 3D models of a football player used by the *3D Model Animation (3DMA)* display program. The two models differ only in the shirt logo.

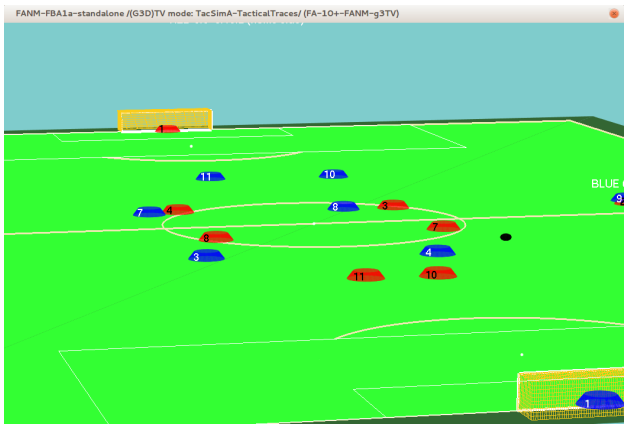


Fig. 8: The OpenGL-based display program of the development team FBA1.

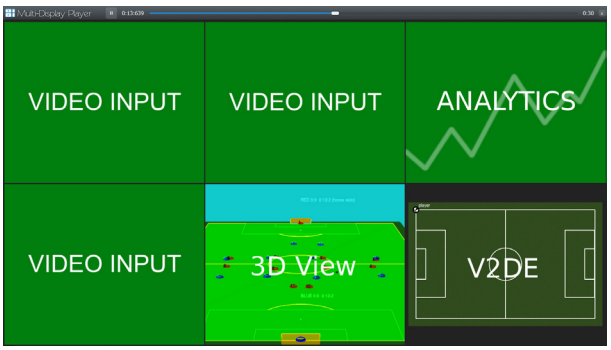


Fig. 9: The modular layout of Multi-Display Player (MDP).

V2DE can run detached from MDP as an independent cross-platform application on Windows, Linux, and also on Android tablets.

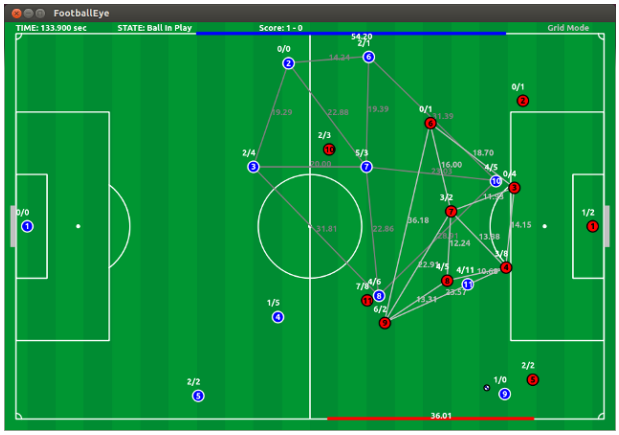


Fig. 10: FootballEye, a QT-based display program of the development team Tunneled Footballers.

(a) The 2D player called FA-droid.

(b) The 2D player called View2DEntity.

(c) Tactic Design Table.

(d) FootballEye with QT5 running on Ubuntu Touch.

Fig. 11: Our four tablet-based solution proposals in the developer competition stage.

H. Project Logo

In accordance with our methodological approach, CP, two competing development teams were formed for the creation of the FootballAvatar logo. We call the resulting logos the *Minimal FootballAvatar Logo* and the *Shield FootballAvatar Logo*.

i) The Minimal FootballAvatar Logo: This logo was created by a one-member team consisting of the sixth author. The logo depicts a letter "F" and a letter "A" that are merged together. The design goal of the artist was to create a logo that is simple and easy to remember.

j) The Shield FootballAvatar Logo: This logo was created by a two-member team consisting of the seventh and the eleventh authors. The design goal was to create a minimalist logo that looks similar to football club logos. The football player shown in the logo is based on a 3D model by the eleventh author.

(a) The minimal FootballAvatar logo.

(b) The shield FootballAvatar logo.

Fig. 12: The two competing FootballAvatar logos.

I. Football Avatars and Avatar-based Simulations

In this section a heuristic mathematical definition is given for the notion of football avatar. The definition is based on statistical hypothesis testing and goes back to [23]. It should be noted that the authors of this paper are currently working on a separate paper devoted to the concept of football avatar.

Let $x = (x_1, \ldots, x_p)$ be a p-dimensional random variable, where x_i, $i = 1, \ldots, p$, are quantities characterizing soccer matches from particular aspects. There are typically such quantities that depend on chance, but their values are known in reality, such as the number of passes of a given player in a match, or the total number of goals scored by a team or a player in a season. The coordinates of x are referred to as probabilistic properties of the soccer. Realizations of this random vector x (known from real soccer) will be called *a-priori observations*. Let $\mathcal{A}$ be a soccer simulation algorithm based on various player, pitch and/or referee features as avatar features that computes independent *simulated observations* for x.

Definition (A Heuristic Definition of Football Avatar)**.** *The pair $(x, \mathcal{A})$ is referred to as football avatar with respect to a given set of probabilistic properties if the probability distribution of the a-priori observations and the probability distribution of the simulated realizations of x are equal.*

The equality of these two probability distributions can be verified by appropriate methods of hypothesis testing.

Statement. *The TF-FANM-NB1 algorithm is a football avatar with respect to the total number of goals scored in a season.*

Following the example of [23], in order to verify the statement we have used the Wald–Wolfowitz and Mann–Whitney tests to determine whether the two distributions are the same or not. The significance level α for all following tests is chosen as $\alpha = 0.05$. In Table IV we have collected the total number of goals scored during seasons 2004/2005–2012/2013 in the Hungarian National Championship.

Season	4/5	5/6	6/7	7/8	8/9	9/10	10/11	11/12	12/13
Goals	681	707	677	746	710	707	690	648	639

TABLE IV: The total number of goals scored during seasons 2004/2005–2012/2013 in the Hungarian National Championship ($\overline{x} = 689.44$, $s_n^* = 33.02$).

Then, we have simulated nine seasons for this championship using the TF-FANM-NB1 algorithm, for which the total number of goals scored obtained from the simulations are shown in Table V. We refer to this sequence of simulations as a.

	1.	2.	3.	4.	5.	6.	7.	8.	9.	test stat.
a	684	696	735	775	780	709	693	695	705	10/24
b	692	720	703	648	689	712	680	708	697	13/34

TABLE V: The total number of goals scored in our simulations ($\overline{x}_a = 719.11$, $s^*_{n_a} = 36.08$ and $\overline{x}_b = 694.33$, $s^*_{n_b} = 21.28$). The last column shows the values of test statistics in form of Wald–Wolfowitz/Mann–Whitney. The structure of the table is the same as the previous one.

Another sequence of simulations, b was run with different parameters in comparison with a. Finally, it should be noted that all these simulations have been computed on the level "-1" in the standalone mode.

1) Simulation-based Decision Making Support: Based on "avatar simulations" introduced in the previous section, we can answer many questions that may be of interest to the coaching staff, such as the following: (1) Which starting eleven should be chosen? (2) Which line-up should be chosen? (3) What is the likely impact of given tactical orders? (4) What is the most likely match result? Details of these topics will be discussed in further works.

J. Project Documents and the Documentation Process

The FootballAvatar project uses DocBook as its primary documentation format. DocBook [47] is an XML vocabulary for writing technical documentation. It is an open standard that is maintained by the OASIS DocBook Technical Committee. Docbook is popular and widely used in the industry. For example, the following projects use DocBook for their documentation: the Fedora Documentation Project, FreeBSD, GNOME, KDE, PHP, PostgreSQL, the Linux Documentation Project. The main advantages of DocBook are the following: (1) it is a platform and vendor independent plain-text based format, (2) DocBook documents can be transformed into various presentation formats, including HTML and PDF.

An important additional advantage comes from its plain-text nature: DocBook documents can be stored under version control and developers can work concurrently on the same document. Therefore, we keep all documentation in our SCM repository, similar to source code. This allows us to track changes and also the activity of our developers.

We use dblatex [48] to create high quality PDF documentation from DocBook XML sources. It is a free and open source tool distributed under the GNU GPL that uses the LaTeX typesetting system to transform DocBook into PostScript or PDF.

It is mentioned here that weekly team meetings and other project activities are recorded on video as part of our documentation process.

Developer's Guide: We follow the principles of agile documentation [49]. In order to implement these, we maintain only one document simply called "Developer's Guide", which includes the Conceptual Plan, the Software Requirements Specification, and the System and Implementation Plan together. It also documents our competition and open source policy related activities using the competition and the open source submission forms shown in Tables I and II.

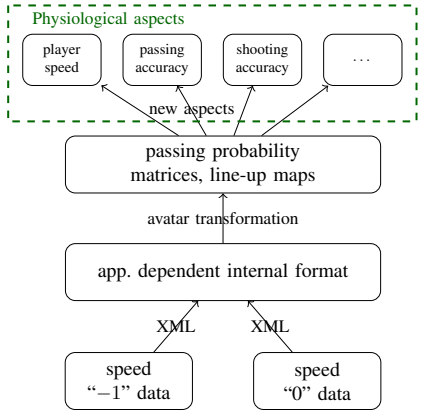


Fig. 13: An avatar is a cross-cutting aspect that can be used to prepare the input files for the simulation algorithms.

IV. THE IT MANIFESTATION OF AVATARS

In Sect. III-I we showed that the notion of "football avatar" is a mathematical definition that can help to evaluate the goodness of soccer simulation algorithms, where the evaluation is based on the comparison of the simulation results and real data. Such a comparison was done in the statement of Sect. III-I where the passing probability matrices and the line-up maps were used. However, it was necessary to include additional properties into the simulation in order to satisfy the TV criterion and statistical requirements. In all cases, the cause of introducing additional properties is that we have applied new approaches to the given simulations. Therefore, an avatar is simply an aspect from an AOP-based [42] viewpoint. To be more precise, avatars are the results of applying avatar transformation aspects to the input of simulation algorithms, as is shown in Fig. 13. In our experiments, the most basic football avatars consist of passing probability matrices and line-up maps. These properties can be extended by introducing new aspects, in which case the appropriate functions of the simulation algorithms must also be overridden to handle new input data by the newly introduced aspects. A few examples for such additional aspects are the following: (1) Physiological aspects: incorporate additional physiological properties into the simulation that affect the performance of players, eg., a stamina model. (2) Environmental aspects: incorporate environmental properties into the simulation, such as current weather conditions, or the type of playing surface. (3) Referee aspects: incorporate properties of the referees into the simulation.

V. VALIDATION OF SIMULATIONS

The following questions naturally arise: (1) How can FootballAvatar simulations be validated? (2) What kind of results

can be expected from the FootballAvatar system? (3) Who knows what can be expected from a computer program in the field of soccer? We are looking for answers to these and similar questions. To be able to answer them, we have to take into account that what can be expected at all from a human coach. Thus, we asked for help from football clubs. At our request, they provided us with preliminary tactical scenarios prepared for their matches. We would like to ground our answers to the above questions on the comparison of the scenarios from football clubs and the scenarios that occur in our simulations. This requires further work, that will be presented in a later publication.

The concept of football avatar, in the strict sense of the definition, by itself, implies a validation criterion (i.e., a statistical test). Some examples of the simulations referred in the definition of football avatar were presented in detail in Sect. III-I.

VI. Conclusion

Finding successful soccer simulations algorithms that can reproduce the distributions observed in reality have proved to be a challenging research and development task. We have known this since the start of the project. To help the search for suitable simulation algorithms we have developed our own methodology that we named as Competitive Programming, or CP for short. We hope that CP can be successfully applied in R&D projects, in which sufficient number of developers are available to allocate multiple competing teams to a specific task. This is often the case when the R&D activity happens fully or partially in a university environment. In order to support the adoption of CP we have made our standard project document templates available at http://footballavatar.hu/CP.

On the basis of the results shown in this paper, it is clear that our CP-based efforts have been successful, because we have found simulation algorithms that can fulfill the definition of football avatar. Thus, in the strict sense the research purpose has been achieved.

VII. Ongoing and Future Work

This section briefly summarizes ongoing and future work mentioned in the paper. (1) We are working on a performance measure that is similar to the EA SPORTS Player Performance Index (PPI) but its development is in a very early stage (see Sect. III-C). (2) The stamina and foul models mentioned in Sect. III-D, and also MABSA simulation algorithms mentioned in Sect. III-E are discussed in detail in another paper that is devoted to the concept of football avatar and has been submitted for publication [38]. (3) We are planning a paper that will address questions regarding the validation of FootballAvatar simulations (see Sect. V).

Acknowledgment

The authors would like to thank the members of the research group "World Football—Modeling and Visualization" at the University of Debrecen for the meetings and their useful comments that help them better understand soccer. During the development of the FootballAvatar system authors worked in cooperation with other project partner companies (namely, U1 Research Ltd., IQRS Ltd., and Satrax Ltd.) and they would like to thank them for their contributions that will be important in the operation of the FootballAvatar system at real football clubs. We, the authors would like to express our special thanks to Tamás Sándor, Péter Szakály, Elemér Kondás, Ferenc Frida and Sándor Szilágyi. Last, but not least, thanks to all members (especially Prof. György Terdik, Tibor Balla and Piroska Biró) of the "Nagyerdei Gerundium" working group of SziMe3D Ltd. for their continued help and support.

The publication was supported by the GOP-1.2.1-11-2012-0005 (*SziMe3D – 3D-s technológiai innovációk a turizmus, oktatás és sport területén*, *SziMe3D–3D technological innovation in tourism, education and sport*) project. The project has been supported by the European Union.

References

[1] N. Bátfai, "Footballer and Football Simulation Markup Language and Related Simulation Software Development," Journal of Computer Science and Control Systems, vol. 3, no. 1, pp. 13–18, 2010.

[2] N. Bátfai, R. Dóczi, J. Komzsik, A. Mamenyák, Cs. Székelyhídi, J. Zákány, M. Ispány, and Gy. Terdik, "Applications of a simplified protocol of RoboCup 2D soccer simulation," Infocommunications Journal, vol. 5, no. 1, pp. 15–20, 2013.

[3] N. Bátfai and Gy. Terdik, "The application of the data of RoboCup 2D soccer simulation league to test several sport science results," 2012, (in manuscript). [Online]. Available: http://robocup.inf.unideb.hu/fersml/assr/

[4] K. Furugaki, T. Takagi, A. Sakata, and D. Okayama, "Innovation in software development process by introducing Toyota Production System," FUJITSU Scientific & Technical Journal, vol. 43, no. 1, pp. 139–150, 2007. [Online]. Available: http://www.fujitsu.com/downloads/MAG/vol43-1/paper16.pdf

[5] O. Salo and P. Abrahamsson, "Integrating agile software development and software process improvement: a longitudinal case study," in Proceedings of the 2005 International Symposium on Empirical Software Engineering (ISESE 2005). IEEE, 2005, pp. 193–202.

[6] M. Fowler and J. Highsmith, "The agile manifesto," Software Development Magazine, vol. 9, no. 8, pp. 29–30, 2001.

[7] K. Schwaber, "Scrum development process," in Proceedings of the 10th Annual ACM OOPSLA, 1995, pp. 117–134.

[8] K. Schwaber and M. Beedle, Agile Software Development with Scrum. Prentice Hall, 2001.

[9] K. Beck and C. Andres, Extreme Programming Explained: Embrace Change, 2nd ed. Addison-Wesley, 2004.

[10] B. Fitzgerald, G. Hartnett, and C. K., "Customising agile methods to software practices at Intel Shannon," European Journal of Information Systems, vol. 15, no. 2, pp. 200–213, 2006.

[11] P. Bell. (2007, Jun. 17) Solo Scrums. blog post. Accessed: 2015-12-04. [Online]. Available: http://www.pbell.com/index.cfm/2007/6/17/Solo-Scrums

[12] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, and D. Dixon, "Gamification: Using game design elements in non-gaming contexts," in CHI '11 Extended Abstracts on Human Factors in Computing Systems, ser. CHI EA '11. ACM, 2011, pp. 2425–2428.

[13] O. Schenk, M. Christen, and H. Burkhart, "Algorithmic performance studies on graphics processing units," Journal of Parallel and Distributed Computing, vol. 68, no. 10, pp. 1360–1369, 2008.

[14] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using CUDA," Journal of Parallel and Distributed Computing, vol. 68, no. 10, pp. 1370–1380, 2008.

[15] NVIDIA Corporation. NVIDIA CUDA Toolkit. Accessed: 2015-12-04. [Online]. Available: https://developer.nvidia.com/cuda-toolkit

[16] J. Pendlebury, H. Xiong, and R. Walshe, "Artificial neural network simulation on CUDA," in Proceedings of the 2012 IEEE/ACM 16th International Symposium on Distributed Simulation and Real Time Applications, ser. DS-RT '12. IEEE, 2012, pp. 228–233.
[17] H. Akiyama, H. Shimora, T. Nakashima, Y. Narimoto, and T. Okayama, "HELIOS2011 team description," 2011.
[18] A. Bai, G. Lu, H. Zhang, and X. Chen, "WrightEagle 2D Soccer Simulation Team Description 2011," 2011. [Online]. Available: http://ai.ustc.edu.cn/en/robocup/2D/tdps/WrightEagle2011_2D_Soccer_Simulation_Team_Description_Paper.pdf
[19] N. Bátfai, "Quantum consciousness soccer simulator," CoRR, vol. abs/1211.2719, 2012. [Online]. Available: http://arxiv.org/abs/1211.2719
[20] R. Koning, M. Koolhaas, G. Renes, and G. Ridder, "A simulation model for football championships," European Journal of Operational Research, vol. 148, no. 2, pp. 268–276, 2003.
[21] R. Koning, "Balance in competition in Dutch soccer," Journal of the Royal Statistical Society: Series D (The Statistician), vol. 49, no. 3, pp. 419–431, 2000.
[22] A. C. Constantinou, N. E. Fenton, and M. Neil, "pi-football: A Bayesian network model for forecasting association football match outcomes," Knowledge-Based Systems, vol. 36, pp. 322–339, 2012.
[23] N. Bátfai, "The socceral force," CoRR, vol. abs/1004.2003, 2010. [Online]. Available: http://arxiv.org/abs/1004.2003
[24] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "RoboCup: The robot world cup initiative," in Proceedings of the first international conference on Autonomous agents, ser. AGENTS'97. ACM, 1997, pp. 340–347.
[25] D. Wells. XP flow chart. Accessed: 2015-12-04. [Online]. Available: http://www.extremeprogramming.org/map/project.html
[26] J. Martin, Rapid Application Development. Macmillan Publishing Co., 1991.
[27] S. Halim and F. Halim, Competitive Programming 3: The New Lower Bound of Programming Contests, 3rd ed. Lulu, 2013. [Online]. Available: https://sites.google.com/site/stevenhalim/
[28] Wikipedia, "Competitive programming — Wikipedia, the free encyclopedia," 2013. [Online]. Available: en.wikipedia.org/wiki/Competitive_programming
[29] M. Bouca, "Mobile communication, gamification and ludification," in Proceedings of the 16th International Academic MindTrek Conference, ser. MindTrek '12. ACM, 2012, pp. 295–301.
[30] L. F. Wurster, "User survey analysis: Open-source software, worldwide," Gartner, Inc., Tech. Rep., 2008. [Online]. Available: http://www.gartner.com/id=757916
[31] A. Raina and L. F. Wurster, "Market trends: Application development software, worldwide, 2012–2016," Gartner, Inc., Tech. Rep., 2012. [Online]. Available: http://www.gartner.com/id=2098416
[32] R. T. Watson, D. Wynn, and M.-C. Boudreau, "JBOSS: The evolution of professional open source software," MIS Quarterly Executive, vol. 4, no. 3, pp. 329–341, Sep. 2005.
[33] B. Fitzgerald, "The transformation of open source software," MIS Quarterly, vol. 30, no. 3, pp. 587–598, Sep. 2006.
[34] R. T. Watson, M.-C. Boudreau, P. T. York, M. E. Greiner, and D. Wynn, Jr., "The business of open source," Communications of the ACM, vol. 51, no. 4, pp. 41–46, Apr. 2008.
[35] M. Ruffin and C. Ebert, "Using open source software in product development: A primer," IEEE Software, vol. 21, no. 1, pp. 82–86, Jan. 2004.
[36] H. J. Meeker, The Open Source Alternative: Understanding Risks and Leveraging Opportunities. John Wiley & Sons, 2008.
[37] Free Software Foundation. (2013) Frequently asked questions about the GNU licenses. Accessed: 2015-12-04. [Online]. Available: http://www.gnu.org/licenses/gpl-faq.html
[38] N. Bátfai, A. Mamenyák, P. Jeszenszky, G. Kövér, M. Smajda, R. Besenczi, B. Halász, Gy. Terdik, and M. Ispány, "Sport Science Soccer Simulations," 2014, submitted manuscript.
[39] EA SPORTS Player Performance Index. Football Association Premier League Limited. Accessed: 2015-12-04. [Online]. Available: http://www.premierleague.com/en-gb/players/ea-sports-player-performance-index/
[40] I. G. McHale, P. A. Scarf, and D. E. Folker, "On the development of a soccer player performance rating system for the English Premier League," Interfaces, vol. 42, no. 4, pp. 339–351, 2012.
[41] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi, "The similarity metric," IEEE Transactions on Information Theory, vol. 50, no. 12, pp. 3250–3264, 2004.
[42] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin, "Aspect-oriented programming," in ECOOP'97 – Object-Oriented Programming, ser. Lecture Notes in Computer Science, vol. 1241. Springer-Verlag, 1997, pp. 220–242.
[43] Google Inc. Protocol Buffers. Accessed: 2015-12-04. [Online]. Available: https://developers.google.com/protocol-buffers/
[44] A. A. Janes and G. Succi, "The dark side of agile software development," in Proceedings of the ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software. ACM, 2012, pp. 215–228.
[45] Blender Foundation. Blender. Accessed: 2015-12-04. [Online]. Available: http://blender.org/
[46] The MakeHuman team. MakeHuman. Accessed: 2015-12-04. [Online]. Available: http://makehuman.org/
[47] N. Walsh and L. Muellner, DocBook: The Definitive Guide. O'Reilly Media, 1999. [Online]. Available: http://www.docbook.org/tdg/en/html/docbook.html
[48] B. Guillon. dblatex: DocBook to LaTeX publishing. Accessed: 2015-12-04. [Online]. Available: http://dblatex.sourceforge.net/
[49] A. Rüping, Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects. John Wiley & Sons, Inc., 2003.

Norbert Bátfai is working as an assistant professor in Faculty of Informatics at the University of Debrecen, Hungary. He received his M.Sc. (summa cum laude) in Computer Science in 1998 from the Kossuth Lajos University (KLTE), Debrecen, Hungary. In 1999, he won the first prize in the Java Programming Contest organized by Hungarian Java Alliance: Sun, IBM, Oracle, Novell and IQSoft. In 2004, his company won the first prize in the Hungarian Mobile Java Developer Contest organized by Sun Hungary and Nokia Hungary. In 2008, the Hungarian Chief Information Officers' Association awarded him the IT trainer of the year title. He received his Ph.D. degree in 2011. He won the Pollák–Virág award from the Scientific Association for Infocommunications, Hungary in 2012.

Péter Jeszenszky received his Ph.D. degree in 2012 from the University of Debrecen, Hungary. Currently, he is an assistant professor at the Department Technology at the University of Debrecen. He won the the Pollák–Virág award from the Scientific Association for Infocommunications, Hungary in 2012.

András Mamenyák is majoring in Engineering Information Technology BSc at the University of Debrecen, Hungary. In 2013, he won the first prize in the XDA Tablet Z Development Competition organized by XDA Developers.

Béla Halász is working as the project manager for the FootballAvatar project at SziMe3D Ltd. He has extensive working experience in managing various development projects in different IT fields, from Health Informatics to Software Licence Protection.

Renátó Besenczi is majoring in Engineering Information Technology BSc and working as a research team member at the University of Debrecen, Hungary.

Balázs Kóti is majoring in Software Information Technology BSc at the University of Debrecen, Hungary. He started his studies at the University in 2012. He recieved technician degree from geodesy and geographical information systems at Vásárhelyi Pál Engineering High School and Technicum.

Gergely Kövér is majoring in Software Information Technology BSc at the University of Debrecen, Hungary. He started his studies at the University in 2012.

Máté Smajda is majoring in Software Information Technology BSc at the University of Debrecen, Hungary. He started his studies at the University in 2012, and he is also an experienced football player, winner of the Hungarian National Football Olympiad for Students.

Csaba Székelyhídi is majoring in Software Information Technology BSc at the University of Debrecen. He is a member of World Football Modelling and Visualizing Research Group and FootballAvatar Project. He started his studies at the University in 2011.

Tamás Takács is majoring in Software Information Technology BSc at the University of Debrecen, Hungary. He started his studies at the University in 2012.

János Komzsik is majoring in Engineering Information Technology BSc at the University of Debrecen, Hungary.

Géza Róka is the club director of DVSC Futball Szervező Zrt., the football club of Debrecen, which has won the Hungarian Championship and the Hungarian National Cup six times, the Hungarian Super Cup five times, and has participated in the UEFA Champions League and Europa League group phases. His main area of expertise is sports law, particularly international football law. He is currently working on his Ph.D. thesis, which examines the impact of the EU legal system to the international regulation of football.

Márton Ispány received his Ph.D. degree in 1997 from the Kossuth Lajos University, Hungary. He is an associate professor at the Department of Information Technology at the University of Debrecen, Hungary. Dr. Ispány's research areas include integer valued time series analysis, branching processes, and data mining. He won the Alexits György award from the Hungarian Academy of Sciences and the Pollák–Virág award from the Scientific Association for Infocommunications, Hungary in 2012

IEEE Wireless Communications and Networking Conference
19-22 March 2017
San Francisco, CA, USA

IEEE WCNC is the premier event for wireless communications researchers, industry professionals, and academics interested in the latest development and design of wireless systems and networks. Sponsored by the IEEE Communications Society, IEEE WCNC has a long history of bringing together industry, academia, and regulatory bodies. In 2017, the city of San Francisco and the Silicon Valley will become the center of the wireless world by hosting IEEE WCNC'17. The conference will include technical sessions, tutorials, workshops, and technology/business panels. You are invited to submit papers in all areas of wireless communications and networks. Potential topics include, but not limited to:

Track 1: PHY and Fundamentals
- Channel modeling, characterization and estimation
- Modulation, coding, diversity, equalization, synchronization
- OFDM, multi-carrier modulation, waveform design
- Interference modeling, management, cancellation and alignment
- PHY strategies for low-rate, sporadic and asynchronous communications
- MIMO, massive MIMO and cloud-RAN
- Cooperative, device-to-device and multi-hop communication
- Cognitive radio, spectrum sensing
- Content caching and storage in wireless networks
- PHY layer design for cellular, wireless LAN, ad hoc and sensor networks
- Energy efficient and energy harvesting PHY layer design
- Joint information and energy transmission
- PHY layer security and privacy, ultra-wideband, mmWave and sub-THz communication
- Information-theoretic aspects of wireless communications
- Signal processing for wireless communications
- Molecular and nano communications

Track 2: MAC and Cross-Layer Design
- Wireless MAC protocols for 5G: design, analysis, and optimization
- Cognitive and cooperative MAC
- MAC for mesh, ad hoc, relay and sensor networks
- Scheduling and radio resource management
- Cross-layer MAC design
- Software defined radio, RFID MAC
- QoS support and energy efficient MAC
- MAC protocol for energy harvesting wireless networks
- MAC design for multitier cellular/small cell networks
- Multiple access in machine-to-machine communication
- MAC for cloud-RAN
- MAC protocols for molecular and nano networks
- MAC protocols for mmWave networks
- Full-duplex MAC design
- Cross-layer design for massive MIMO and multiuser MIMO networks

Track 3: Wireless Networks · Network Estimation and Processing Techniques
- Mesh, relay, sensor and ad hoc networks
- Mobility, location, and handoff management
- Wireless routing
- Multimedia QoS and traffic management
- Wireless broadcast, multicast and streaming
- Congestion and admission control
- Proxies and middleware for wireless networks
- Wireless network security and privacy
- Software-defined wireless networks
- Cognitive radio networks
- Mobile social networks
- Mobile cloud and fog networking
- Mobile big data and network data analytics

Track 4: Emerging Technologies, Architectures and Services
- Adaptive content distribution in on-demand services
- Context and location-aware wireless services and applications
- User-centric networks and adaptive services
- Wireless body area networks and e-health services
- Intelligent transportation systems
- Dynamic sensor networks for urban applications
- Wireless emergency and security systems
- Ultra-reliable communication
- Enabling regulations, standards, spectrum management
- Hybrid licensed/unlicensed spectrum access schemes (e. g. licensed-assisted access)
- Technologies, architectures and enabling business models for rural communications
- Satellite-based mobile access and backhaul
- Full duplexing · Joint access and backhaul schemes
- Testbed and prototype implementation of wireless services

Accepted and presented papers will be published in the IEEE WCNC 2017 Conference Proceedings and submitted to IEEE Xplore®.
Full details of submission procedures and requirements for authors of accepted papers are available at http://wcnc2017.ieee-wcnc.org.

IMPORTANT DATES

Paper Submission Deadline:	30 September 2016
Notification of Acceptance:	15 December 2016
Camera-Ready Submission:	12 January 2017
Tutorial Proposals:	30 September 2016
Workshop Proposals:	Separate Call-for-Workshops
Panel Proposals:	30 September 2016

COMMITTEE CHAIRS

General Chairs
Andrea Goldsmith, Stanford University
Katie Wilson, Santa Clara University

Steering Committee Chair
Khaled Letaief, HKUST

Technical Program Chairs
Shungang Cui, Texas A&M University
Elza Erkup, New York University
Angel Lozano, Universitat Pompeu Fabra

Participation should be submitted to the suitable conference track on EDAS.

For more information about IEEE WCNC 2017, please visit http://wcnc2017.ieee-wcnc.org.

Guidelines for our Authors

Format of the manuscripts

Original manuscripts and final versions of papers should be submitted in IEEE format according to the formatting instructions available on
http://www.ieee.org/publications_standards/publications/authors/authors_journals.html#sect2,
"Template and Instructions on How to Create Your Paper".

Length of the manuscripts

The length of papers in the aforementioned format should be 6-8 journal pages.
Wherever appropriate, include 1-2 figures or tables per journal page.

Paper structure

Papers should follow the standard structure, consisting of *Introduction* (the part of paper numbered by "1"), and *Conclusion* (the last numbered part) and several *Sections* in between.
The Introduction should introduce the topic, tell why the subject of the paper is important, summarize the state of the art with references to existing works and underline the main innovative results of the paper. The Introduction should conclude with outlining the structure of the paper.

Accompanying parts

Papers should be accompanied by an *Abstract* and a few *index terms (Keywords)*. For the final version of accepted papers, please send the *short cvs* and *photos* of the authors as well.

Authors

In the title of the paper, authors are listed in the order given in the submitted manuscript. Their full affiliations and e-mail addresses will be given in a footnote on the first page as shown in the template. No degrees or other titles of the authors are given. Memberships of IEEE, HTE and other professional societies will be indicated so please supply this information. When submitting the manuscript, one of the authors should be indicated as corresponding author providing his/her postal address, fax number and telephone number for eventual correspondence and communication with the Editorial Board.

References

References should be listed at the end of the paper in the IEEE format, see below:

a) Last name of author or authors and first name or initials, or name of organization
b) Title of article in quotation marks
c) Title of periodical in full and set in italics
d) Volume, number, and, if available, part
e) First and last pages of article
f) Date of issue

[11] Boggs, S.A. and Fujimoto, N., "Techniques and instrumentation for measurement of transients in gas-insulated switchgear," IEEE Transactions on Electrical Installation, vol. ET-19, no. 2, pp.87–92, April 1984.

Format of a book reference:

[26] Peck, R.B., Hanson, W.E., and Thornburn, T.H., Foundation Engineering, 2nd ed. New York: McGraw-Hill, 1972, pp.230–292.

All references should be referred by the corresponding numbers in the text.

Figures

Figures should be black-and-white, clear, and drawn by the authors. Do not use figures or pictures downloaded from the Internet. Figures and pictures should be submitted also as separate files. Captions are obligatory. Within the text, references should be made by figure numbers, e.g. "see Fig. 2."
When using figures from other printed materials, exact references and note on copyright should be included. Obtaining the copyright is the responsibility of authors.

Contact address

Authors are requested to send their manuscripts via electronic mail or on an electronic medium such as a CD by mail to the Editor-in-Chief:

Rolland Vida
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
2 Magyar Tudósok krt.
Budapest, 1117 Hungary
vida@tmit.bme.hu

SCIENTIFIC ASSOCIATION FOR INFOCOMMUNICATIONS

Who we are

Founded in 1949, the Scientific Association for Infocommunications (formerly known as Scientific Society for Telecommunications) is a voluntary and autonomous professional society of engineers and economists, researchers and businessmen, managers and educational, regulatory and other professionals working in the fields of telecommunications, broadcasting, electronics, information and media technologies in Hungary.

Besides its 1000 individual members, the Scientific Association for Infocommunications (in Hungarian: Híradástechnikai Tudományos Egyesület, HTE) has more than 60 corporate members as well. Among them there are large companies and small-and-medium enterprises with industrial, trade, service-providing, research and development activities, as well as educational institutions and research centers.

HTE is a Sister Society of the Institute of Electrical and Electronics Engineers, Inc. (IEEE) and the IEEE Communications Society.

What we do

HTE has a broad range of activities that aim to promote the convergence of information and communication technologies and the deployment of synergic applications and services, to broaden the knowledge and skills of our members, to facilitate the exchange of ideas and experiences, as well as to integrate and harmonize the professional opinions and standpoints derived from various group interests and market dynamics.

To achieve these goals, we…

- contribute to the analysis of technical, economic, and social questions related to our field of competence, and forward the synthesized opinion of our experts to scientific, legislative, industrial and educational organizations and institutions;
- follow the national and international trends and results related to our field of competence, foster the professional and business relations between foreign and Hungarian companies and institutes;
- organize an extensive range of lectures, seminars, debates, conferences, exhibitions, company presentations, and club events in order to transfer and deploy scientific, technical and economic knowledge and skills;
- promote professional secondary and higher education and take active part in the development of professional education, teaching and training;
- establish and maintain relations with other domestic and foreign fellow associations, IEEE sister societies;
- award prizes for outstanding scientific, educational, managerial, commercial and/or societal activities and achievements in the fields of infocommunication.

Contact information

President: **GÁBOR MAGYAR, PhD** • *elnok@hte.hu*

Secretary-General: **ISTVÁN BARTOLITS** • *bartolits@nmhh.hu*

Operations Director: **PÉTER NAGY** • *nagy.peter@hte.hu*

International Affairs: **ROLLAND VIDA, PhD** • *vida@tmit.bme.hu*

Address: H-1051 Budapest, Bajcsy-Zsilinszky str. 12, HUNGARY, Room: 502

Phone: +36 1 353 1027, Fax: +36 1 353 0451

E-mail: *info@hte.hu*, Web: *www.hte.hu*