Pattern Handling for Quantifying Hardware Components of Signature-Based Cybersecurity Systems

Serhii Ya. Hilgurt[a]

[a] *Pukhov Institute for Modelling in Energy Engineering of NAS of Ukraine, 15, General Naumov str., Kyiv, 03164, Ukraine*

Abstract

Cybersecurity Systems such as network intrusion detection systems (NIDS) successfully use artificial intelligence technologies and other anomaly-based approaches to detect malicious activity. But only the signature approach allows you to completely eliminate recognition errors. This is especially relevant for critical infrastructure facilities. However, a significant drawback of signature-based tools is the high computational complexity: the NIDS has in real time to look through an intensive flow of incoming data against a lot of signatures –registered signs of known attacks. Therefore, the developers of such systems turn to hardware implementation, primarily on a reconfigurable platform, that is, using field programmable gate arrays (FPGA). The property of FPGAs to quickly reprogram their internal structure gives reconfigurable cybersecurity systems unprecedented flexibility, allowing them to adapt to external conditions. There are many different approaches to construct schemes to recognize patterns (which are parts of the signatures), as well their variations, modifications and improvements. Choosing the optimal technical solution for recognizing a specific set of patterns is not a trivial task. Besides having qualitative information about the variety of known schemes, it is strongly needed to be able to quickly quantify reconfigurable components to build on them optimized signature-based cybersecurity systems. This requires in turn to effectively handling the patterns that are important parts of FPGA-based schemes of NIDS' recognition components. The paper proposes a pattern set formalization description and a technique to order patterns that simplifies calculating quantitative characteristics of recognition schemes without performing expensive procedure of digital circuit synthesis using CAD tool.

Keywords[1]

NIDS, signature analysis, multi-pattern matching, FPGA, quick quantifying, pattern sorting

1. Introduction

The signature approach, due to the precise execution of the recognizing malicious activity, is still a relevant technology used in such protection tools as network intrusion detection systems (NIDS) and other cybersecurity tools. Due to the increase in the size of signature databases and the high bandwidth of modern networks, traditional software solutions can no longer meet the demands placed on the performance of such systems. Hardware approaches are increasingly used to speed up the resource-intensive procedure of multi-pattern matching [1], which is carried out in signature means of information protection. Reconfigurable accelerators (RA) based on FPGA are usually used as a platform for the implementation of such solutions due to their almost software flexibility and near-ASIC performance [2], [3].

ITTAP'2022: 2nd International Workshop on Information Technologies: Theoretical and Applied Problems, November 22–24, 2022, Ternopil, Ukraine
EMAIL: hilgurt@ukr.net (S.Ya. Hilgurt)
ORCID: 0000-0003-1647-1790 (S.Ya. Hilgurt)

A comprehensive analysis of numerous studies from around the world was performed by author in [4]. Figure 1 schematically presents the relationships between the main approaches to the construction of recognition modules of reconfigurable cyber defense systems, the technologies on which they are based, and techniques for improving the performance indicators of the created developments. Here, CAM is content-addressable memory, TCAM is a ternary associative memory, DFA is a deterministic finite state automaton, NFA is a nondeterministic finite state automaton, H+R+Cmp is a cascade recognition scheme based on a hash function, a non-volatile memory device, and digital comparators. Actually, there are many more components of the hierarchy of approaches, technologies and techniques in this area of research, as well as the connections between them. Most of them are not shown in the figure for the sake of clarity.

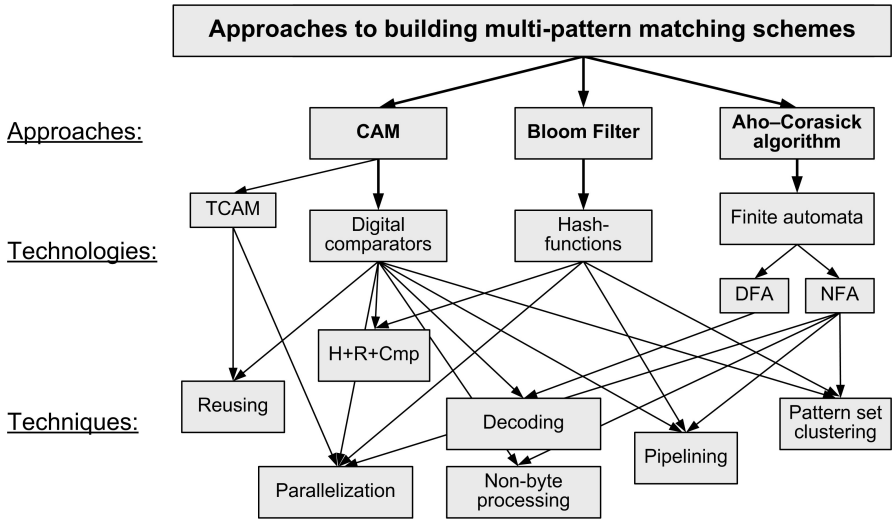

Figure 1: Approaches, technologies and techniques to construct reconfigurable recognition modules

Here CAM – Content-addressable memory; TCAM – ternary CAM; DFA / NFA – deterministic / nondeterministic finite automaton; H+R+Cmp – cascaded matching scheme based on hash-functions, read-only memory and digital comparators [5]. Actually, there are much more elements in this hierarchy of approaches, technologies and techniques, as well as the links between them. For clarity, most of them are not shown in the figure.

Each of approaches has advantages and disadvantages. Researchers offer modifications to the basic circuits, use a variety of methods, tricks and techniques in an attempt to increase their efficiency and overcome the shortcomings of each approach.

There is a need for a tool to quantify the technical characteristics of each solution in order to work with them productively. In [6] a hardware costs estimation technique was proposed to speed up the calculation of the quantitative characteristics of signature-based cybersecurity systems. Several examples of building so called estimation functions (EF) are given for the basic recognition schemes. Nevertheless as he analysis shows, that the basic schemes actually aren't used in practical applications. On the other hand the creating of EFs for modifications of basic schemes are more effective require ability to effectively operate with the patterns, because them play an important role in FPGA-based schemes for NIDS.

In this paper, a technique for dealing with patterns of signature databases is proposed, which allows simplifying the procedures of estimation and comparison of various technical solutions of reconfigurable signature-based cybersecurity systems.

The rest of this paper is organized as follows. Section 2 briefly considers the approach to the building based on content-addressable memory and digital comparators. Section 3 discusses the ordering technique principles as a base for the formalization of pattern set properties. Section 4 introduces quantitative descriptions of hardware recognition schemes for making quick estimations of resource costs. Experiments to test the proposed technique are given in the 5th section. Section 6 concludes the research.

2. Approach to the building of reconfiguring matching modules based on content-addressable memory and digital comparators

The survey on hardware solutions for signature-based security systems [4] shows that when building NIDS and other signature-based cybersecurity systems, the following three approaches using the corresponding technical solutions based on appropriate circuit technologies show the best results:

- Content-addressable memory on digital comparators [7], [8], [9];
- Bloom filter scheme that uses hash functions [10], [11], [12];
- Aho–Corasick finite automata [13], [14], [15], [16].

Let's remind some moments about the approach based on content-addressable memory and digital comparators analyzed in [4] to better understand the pattern dealing technique when building EFs for some matching schemes. Namely the most effective modifications of the basic CAM scheme in the sense of saving hardware will be considered.

Content-addressable memory devices have been specifically designed to meet the needs of fast matching. They operate in the opposite manner to traditional random access memory (RAM) devices. If the RAM stores the given data at a certain address and finds it at this address, then the CAM, on the contrary, looks for the location (i.e. address, or cell number) of the data in the storage device by their content or indicates absence. Digital comparators (DCs) are the fast-acting basis of CAMs; therefore, further these two names are sometimes used as synonyms.

The easiest way to detect the presence of a suspicious pattern in a data stream is a set of DCs that compare the bytes of the input sequence with all the characters of this pattern. The Figure 2 a shows an example that is able to recognize "ABC" pattern. It contains input conveyor composed of 8-bit registers RG_i, three digital comparators CMP_i, – one for every character, and the "AND" gate. In common case the number of elements depends from pattern set parameters. This scheme has been called the Basic Content-Addressable Memory Scheme, or ***BsCAM Scheme***.

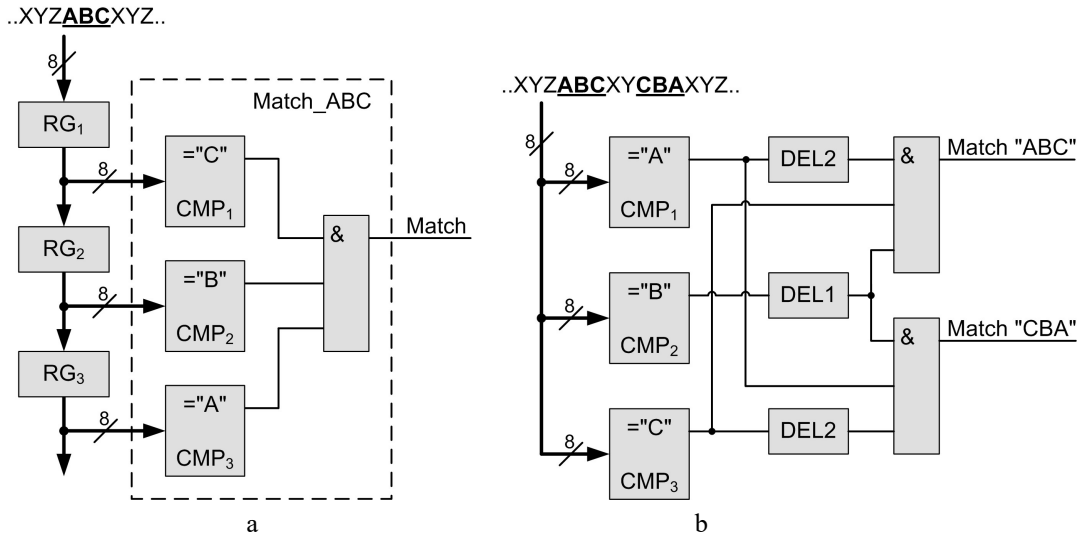

Figure 2: Recognition by digital comparators: direct (a); DCAM solution (b)

BsCAM scheme provides very high performance. In fact, it is able to output the result in one clock cycle after the input data is fed. Its structure is regular, that is also an advantage. The main drawback of the scheme is the huge hardware costs. To create a BsCAM scheme, you need as many DCs as there are characters in the pattern dictionary. In addition, multi-input AND gates are needed to join the output signals from comparators.

To reduce the hardware costs of the CAM-based recognizing scheme the DCs can be reused. In the extreme case only one DC is allocated for each character, and the resulting match signal is obtained by joining digital delay circuits with help of AND gates, as shown in example in Figure 2 b. In this figure, the delay circuits DEL1 and DEL2 provide one cycle and two cycle's delays respectively.

Such a solution is called the Recognition Scheme Based on Decoded CAM or ***DCAM Scheme***, because the full set of DCs in fact composes a decoder [4], [17].

Further research has made it possible to achieve a greater reduction in hardware costs with a partial recognition scheme. Long patterns can be split into shorter pieces and matched sequentially. It is enough to delay only the partial match signal instead of using many circuits of long delay for large patterns. Figure 3 shows a 31-character pattern recognition scheme built according to this idea. This solution has been called the Partially Decoded CAM Recognition Scheme or ***DpCAM Scheme*** [4], [9].

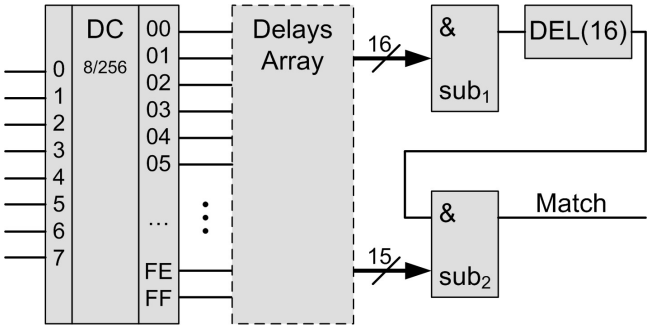

Figure 3: DpCAM technical solution

Experience shows that DCAM and DpCAM schemes can reduce the consumption of FPGA hardware resources by about half compared to the basic scheme.

As we can see in the Figures 2 and 3, information that the pattern set contains actually "wired" in the structure of the recognition circuit. Therefore the parameters of the patterns significantly affect the properties of the hardware matching scheme. That is why it is very important to formalize these parameters in order to operate with them. And first of all it is necessary a technique to sort patterns in a set.

As we see in Figures 2 and 3, the information contained in the set of patterns is actually "hardwired" into the structure of the recognition scheme. Therefore, the peculiarities of pattern set significantly affect the properties of the hardware matching scheme. That is why it is very important to formalize these peculiarities in order to effectively operate with them, and first of all, a technique for sorting patterns in a set is needed.

3. Pattern ordering technique

Let's represent the set of patterns that the cybersecurity system matching module should recognize as an array of its members and a list of its parameters:

$$P = \left\{ p_1,\ p_2,\ p_3, \ldots\ p_k, \ldots\ p_\sigma \middle|\ \sigma, \Omega, m_{\min}, m_{\max}, \delta,\ \mu,\ \mu_z, \nu \right\},$$

where $p_1, p_2, p_3, \ldots p_k, \ldots p_\sigma$ – the set of patterns itself, σ – the number of patterns in the set, Ω – the total number of symbols in the set of patterns, $m_{\min}$ – the length of the shortest pattern in the set, $m_{\max}$ – the length of the longest pattern in the set, δ – the length distribution function, μ – the first self-similarity function, μ_z – the first partial self-similarity function, ν – the second self-similarity function. Patterns are fixed sequences of symbols, each code of which belongs to a certain alphabet Σ. In the case of encoding by bytes $\Sigma = \left\{ 00_{16}, 01_{16}, 02_{16}, ..., \mathrm{FF}_{16} \right\}$.

Self-similarity functions μ, μ_z and ν are quantitative parameters of the pattern set that characterize the degree of similarity of patterns among themselves, determining the redundancy present in a set of patterns, which can be used to reduce the number of comparison operations when performing multi-pattern matching. These functions play an important role for quantifying the recognizing schemes, especially – for effective modifications of basic matching schemes grounded on CAMs and DCs. To provide quick quantification procedures, using self-similarity functions and

length distribution function as well, it is necessary to formalize pattern sets of the NIDS' signature database in order to have possibility to operate them in mathematically strict manner. Below, a technique for patterns handling is discussed, which is based on a proposed certain sorting order.

Let's sort all the patterns in the set P by increasing length, as shown in Figure 4. Columns made of squares here represent strings of characters.

Let's call "a package" a subset of patterns of the same length, ignoring how the patterns within this subset are ordered. Let's enter the index j such that it coincides with the length of the patterns in the package $j = m_{\min}, m_{\min+1}, m_{\min+2}, \ldots m_j, \ldots m_{\max}$, and each subsequent value of this index must be necessarily one more than the previous one, that is, there are no gaps in the numbering. Then the length of each pattern in the set will be equal to the index of its package: $m_j = j$.

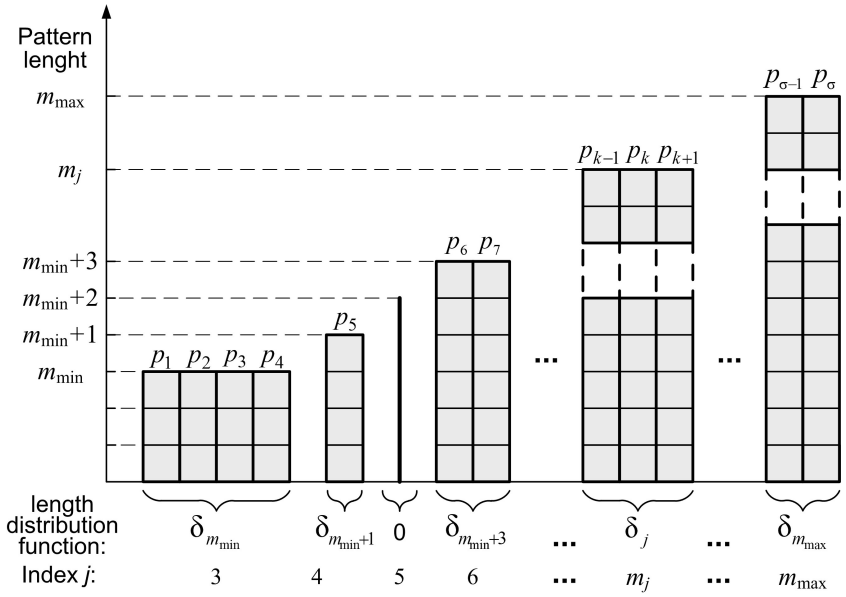

Figure 4 The principle of pattern ordering by packages

The reverse statement is not true, since for some indices j there may not be patterns of correspondent length. Therefore, the total number of packets in the set of patterns P is generally less than the difference in the lengths of the extreme dimensions ($m_{\max} - m_{\min}$).

Let's define the length distribution function δ as a function of the index j equal to the number of patterns in the corresponding package: in the example in Figure 2 j = 3, 4, 5, … $m_{\max}$, $\delta(3) = 4$, $\delta(4) = 1$, $\delta(5) = 0$, $\delta(6) = 2$, $\delta(m_{\max}) = 2$.

Using the length distribution function, it is convenient to calculate the number of non-zero packets and the total number of characters Ω in the set. The total number of characters is equal to the sum of characters in each packet, which in turn is equal to the product of the length of the lines by their number in the packet:

$$\Omega = \sum_{j=m_{\min}}^{m_{\max}} \delta_j m_j = \sum_{j=m_{\min}}^{m_{\max}} \delta_j j .$$

To count the number of packets ξ in the set P, we need to define the zero inequality function:

$$\mathrm{NotZ}(x) = \begin{cases} 0, & x = 0 \\ 1, & x > 0 \end{cases}.$$

Then the number of packets in the set P:

$$\xi = \sum_{j=m_{\min}}^{m_{\max}} \mathrm{NotZ}(\delta_j).$$

To estimate quantitative parameters of technical solutions for signature cybersecurity systems, it is also necessary to determine the self-similarity functions of the set of patterns P.

The first self-similarity function $\mu(s, l)$ of the set of patterns P is equal to the total number of symbols with the code s($s \in \Sigma$), located at the l-th position of all patterns of this set (the numbering of the position j of the symbol in the pattern is carried out from the end of the pattern to the beginning). For example, for P = {"SHIP", "HIS", "HER", "IN"} μ("S", 4) = 1; μ("H", 3) = 3; μ("I", 2) = 3; μ("P", 1) = 1; μ("S", 1) = 1; μ("N", 4) = 0; μ("N", 3) = 0; μ("N", 2) = 0, where the image of the character in quotes indicates its code in the corresponding encoding.

It is difficult to write down the expression for the first self-similarity function μ in an analytical form; however, using the proposed technique, it is simply to calculate all its values algorithmically to use in the software implementation.

The first partial self-similarity function $\mu_z(s, j)$ of the set of patterns P is equal to the total number of symbols with the code s($s \in \Sigma$), located periodically at positions $(j - 1)\ k\ z$, j = 2, 3, 4, … $(z - 1)$, k = 1, 2, 3, ... $\lceil m_j/z \rceil$ of all patterns of this set (the numbering of the position of the j symbol in the pattern is also carried out from the end of the pattern to the beginning). For example, for z =4 and P = {"43214321", "HGFA4EDA4CBA", "555515555", "277727} μ_z("4", 4) = 4; μ_z("3", 3) = 2; μ_z("2, 2) = 4; μ_z("1", 1) = 3; μ_z("A", 1) = 3; μ_z("A", 3) = 0, where the image of the character in quotes also indicates its code in the corresponding encoding.

The first and second self-similarity functions make it possible to expose redundancy when estimating quantitative characteristics of a recognition schemes based on decoded and partially decoded CAM, respectively [9], [17].

The second self-similarity function ν of the set of patterns P quantitatively determines the degree of coincidence of fragments of different patterns of the signature database. It is also difficult to formulate an analytical notation for the ν function, but it is implicitly present in signature databases and plays an important role in estimating the quantitative characteristics of recognizing schemes based on the Aho–Corasick algorithm [13].

4. Estimation of hardware costs

In order to ensure the methodical rigor of the comparison of technical solutions, it is necessary to bring the calculation of resources of different types to some single conventional unit. Such a unit can be a primitive element of the FPGA internal structure, say, a Look Up Table (LUT). Then it can be assumed that all calculations are performed in conditional LUT (CLT). Using this approach, the value R of the resources required for the synthesis of any digital component will be written in the form:

$$R = L + \alpha F + \beta B + \gamma M, \quad (1)$$

where L – amount of FPGA logics resources, which is required to synthesize this component (in LUTs), F – amount of distributed memory resources (in flip-flops), B – amount of block memory resources (in BRAM blocks), M – volume of on-board memory of reconfigurable accelerator (in Mb), α, β, γ – normalization coefficients of different type resources in relation to LUTs.

The hardware cost estimation function is defined as an expression that, for a given recognition scheme to be synthesized within a reconfigurable accelerator with given parameters, and a given set of patterns P_i to be recognized by this scheme, outputs a numerical estimate of the hardware costs R_i required to build this scheme. The EFs are constructed by direct calculation of all types of resources according to (1).

The work [6] shows how to find the EF expression for the BsCAM matching scheme:

$$R_{\text{BsCAM}} = \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\Lambda(x) j + \left\lceil \frac{j-1}{x-1} \right\rceil \right) + \alpha \left(8 m_{\max} + m_{\min} \left\lceil \frac{\sigma - 1}{y-1} \right\rceil + \sum_{j=m_{\min}+1}^{m_{\max}} \left\lceil \frac{\sum_{i=j}^{m_{\max}} \delta_i - 1}{y-1} \right\rceil \right). \quad (2)$$

where x – the number of inputs of the LUT for a given FPGA, y – FAN-OUT property of internal circuits of the given FPGA, $\Lambda(x) = \begin{cases} 1, & x \geq 8 \\ 2, & x < 8 \end{cases}$ – a qualifier function to define number of LUTs required to realize a 8-bit logic function depending on how many inputs the LUTs have.

As we can see, the formula (2) doesn't contain any self-similarity function. I.e. BsCAM scheme doesn't exploit redundancy available in pattern set. This indicates the low efficiency of the basic scheme in terms of resource consumption.

The expression of EF for the DCAM scheme (we omit intermediate calculations again) has a little more complex appearance:

$$R_{\text{DCAM}} = 256\Lambda(x) + \sum_{j=2}^{m_{\max}} \left(\left\lceil \frac{j-1}{z+1} \right\rceil \sum_{s=0}^{255} \text{NotZ}(\mu(s,j)) \right) +$$

$$+ \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left\lceil \frac{j-1}{x-1} \right\rceil + \alpha \sum_{s=0}^{255} \left\lceil \frac{\sum_{j=2}^{m_{\max}} \text{NotZ}(\mu(s,j)) + \mu(s,j) - 2}{y-1} \right\rceil. \quad (3)$$

The expression (3) already has a self-similarity function, namely, the first self-similarity function μ. This indicates the DCAM scheme is more effective in terms of resource consumption compared to the BsCAM scheme.

The EF for DpCAM scheme looks like even more complex:

$$R_{\text{DpCAM}} = 256\Lambda(x) + \sum_{j=2}^{z-1} \left(\sum_{s=0}^{255} \left\lceil \frac{j-1}{z+1} \right\rceil \text{NotZ}(\mu_{z-1}(s,j)) + \left\lceil \frac{\mu_{z-1}(s,j) - 1}{y-1} \right\rceil \right) +$$

$$+ \sum_{j=m_{\min}}^{m_{\max}} \delta_j \left(\left\lfloor \frac{j}{z-1} \right\rfloor \left(\left\lceil \frac{z}{x} \right\rceil + 1 \right) + \left\lceil \frac{j \bmod (z-1)}{x} \right\rceil + 1 \right) + \sum_{s=0}^{255} \left\lceil \frac{\sum_{j=2}^{z-1} \text{NotZ}(\mu_{z-1}(s,j)) - 1}{y-1} \right\rceil. \quad (4)$$

Nevertheless, this recognizing circuit (which uses the first partial self-similarity function μ_z) is a little more effective compared to the DCAM scheme.

Expressions (3) and (4) look cumbersome, but they are calculated easily and quickly, because they contain mainly addition and rounding operations. The grate advance of using EF to quickly estimate hardware resources is that it is not necessary to perform time consuming procedures of synthesizing digital circuits using a CAD for FPGA project compilation.

5. Experiments

Let's check the proposed technique of handling patterns on the example of the implementation of the parallel combining method [6].

This method makes it possible to reduce the cost of building a pattern recognition module (PRM), which consists of several matching blocks (MB_i), which are built on the basis of recognition schemes

of different nature (Figure 5). It is possible to achieve saving of resources due to the distribution of patterns between recognition blocks in such a way that the advantages of each scheme are highlighted and the disadvantages are eliminated.

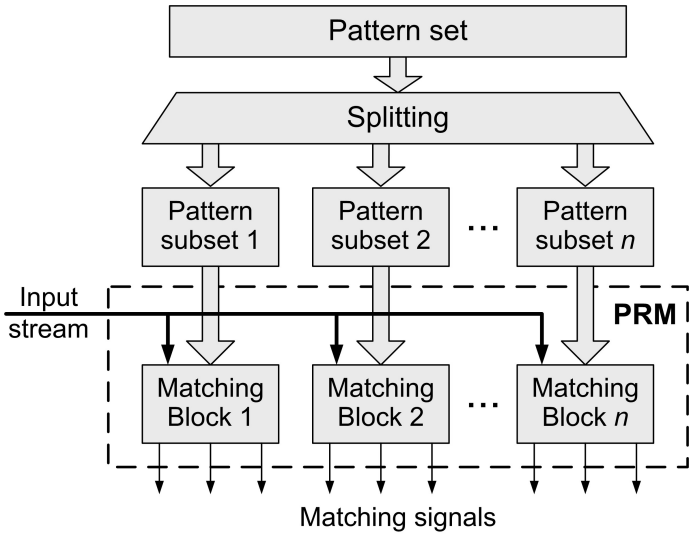

Figure 5 Schematic representation of Parallel combining method

Assume that the PRM of a cybersecurity system consists of two MBs connected in parallel. Their input is concurrent with a stream of data that has to be checked for malicious content. The challenge is to optimally partition the pattern set so that the total amount of hardware required to generate the entire PRM is minimal.

To solve the task, the pattern set can be sorted by some characteristic, for example, by length. The first MB synthesizes to recognize patterns starting from the shortest to some with number *j*. The second MB constructs to recognize the remaining patterns. Sorting can also be done by the number of patterns of the same length. I.e. we assign the 1st portion of patterns to recognize MB_1 is the largest group of patterns with the same number of characters. The 2nd part for MB_1 is the closest packet in size, etc. You can also use a sort order that has the value of multiplying the length of the patterns by the number of them in the packet.

The Figure 6 shows the hardware costs minimization procedure under the following conditions:

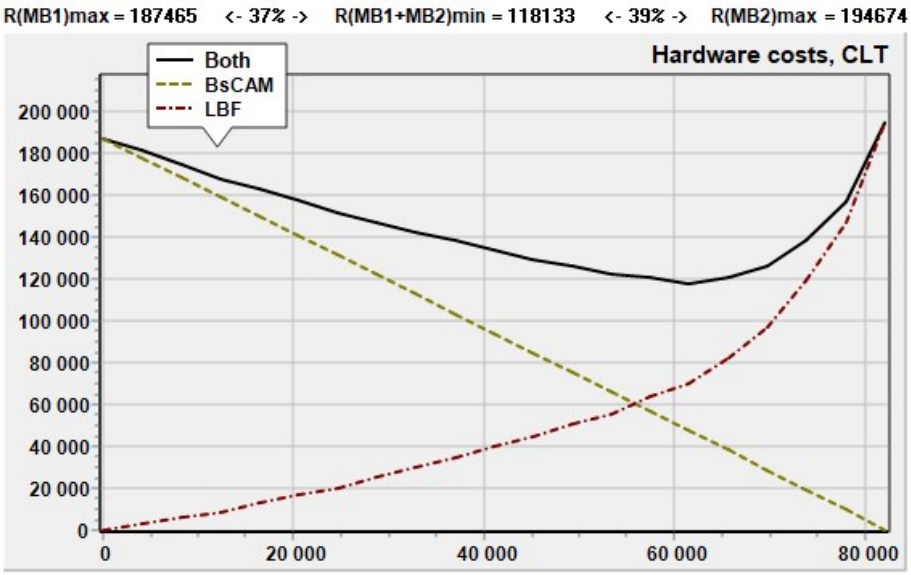

Figure 6 Parallel combining of BsCAM and LBF matching schemes

- Matching scheme of MB_1 – BsCAM, MB_2 – the Large Bloom Filter scheme (LBF);

- Method of ordering – by multiplication the length by quantity type;
- The pattern set – from free signature database "Community Ruleset" of the NIDS Snort, ver. 3.0, which contains $\sigma = 4208$ patterns from $m_{min} = 1$ to $m_{max} = 364$ character length, in total $\Omega = 82081$ characters;
- Reconfigurable accelerator – VC709 Evaluation Kit from Xilinx (Virtex-7 VX690T FPGA, Xilinx).

The abscissa shows the number of characters in the patterns that MB_2 recognizes and MB_1 doesn't recognize. I.e. at the left edge of chart all patterns are matched by MB_1 and none – by MB_2; at the right edge – vice versa.

The ordinate axis shows the amount of hardware costs in CLTs is needed to build every of MBs or the entire PRM. The curve for the MB_1 is depicted by dashed line; for MB_2 – by dot-dash line, and for PRM – by solid line.

The Equipment cost line for PRM has an explicit minimum, at which the entire device requires 37% less resources than the component MB_1 (when it processes all patterns alone), and 39% less than the component MB_2 (when it does).

The Figure 7 depicts the experiment results under the same conditions, but MB_i are built now using not basic, but modified matching schemes: MB_1 created by DpCAM scheme based on CAM and comparators whereas the MB_2 – by Simplified Bloom Filter (SBF) scheme.

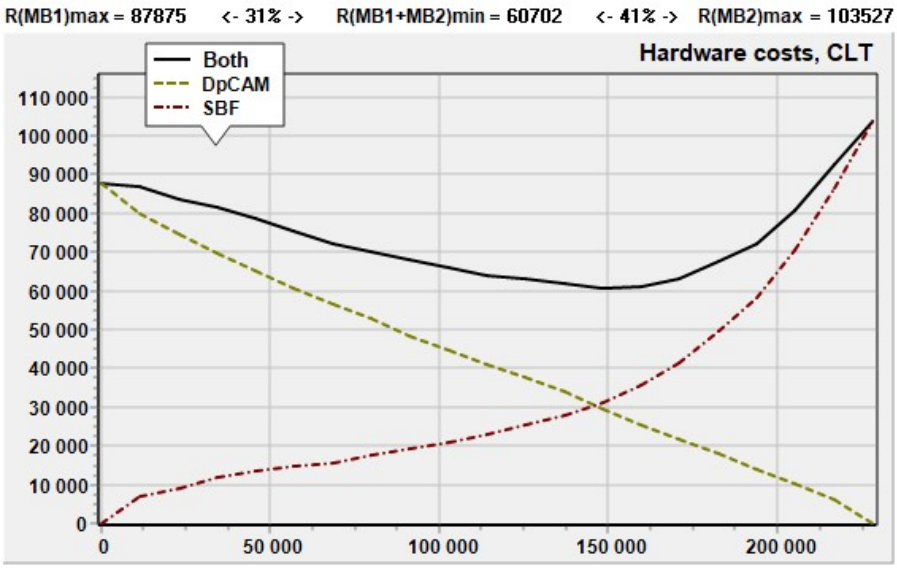

Figure 7 Parallel combining of BsCAM and LBF matching schemes

As we can see, the cost curve for the second case, when the PRM constructed from more effective modifications of matching schemes, has a pronounced minimum as well. The entire device shows 31% less costs requirements than the MB_1 (if it recognizes all the patterns), and 41% less than the component MB_2. On the other hand, the absolute values of hardware costs are approximately two times less than for basic circuits.

6. Conclusion

The task of multi-pattern string matching that signature-based security systems perform is computational-intensive, and software solutions on traditional processors do not provide necessary performance already. Hardware accelerators using FPGA are suitable platform for this purpose. Content addressable memory based on digital comparators, Bloom filter based on hash-functions and Aho–Corasick algorithm implemented as a finite automaton are most promising approaches to building matching schemes. Researchers over the world have proposed a lot of modifications and improvements to the basic solutions of each approach. In fact, pattern set is "hardwired" into the

structure of the recognition circuit. Therefore, its properties are very important and have to be formalized to effectively operate with matching schemes, and first of all. A technique for sorting patterns in a set is needed.

The contribution of this work is as follows. A pattern set description formalization and a technique to order patterns that simplifies calculating quantitative characteristics of recognition schemes are proposed. This allows applying more effective modifications of matching schemes to build pattern recognition modules for signature-based cybersecurity systems. Due to the use of estimation functions an expensive procedure of digital circuit synthesis using CAD tool can be eliminated.

Acknowledgment

This work was supported in part by the Informatization Program of the National Academy of Sciences of Ukraine for 2020-2024.

References

[1] B. Smyth, Computing Patterns in Strings, Pearson Addison Wesley, Essex, 2003.
[2] R. Abdulhammed, M. Faezipour, K. M. Elleithy, Network Intrusion Detection Using Hardware Techniques: A Review, in: Proceedings of the IEEE Long Island Systems, Applications and Technology Conference (Lisat), 2016, p. 7.
[3] V. Jyothi, S. K. Addepalli, R. Karri, DPFEE: A High Performance Scalable Pre-Processor for Network Security Systems, IEEE Transactions on Multi-Scale Computing Systems, Article volume 4, 1 (January-March 2018) 55-68. doi:10.1109/tmscs.2017.2765324.
[4] S. Y. Hilgurt, A survey on hardware solutions for signature-based security systems, in: 1st International Workshop on Information Technologies: Theoretical and Applied Problems 2021 (ITTAP 2021), vol. 3039: CEUR-WS, 2021, pp. 6-23, Available online: http://ceur-ws.org/Vol-3039.
[5] S. Hilgurt, A Concise Review of FPGA-Based Hardware Solutions for Network Intrusion Detection, in: 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 164-168, doi: 10.1109/PICST54195.2021.97.
[6] S. Hilgurt, Parallel combining different approaches to multi-pattern matching for FPGA-based security systems, Advances in cyber-physical systems, volume 5, 1 (2020) 8-15.
[7] S. A. Guccione, D. Levi, D. Downs, A reconfigurable content addressable memory, in: Proceedings of the Parallel and Distributed Processing, vol. 1800, 2000, pp. 882-889.
[8] Y. H. Cho, W. H. Mangione-Smith, Deep packet filter with dedicated logic and read only memories, in: Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2004, pp. 125-134. doi:10.1109/fccm.2004.25.
[9] I. Sourdis, D. N. Pnevmatikatos, S. Vassiliadis, Scalable multigigabit pattern matching for packet inspection, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Article volume 16, 2 (February 2008) 156-166. doi:10.1109/tvlsi.2007.912036.
[10] B. H. Bloom, Space/Time Trade-offs in Hash Coding with Allowable Errors, Communications of the ACM, Article volume 13, 7 (1970) 422-426. doi:10.1145/362686.362692.
[11] S. Dharmapurikar, M. Attig, J. Lockwood, Design and Implementation of a String Matching System for Network Intrusion Detection using FPGA-based Bloom Filters, All Computer Science and Engineering Research, Report Number: WUCSE-2004-12, 2004-03-25, Washington University in St. Louis, 2004.
[12] S. Geravand, M. Ahmadi, Bloom filter applications in network security: A state-of-the-art survey, Computer Networks, Article volume 57, 18 (December 2013) 4047-4064. doi:10.1016/j.comnet.2013.09.003.
[13] A. V. Aho, M. J. Corasick, Efficient String Matching: An Aid to Bibliographic Search, Communications of the ACM, vol. 18, 6 (1975) 333-340. doi:10.1145/360825.360855.
[14] J. Lunteren, High-performance pattern-matching for intrusion detection, in: Proceedings of the 25th IEEE International Conference on Computer Communications, volumes 1-7, 2006, pp. 1409-1421.

[15] W. Jiang, Y. H. E. Yang, V. K. Prasanna, Scalable multi-pipeline architecture for high performance multi-pattern string matching, in: Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2010. doi:10.1109/IPDPS.2010.5470374.
[16] C. H. Lin, S. C. Chang, Efficient Pattern Matching Algorithm for Memory Architecture, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Article volume 19, 1 (January 2011) 33-41. doi:10.1109/tvlsi.2009.2028346.
[17] I. Sourdis, D. Pnevmatikatos, Pre-decoded CAMs for efficient and high-speed NIDS pattern matching, in: Proceedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, Proceedings, 2004, pp. 258-267. doi:10.1109/fccm.2004.46.