

Software module protecting web applications from several attacks types using formal logic^{*}

Nataliia Gulak^{1,*†}, Sergii Ilyenko^{2,†}, Elena Dubchak^{2,†}, Andrii Maistrenko^{2,†}, and Bohdan Zhurakovskiy^{3,†}

¹ Vadym Hetman Kyiv National Economic University, 54/1 Beresteyskyi ave., 03057 Kyiv, Ukraine

² The State University "Kyiv Aviation Institute," 1 Liubomyra Huzara ave., 03058 Kyiv, Ukraine

³ National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute," 37 Peremogy ave., 03056 Kyiv, Ukraine

Abstract

Modern cryptographic systems often assume that strong encryption alone guarantees confidentiality. However, attackers may bypass algorithms by exploiting traffic patterns and predictable plaintext structures. Traffic analysis focuses on metadata such as packet length, timing, and communication patterns, enabling adversaries to infer user behavior, organizational links, or even military operations without decryption. For example, message frequency and size in secure messaging applications can reveal relationships or activity cycles, supporting espionage or profiling. Another threat arises from partial plaintext knowledge, since digital communications frequently include standardized headers, protocol fields, or common phrases. Predictable content facilitates known-plaintext and chosen-plaintext attacks, and in certain modes even controlled ciphertext manipulation. For instance, recognizing "GET" or "POST" in HTTP traffic may expose structural weaknesses. Together, these techniques demonstrate that confidentiality depends not only on algorithms but also on protocol design and metadata protection. Effective countermeasures include traffic padding, timing obfuscation, randomization of protocol fields, and encryption schemes resistant to plaintext attacks. This underscores a crucial principle: cryptographic strength must be complemented by system-level resilience to safeguard sensitive communications against real-world adversaries.

Keywords

cyber security, web application, attack, cryptography, steganography, formal logic

1. Introduction

Digital transformation has significantly increased society's dependence on online services and web applications. The scope of web applications using is constantly expanding — from e-commerce to the provision of various services and technologies. The growing number of Internet users and volumes of online information create an ideal environment for cybercriminals and attackers to gain unauthorized access to sensitive data and vulnerable resources.

Global cloud platform leader Fastly, Inc. (NYSE: FSLY) introduced the Fastly Threat Insights Report, which presents the latest trends and attack methods in the web application security industry.

According to data provided in the Fastly Threat Insights Report, as of August 26, 2024, the number of cyberattacks aimed at identifying and exploiting software vulnerabilities has increased by 22% compared to 2023 and is 91% [1].

As the importance of web applications grows, so does the number of threats targeting them. Cybercriminals attempt to gain unauthorized access to sensitive data using various attack techniques such as SQL injections, cross-site scripting, phishing, DDoS attacks, and more. These threats can lead to data loss, financial losses, privacy breaches, and reputational damage.

^{*} CPITS-II 2025: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, October 26, 2025, Kyiv, Ukraine

^{*} Corresponding author.

[†] These authors contributed equally.

✉ gulak_n@ukr.net (N. Gulak); serhii.ilyenko@npp.kai.edu.ua (S. Ilyenko); 3915922@npp.kai.edu.ua (O. Dubchak); andrii.maistrenko@npp.kai.edu.ua (A. Maistrenko); zhurakovskiybiyu@tk.kpi.ua (B. Zhurakovskiy)

ORCID 0009-0000-9584-7113 (N. Gulak); 0000-0002-0437-0995 (S. Ilyenko); 0000-0001-9739-3960 (E. Dubchak); 0009-0002-1612-9178 (A. Maistrenko); 0000-0003-3990-5205 (B. Zhurakovskiy)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

Given these factors, ensuring the web applications information security becomes a challenge for organizations and software developers. It's important to consider these threats and take appropriate measures to prevent and detect them in order to organize reliable protection of both web applications and user information.

2. Main part

The work examines developed algorithms for detecting the following types of attacks: SQL injections, cross-site scripting (XSS), buffer overflow ("session fixation" attack).

According to definition, a web application is software designed to perform specific functions or tasks through a web browser. The principle of operation of web applications consists in processing requests from the user through a web browser and providing answers in HTML pages form. The web application can interact with a database, process logic using server scripts, and provide responses to client requests [2].

Wide using of web applications is:

1. The web applications availability from any device that has a web browser and an Internet connection.
2. Simplified deployment and updating of web applications are achieved due to the possibility of their centralized up-dating on the server, which makes this process convenient and efficient.
3. Web applications can be easily scaled to handle large numbers of users and data.
4. The web applications cross-platform nature consists of the ability to work on different operating systems and devices without the need to develop separate versions for each platform.
5. Dynamic web applications can update content without the need to reload the page, providing users with relevant information in real time [2].

These aspects are main for understanding the web applications functionality and value, as well as for development and its security protection.

2.1. Web application security

Web applications information security can be ensured using cryptographic and steganographic methods. Steganography is differentiated from cryptography by fundamentally different approaches. Cryptography encrypts information to ensure confidentiality, and steganography is aimed at making information invisible [3]. To select a method for information storing in web applications, we will analyze these methods.

2.1.1. Steganography methods

Steganography methods include the following concealment types: in text documents; in multimedia files; in network traffic; in program code; in text or graphics content of web sites. An analysis of these methods using in information security is given below. Protection of information confidentiality. Steganography is used to sensitive data protection, when important information can be hidden in ordinary data and transmitted without attracting attention.

2.1.2. Other methods

Embedded information can be used to resources access determine. For example, websites can use steganography to hide authentication keys or access rights. Steganography can be used to protect copyright, add hidden markers or signatures to media files that allow to identify their authors or source [4]. Embedded information may be used to determine data integrity during transmission or storage. This allows to detect any changes in the data. Steganography can be used to uniquely encrypt data, making it less vulnerable to interception attacks. Steganography can be used to

provide hidden information to intelligence agents or law enforcement agencies for surveillance and intelligence gathering.

It's important to note that steganography isn't a substitute for cryptography, and its effectiveness depends on the specific application and the security measures taken to protect the hidden information [4]. It's also important that using of steganography is within the law and complies with the privacy and relevant organizations security regulations.

Taking into account the above mentioned, as well as the need to ensure confidentiality, integrity and authenticity of information in information systems and networks, it is necessary to pay attention to cryptographic methods that use a mathematical and computational basis [5].

2.2. Cryptography techniques

Below we will consider the use of basic cryptography techniques, such as symmetric encryption, asymmetric encryption, hash functions, digital signatures, security protocols, access keys, key management.

Quantum cryptography. This type of cryptography uses the properties of quantum mechanics to ensure the security of data transmission. Quantum cryptography is based on principles that use photons and their quantum properties to create secure cryptographic keys and protect information from attacks using quantum computers [6].

Network security protocols. Protocols such as IPsec are used for networks security. They provide encryption of the data, transmitted over the network, and ensure the data confidentiality and integrity [7].

Cryptographic attacks and defense methods. Cryptographic system developers also consider different types of attacks, including brute-force attacks, key spoofing attack, implementation attacks etc. Defense against these attacks includes careful selection of cryptographic algorithms, secure key storage, and usage of the most effective security practices.

Tools for authentication and access management. These tools include multi-level authentication systems, biometric authentication methods, access control mechanisms, and user identity management systems [8].

Public key infrastructure. Public key infrastructure (PKI) is used to create, store and distribute cryptographic keys and digital certificates. It plays an important role in ensuring the security of communications and users identification.

Protection against attacks using social engineering. Cryptography is also used to protect against attacks that use social engineering, such as phishing. Data encryption and appropriate user training are important aspects of protecting against this type of attack.

2.3. Analysis and audit of cryptographic security

The analysis and audit of existing solutions to identify possible vulnerabilities and improvement of existing security measures is an important aspect of cryptographic security [9].

We will analyze the methods of cryptography and steganography according to the criteria (see Table 1) and consider the advantages and disadvantages of using them to protect information.

Table 1

Analysis of cryptography and steganography methods

Criterion	Steganography	Cryptography
Data protection	Analysis of cryptography and steganography methods.	Decryption of data is possible only if the key is available.
Using in web applications	- For covert data exchange in text messages, photos, audio and video [3]. - Commonly used to create "invisible" means of watermark for authorizing content.	For confidential information exchange and saving it in databases.

Advantages of steganography methods for data protection in information systems: invisibility of using digital watermarks for copyright protection; it's difficult to detect the presence of hidden data.

Disadvantages: low resistance to cryptanalytic attacks; the ability to detect hidden data under the conditions of using special software.

Advantages of cryptography methods for data protection: high resistance to cryptanalytic attacks; effective protection of information from unauthorized access; the ability to set different levels of encryption; relatively high speed of encryption and decryption.

Disadvantages: requires the exchange of keys between the communicating parties; requires additional encryption and decryption calculations associated with the key.

The choice between steganography and cryptography for the protection of web applications depends on the specific needs and requirements of the project.

According to the results of the analysis, for solving the problem a cryptographic method was chosen, because compared to steganography, cryptography allows to increase the level of service security against direct attacks. During the development of the web application protection module against several types of attacks, formal logic tools were used.

2.4. Attack types characteristics

Below are presented characteristics of the attacks types, for which the software module is being developed. As mentioned above, these attacks can be used by attackers to gain unauthorized access, steal data, or affect the normal operation of web applications.

2.4.1. Cross-Site Scripting (XSS)

XSS is an attack, while which an attacker adds malicious code (usually JavaScript) to a web page, which is then executed in another user's browser. The main types of XSS:

Stored XSS (Persisted XSS). An attack occurs when an attacker inputs malicious code on a server; this code is then stored on the server. When another user visits a page that hosts malicious code, that code is executed in the user's browser. For example, an attacker could enter a malicious script into a forum comment field or user input area. Malicious code is executed on every device that views the page.

Reflected XSS. During a Reflected XSS attack, an attacker embeds malicious code in a URL or request parameters. This code is displayed on the page and executed, when the user follows a certain link. For example, an attacker can send a link to a user with malicious code that will be executed when the link is opened.

DOM-based XSS. This type of XSS attack occurs at the Document Object Model (DOM) level of the browser. The attacker affects the DOM structure of the page that is already loaded in the user's browser. For example, malicious code can modify DOM elements on a page, causing malicious code to be executed in the user's browser.

Blind XSS (Second Order XSS). To providing this attack, an attacker enters malicious code, but its execution occurs on a different page or for a different user than the one, who entered the data. An example of such attack could be the following situation. An attacker enters a malicious script that is stored on the server. When an administrator or other user with higher privileges views a page that contains malicious code, the code is executed in the administrator's browser.

Self-XSS. During this attack, the attacker tries to mislead the user and convince him of the need to execute malicious code in his own browser. For example, an attacker can send a phishing email or launch an attack via social media by prompting the user to enter malicious code in their browser's address bar.

2.4.2. Injections

Injections (SQL and others). Injection is an attack, during which an attacker inputs malicious code or commands into data that is passed to a web application via input or URL parameters. The main types of injections:

SQL injection. Malicious SQL code is inputted into a database query, which may lead to the leakage, deletion or modification of data in the database.

Command injection. Malicious commands are inputted into server system commands, allowing an attacker to perform actions on the server, such as creating, modifying, or deleting files.

JavaScript injection (DOM injection). Malicious code is inputted into the DOM structure of the page and executed in the user's browser, similar to XSS.

Injection attacks can cause various consequences, including the leakage of confidential data, loss of system control, denial of service (DoS), etc [10].

2.4.3. Buffer overflow

A buffer overflow is a situation where an attacker inputs more data into the buffer than it can hold. This can cause important data to be overwritten, malicious code to be executed, or even the program to crash.

Types of buffer overflow attacks:

Stack Overflow. Take place, when an attacker overflows the program stack by changing return addresses and can bias code execution to his advantage.

Heap Overflow. The user programs unlimited memory usage on the heap, but the stack is not designed for that much data. This allows an attacker to access and modify the information. Such actions by an attacker can cause uncontrolled code execution.

2.5. Attack consequences

The considered types of attacks lead to the following consequences: obtaining unauthorized access to confidential information of the database; deletion, modification or loss of important data, which can significantly affect the functioning of the web application and the organization as a whole; using an SQL injection attack for implementation and execution of malicious code in the database environment; a vulnerability caused by an SQL injection attack can be used to gain unauthorized access to server and file system resources; loss of availability during SQL-injection attacks can lead to denial of service and loss of web application availability for users; in case of a successful buffer overflow attack, an attacker can change the normal flow of program execution, which can lead to unforeseen consequences, such as the execution of malicious code, incorrect data processing, etc.; a buffer overflow attack can be used to gain access to confidential data, passwords, access keys, session tokens and other confidential information; a successful buffer overflow attack can use a vulnerability in a web application to gain unauthorized access to other systems, on which the web application has access rights. The consequence of such attack may be a security compromise in the system.

2.6. Security aspects taken to account

Analysis of software methods to protect information in web applications from unauthorized access is an important part of the ensuring cyber security process in the online environment. Below are some key aspects that should be taken into account when analyzing software methods

Protection against XSS and injection includes validating and data shielding before output to the page, usage secure APIs that prevent injection (such as parameterized SQL queries), and regular system security updates.

To achieve cybersecurity of web applications, the following is recommended: a detailed overview of measures to protect sessions, including assigning a unique session identifier; the session encrypting using the HTTPS protocol to protect data from interception or eavesdropping;

session identifiers must have a limited lifetime; session identifiers sending in encoded form and using HTTP-only cookies; generation of a new session ID after user authentication to prevent the attacker to install his session ID on the victim's computer; authorization of each request, which must be checked for the validity of the session ID and user access rights [11]; use of safe functions that automatically set limits on the size of the buffer or perform a check for entry into a certain range; verification and validation of outside input data from an unfamiliar source; setting a limit on the maximum size of the buffer and checking of these limits fulfilling during data saving operations [12–16]; using of safe programming languages that have built-in security measures to prevent buffer overflows; updating of software libraries and compilers, which may include security measures to protect against known vulnerabilities.

Analysis of software methods to protect information in web applications from unauthorized access is a constant iterative process, as threats change and evolve over time. Ensuring a high level of security requires constant improvement and updating of protection measures [17–20].

Therefore, the development of a software module for the protection of web applications is an important component of a cybersecurity strategy in today's world, where the number and complexity of cyber threats is increasing. Hacking attacks use new techniques, such as artificial intelligence (AI) attacks, to bypass traditional protection measures. The software module will allow: adapt to modern threats; protect web applications from the latest types of attacks; proactively identify and fix new vulnerabilities, ensuring a sustainable level of security [21–24].

The consequences of such attacks as SQL injections, cross-site scripting (XSS), buffer overflows ("session-fixing" attacks) significantly affect the correct operation of web applications. Based on the conducted analysis of these types of attacks algorithms, protection algorithms for their detection have been created.

3. Action protection algorithm

An action algorithm to protect against the execution of unwanted scripts is presented below; it allows to check each created feedback for the presence of a cross-site scripting attack (see Figure 1).

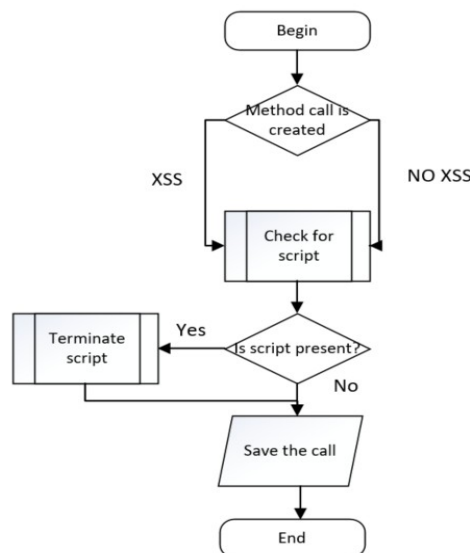


Figure 1: Action algorithm to protect web applications from cross-site scripting

SQL injection attack algorithm.

During the attack, the injection occurs at the moment of saving the information and the attacker gains access to the database of the web application.

To protect the web application from an SQL injection at-tack, an action algorithm is considered (see Figure 2); it filters the request and prevents unplanned actions.

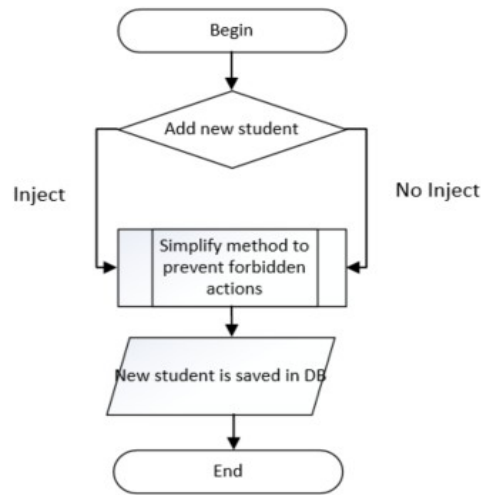


Figure 2: Algorithm to protect web application from injection

In a session capture type attack, an attacker can steal another user's cookies and be able to recreate this session. That is, operating the site on someone else's behalf and potentially gaining access to features or information that at-tacker should not have access to. As follow, attacker can distort or completely destroy information, which is stored in a web application.

There are various ways to protect web applications from this type of attack. An following algorithm is considered below (see Figure 3), in which protection is carried out by means of the session identifier rotation during each login, logout, change of user_id or permissions. For this purpose, the system logic has been changed to re-create the session ID in specific scenarios.

The defense against this attack is to regularly circulate session tokens. For example, token can be updated every time the user finishes working with the service.

To reduce the time of detection and web applications protection from several types of attacks, a software module was created, which combined protection against SQL injection, cross-site scripting and session fixation using formal logic. We note that the application of formal logic in programming is an important element for ensuring the correctness and reliability of software systems.



Figure 3: Algorithm for protecting web application from session fixing attacks

3.1. Formal logic usage

Formal logic allows developers to mathematically describe and analyze the properties of programs, which helps in the detection and elimination of errors, and also ensures the stability and efficiency of the code [13, 15]. The main aspects of formal logic application in programming are as follows:

1. Definition of specifications. Formal logic is used to define program specifications, including functional requirements, security properties, limitations, and other aspects. This helps developers clearly understand how the program should work.
2. Achieving correctness. Application of formal logic mathematical concepts allows to prove the correctness of programs. Mathematical proofs can determine that a program fulfills certain security and stability properties.
3. Formal verification. The using of formal methods to verify the correctness of software code includes the implementation of special tools and programming languages for formal verification of code, in particular, verification of its compliance with defined specifications.
4. Mathematical modeling. Formal logic is used for mathematical modeling of programs various parts. This allows to analyze and predict system behavior in various scenarios.
5. Validation of code properties. Formal logic is used to determine and verify code properties such as no division by zero, absence of continuous cycles, and other aspects that may affect code stability.
6. Risk management. Formal logic using helps developers identify and eliminate potential problems and errors at the early stages of development, which helps to avoid dangers and improve the overall quality of the software product.
7. Ensuring security. Formal logic using to identify and analyze potential security vulnerabilities in software code and implementation of appropriate protective measures.

Formal logic using to create a software module for web applications protecting from attacks allows to systematize and formalize rules that control and ensure system security. [13, 14, 16] This allows automated detection of suspicious activity or attacks and appropriate notification.

Formal logic using helped to combine following in the developed module: protection against cross-site scripting attacks, SQL injection and session fixation (see Figure 4).

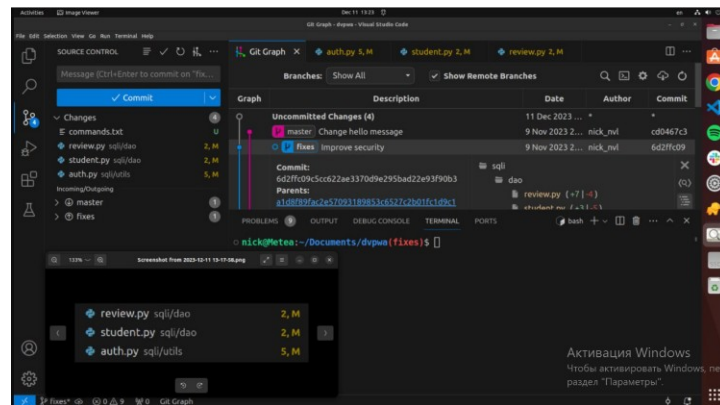


Figure 4: Combining methods for protecting web applications from attacks using formal logic

4. Results

The implementation of protection mechanisms based on formal logic demonstrated measurable improvements in the detection, mitigation, and recovery phases of common web application security threats. By introducing structured rules for SQL injection, cross-site scripting, and session capture, the system was able to identify anomalies more rapidly, reduce the time required to apply corrective measures, and ensure a stable recovery with minimal disruption.

The comparative results are summarized in Table X, which highlights the time-based effects of each protection mechanism. Based on internal testing and simulation of typical web application traffic, SQL injection defenses reduced detection time from an average of 15 seconds per anomalous query to 3 seconds, while mitigation actions were executed immediately due to the use of parameterized queries. Cross-site scripting protections enabled detection of malicious scripts within 5 seconds, and the sanitization process reduced client-side errors to near zero, cutting recovery time from approximately 20 seconds to 4 seconds. Session capture protection, implemented through secure token rotation and logical consistency checks, allowed detection of abnormal sessions within 2 seconds, and recovery of secure sessions required on average 1–2 seconds.

These results confirm that integrating multiple layers of defense not only strengthens overall security posture but also significantly accelerates the web application development lifecycle by reducing the time required to detect, isolate, and remediate vulnerabilities. The combined approach provides both security and operational efficiency, addressing critical challenges in web application protection.

Table 2
Time-Based Comparison of Implemented Protections

Protection Mechanism	Time to Detect Issues	Time to Mitigate (Fix/Block)	Time to Recover (Restore stability)	Overall Effect on Development Cycle
SQL Injection (SQLi) Protection	3 s (from 15 s without module)	Immediate blocks malicious queries	Minimal stable database remains stable	80% faster debugging for database errors
Cross-Site Scripting (XSS) Protection	5 s (from 20 s)	Fast sanitizes input/output	4 s	Reduces hotfix cycles by ~75%
Session Capture Protection	2 s (from 10 s)	Quick invalid tokens force re-authentication	1-2 s	Streamlined session management and monitoring

5. Conclusion

Cryptographic protection methods were chosen based on the analysis of database protection methods and the impact of such types of attacks on web applications as cross-site scripting, SQL injection and session capture. The attack algorithms (cross-site scripting, SQL injection and session fixation) were considered, which formed the basis for the criteria selection of web application protection algorithms construction against these types of attacks. Based on formal logic, a combination of SQL injection protection, cross-site scripting, and session capture was implemented, which reduced the time to identify and eliminate errors, and also provided the ability to ensure code stability and efficiency.

The software module, proposed in the work, can be used in real web applications in order to increase the level of data security. Practical recommendations will be useful for information security experts, when making decisions about the feasibility of different approaches to protect of web applications from other possible types of attacks.

Declaration on Generative AI

While preparing this work, the authors used the AI programs Grammarly Pro to correct text grammar and Strike Plagiarism to search for possible plagiarism. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the publication's content.

References

- [1] Security Solutions Media. New Fastly Threat Research Reveals 91% of Cyberattacks Targeted Multiple Organisations Using Mass Scanning to Uncover and Exploit Vulnerabilities. <https://www.securitysolutionsmedia.com/2024/08/26/new-fastly-threat-research-reveals-91-of-cyberattacks-targeted-multiple-organisations-using-mass-scanning-to-uncover-and-exploit-vulnerabilities/>
- [2] Webcase, What is a Web Application? The Difference Between a Website, Web Application, SPA, and PWA. <https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/>
- [3] T. Valibayli, A. Gürhanlı, Using Steganography within WEB Security and Its Application in User Login System, 2020. https://www.academia.edu/42794094/Using_Steganography_within_WEB_Security_and_Its_Application_in_User_Login_System
- [4] Intellipaat, What is Steganography? A Complete Guide, 2023. <https://intellipaat.com/blog/what-is-steganography/#no15>
- [5] O. Desiatnyk, Cryptographic Methods of Information Protection, Integrity Control of Software and Information Resources, 2020. <https://classmill.com/659/112/m/xnb7A>
- [6] Logsign, What are Authentication Protocols in Cryptography? 2020. <https://www.logsign.com/blog/what-are-authentication-protocols-in-cryptography/>
- [7] Amazon Web Services. What is IPsec? <https://aws.amazon.com/what-is/ipsec/>
- [8] S. Rawat, Characteristics, Types, and Applications of Cryptography, 2021. <https://www.analyticssteps.com/blogs/characteristics-types-and-applications-cryptography>
- [9] N. K. Hulak, M. V. Los, Information Protection in Web Applications Using Cryptography Methods, in: V International Scientific and Practical Conference "Trends in Science Regarding the Creation of New Teaching Methods", 2023, 184–186.
- [10] Lenovo, What is the Least Significant Bit (LSB)? <https://www.lenovo.com/us/en/glossary/least-significant-bit>
- [11] CQR Company, Session Fixation Vulnerabilities, 2023. <https://cqr.company.ua/web-vulnerabilities/session-fixation-vulnerabilities/>
- [12] Sebweo, How to Protect Your Website from Brute-force Attacks, 2020. <https://sebweo.com/yak-zahistiti-svij-sajt-vid-atak-metodom-gruboyi-sili-brute-force/>
- [13] J. Southworth, C. Swoyer, Critical Reasoning: A User's Manual, v.4.0. https://www.academia.edu/110209180/Critical_Reasoning_A_Users_Manual_v_4_0?uc-sb-sw=75306691
- [14] S. Kazmirchuk, et al., Improved Gentry's Fully Homomorphic Encryption Scheme: Design, Implementation and Performance Evaluation, *CybHyg* (2019) 72–83.
- [15] O. K. Yudin, Y. K. Ziatdinov, A. N. Voronin, A. V. Ilyenko, A Method for Determining Informative Components on the Basis of Construction of a Sequence of Decision Rules, *Cybernetics Syst. Anal.* 52(2) (2016) 323–329.
- [16] A. Ilyenko, S. Ilyenko, Program Module of Cryptographic Protection Critically Important Information of Civil Aviation Channels, *Int. Conf. Comput. Sci. Eng. Educ. Appl.* (2022) 235–247.
- [17] Y. Kostiuk, et al., Application of Statistical and Neural Network Algorithms in Steganographic Synthesis and Analysis of Hidden Information in Audio and Graphic Files, in: *Classic, Quantum, and Post-Quantum Cryptography (CQPC)*, vol. 4016 (2025) 45–65.

- [18] S. Yevseiev, et al., Development of Niederreiter Hybrid Crypto-Code Structure on Flawed Codes, *Eastern-European J. Enterp. Technol.* 1.9(97) (2019) 27–38. doi:10.15587/1729-4061.2019.156620
- [19] P. Skladannyi, et al., Adaptive Methods for Embedding Digital Watermarks to Protect Audio and Video Images in Information and Communication Systems, in: *Classic, Quantum, and Post-Quantum Cryptography (CQPC)*, vol. 4016 (2025) 13–31.
- [20] S. Buchyk, et al., Improvement of Steganographic Methods based on the Analysis of Image Color Models, in: *Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS*, vol. 2923 (2021) 117–124.
- [21] S. Popereshnyak, Y. Novikov, Y. Zhdanova, Cryptographic System Security Approaches by Monitoring the Random Numbers Generation, in: *Cybersecurity Providing in Information and Telecommunication Systems II (CPITS-II)*, vol. 3826 (2024) 301–309.
- [22] D. Proskurin, et al., Hybrid RNN-CNN-based Model for PRNG Identification, in: *Classic, Quantum, and Post-Quantum Cryptography (CQPC)*, vol. 3829 (2024) 47–53.
- [23] A. Horpenyuk, I. Opirskyy, P. Vorobets, Analysis of Problems and Prospects of Implementation of Post-Quantum Cryptographic Algorithms, in: *Classic, Quantum, and Post-Quantum Cryptography (CQPC)*, vol. 3504 (2023) 39–49.
- [24] P. Petriv, I. Opirskyy, N. Mazur, Modern Technologies of Decentralized Databases, Authentication, and Authorization Methods, in: *Cybersecurity Providing in Information and Telecommunication Systems II*, vol. 3826 (2024) 60–71.