Computer Science and Interdisciplinary Research Journal
Volume 1, Issue 2, 2024. ISSN: 3033-1218, DOI: 10.70862/CSIR.2024.0102-09

Opportunities to Achieve Cyber Security Through Blockchain and Smart Contracts

Margarita Terziyska [1], Zhelyazko Terziyski [2],
Valentina Nikolova-Alexieva [3], Katina Valeva [4]

[1, 3, 4] University of Food Technologies, 4002 Plovdiv, Bulgaria
[2] Trakia University, 6000 Stara Zagora, Bulgaria

[1] mterziyska@uft-plovdiv.bg, [2] zheyazko.terziyski@trakia-uni.bg,
[3] v_aleksieva@uft-plovdiv.bg, [4] k_valeva@uft-plovdiv.bg

Abstract – In the context of growing threats in the digital space, cyber security has established itself as a critical component for protecting information systems. With the increasing reliance on digital technology and the Internet, the risks of cyber-attacks and unauthorized access to sensitive data have greatly increased. This highlights the need to develop advanced security mechanisms that ensure the overall safety of systems. Traditional security methods often fail to cope with the complexity of modern threats. Organizations are therefore exploring new technologies, such as blockchain and artificial intelligence, that offer higher levels of security. Blockchain provides innovative solutions that decentralize identity management and reduce dependence on centralized servers. This paper aims to present a methodology combining two blockchain authentication methods – MetaMask Authentication and Smart Contract-Based Authentication, for implementation and analysis, ensuring overcoming challenges in real applications.

Keywords – Cybersecurity, Blockchain, Smart Contracts, Authentication, Decentralized Applications.

DOI: https://doi.org/10.70862/CSIR.2024.0102-09

Corresponding author: Margarita Terziyska, University of Food Technologies
Email: mterziyska@uft-plovdiv.bg

Received: 18 August 2024.
Revised: 24 February 2025.
Accepted: 26 February 2025.
Published: 26 February 2025.

© 2024 Margarita Terziyska, Zhelyazko Terziyski, Valentina Nikolova-Alexieva & Katina Valeva. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

The article is published with Open Access at http://csirjournal.com/

1. Introduction

The Internet has undergone a significant transformation in recent decades. What started as static web pages has evolved into a web of intelligent and immersive experiences. From the beginning of the 1990s to today, the evolution of the web has passed through four stages and puts us on the threshold of the fifth - Web 5.0. [1].

Web 1.0, is called the static web. This stage is characterized by the appearance of static sites without interactive features. The information provided in them can only be read by users.

Web 2.0, is called the social network. This web generation has been developing since the beginning of the 21st century. It is characterized by the emergence of social media and the generation of content by users who can interact, collaborate, or share information, photos, links, or videos.

Web 3.0 is a stage of development known as the Semantic Web. This network brings semantic integration and data interoperability. It is characterized by the application of artificial intelligence and machine learning that understand and interpret human language. Technological developments contribute to the ability of websites to provide more personalized experiences for users.

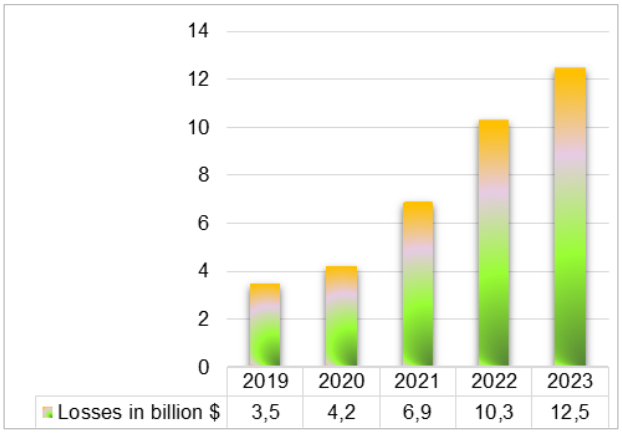
Web 4.0 is associated with a transition from physical to digital technologies. It is in the early stages of development. The aspiration in this stage is through the use of AI, big data, semantic web social networking sites, etc., to develop the Internet to be easier and more convenient to use, more efficient, and personalized for the user.

Web 5.0 is under development. It is seen as a concept that promises to change the way we use and perceive the Internet. At the core of Web 5.0 are modern information technologies such as machine learning, artificial intelligence, and blockchain, which

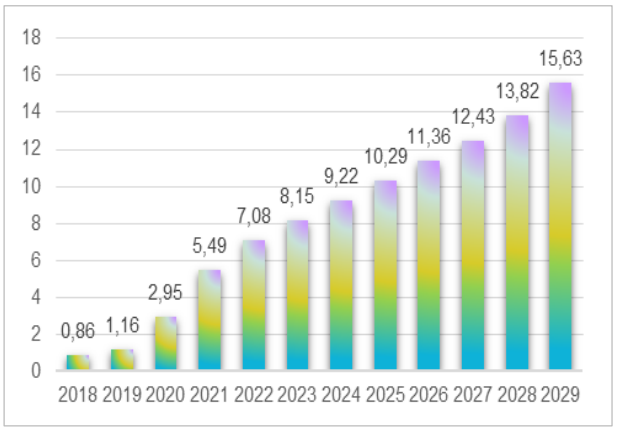
introduce decentralization, increase personal data security, and improve Internet services' sustainability. This is of utmost importance in the era of digitization and global connectivity when cyber security has become critical.

Information that is stored and transmitted in digital form is becoming increasingly valuable. This makes it a prime target for cybercrimes. Data of companies, government agencies, and individual users are constantly under threat from hackers, malware, phishing attacks, and other forms of cyberattacks.

According to the FBI's latest IC3 Internet Crime Report, 880418 complaints were reported in 2023. [2]. The data shows that there has been an annual increase in complaints over the past five years. Despite the reported decline in hacker attack complaints in 2022 compared to the previous year 2021, the results outline a significant increase that is forecast to continue in the future. At the same time, the financial losses of the victims are increasing annually (Fig. 1a), reaching a record value of 12.5 billion dollars in 2023 [3].



(**a**)



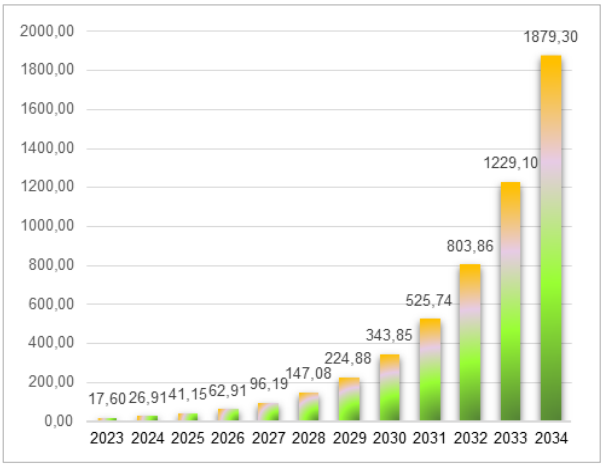
(**b**)

Figure 1. Cybercrime losses and investment: **(a)** Cybercrime losses reported in the last five years; **(b)** Cybercrime defense investment market trend, in trillions of dollars.

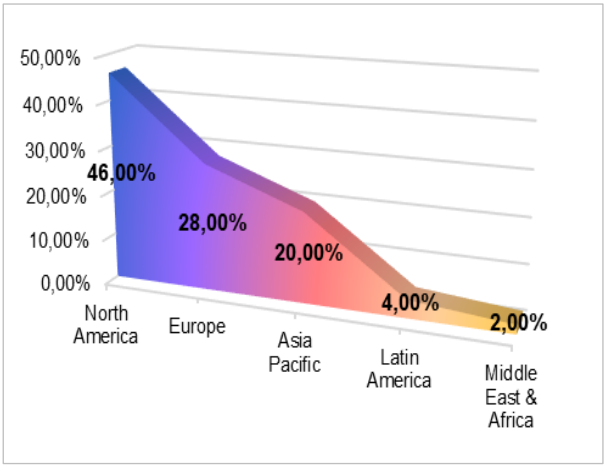
Global cybercrime defence spending is expected to reach $15.63 trillion by 2029 (Fig. 1b) [4]. As digital assets and connectivity increase, cyber threats are also evolving and becoming more sophisticated. Traditional cybersecurity methods, such as antivirus programs and firewalls, are often insufficient to deal with modern attacks. Ransomware attacks, identity theft, DDoS (Distributed Denial of Service) attacks, and attacks against critical infrastructures are just some of the challenges that require new and more effective protection methods [5, 6].

That is why organizations are increasingly turning to new approaches such as blockchain technology and artificial intelligence, which offer increased security and resilience against attacks [4].

Research shows that the blockchain technology market size is projected to grow at a steady steady pace through 2034 (Fig. 2a) In the past year 2023, the global blockchain technology market was valued at USD 17.60 billion. It is expected to reach about US$ 1,879.30 billion at a compound annual growth rate (CAGR) of 52.9% by 2034. [8].



(**a**)



(**b**)

Figure 2. Blockchain technology market and geographic regions distribution: **(a)** Blockchain technology market status and trend, in billion dollars; **(b)** Market share distribution for blockchain technologies by geographic regions, 2023.

The distribution of the blockchain technology market size by geographical regions shows (Fig. 2b) that North America has the largest share for 2023, with 46%, followed [8] by the European market, with 28%. The Asia Pacific region accounts for 20% of the market. The last places are Latin America, with a 4% share, and the Middle East and Africa, with a 2% share.

Blockchain technology, which is often associated with cryptocurrencies, has gained immense popularity recently. It is now widely applied outside of the cryptocurrency context, forming the basis for many applications, including those to improve cybersecurity. Research shows that blockchain can play a key role in protecting data and preventing fraud. A systematic review of the current research on blockchain technology can be found in the [9], which highlights the major challenges and opportunities in the field.

Smart contracts are self-executing programs that are stored and executed on the blockchain. They automate and guarantee the fulfilment of contractual terms without the need for intermediaries. This makes them suitable for building applications in cybersecurity, where they can be used to automate incident response, manage identities, and provide decentralized authentication.

Smart contracts have the potential to transform multiple industries by increasing efficiency and security [10]. This is further confirmed by the extremely diverse applications that are being developed based on blockchain technology. For example, blockchain-based multi-factor authentication provides a higher level of security and integration with various IoT devices [11]. Multi-factor authentication includes additional layers of security by combining traditional passwords with biometrics or other factors, making it more difficult to compromise. The authentication and authorization scheme in mobile cloud environments uses smart contracts to manage access and update user rights while minimizing computational and communication costs [12].

Another application of blockchain technology is systems like Dock.io's Web3 ID, which uses Decentralized Identities (DIDs) and Verifiable Credentials (VCs) to authenticate users without revealing personal data. Such systems improve the user experience by eliminating the need for multiple passwords and accounts. Platforms such as uPort and Sovrin offer stand-alone identity solutions where users securely manage and share their credentials [13]. This concept can be seen as the future of identity management in the digital age.

Blockchain-based solutions often use advanced cryptographic techniques, such as zero-knowledge proofs, which allow users to prove their claims without revealing actual data [10]. This increases the privacy and security of authentication. Crypto wallets such as MetaMask allow users to use their Ethereum accounts for authentication by signing nonce messages with their private keys [14]. This is a secure method of authentication without the need for passwords.

Secure authentication can also be achieved through smart contracts. In his paper [15], Buterin demonstrates how Ethereum smart contracts offer automated authentication and access management solutions using role-based access control (RBAC). This provides better security and reduces the risk of human error.

Blockchain technology finds application in digital identity management in various industries, such as healthcare, education, and logistics. For example, BurstIQ uses blockchain to manage healthcare data, ensuring its integrity and easy transfer between healthcare institutions. This approach ensures a high level of personal data protection and facilitates the exchange of information between the various participants in the healthcare ecosystem.

The use of non-fungible tokens (NFTs) for authentication is a relatively new concept, but it is rapidly gaining popularity due to the advantages it offers in terms of security and control over personal data. NFTs are unique digital assets that use blockchain technology to verify their authenticity and ownership. NFTs are used to verify ownership and access to exclusive digital resources and services [16].

In the literature [17-19], some authors propose different models that use NFTs as unique identifiers for users. A comprehensive review dedicated to NFT can be found in the article [20].

The literature review shows that blockchain offers a range of innovative authentication solutions that can improve the security and efficiency of various systems. By integrating these technologies, organizations can achieve better data protection and more secure identification processes. Among the main challenges are complexity and cost, especially for smaller organizations. This shortcoming applies to all the scientific publications cited above. The authors of this article sincerely believe that the root of these problems is the lack of understanding and, therefore, the limited use of blockchain technology, which is primarily associated with cryptocurrencies. Therefore, the main purpose of this article is to provide basic information about blockchain technology and smart contracts, as well as to demonstrate the step-by-step creation of simple authentication applications called MetaMask Authentication and Smart Contract-Based Authentication.

2. Blockchain Technology and Smart Contracts

Blockchain is a decentralized and distributed digital ledger that securely and transparently records transactions across multiple computers. A detailed overview of the architecture, consensus mechanisms, and future trends in blockchain technology is provided by Zheng et al. [21]. Each transaction is recorded in a block that is linked to the previous block by a cryptographic hash, forming a chain of blocks (blockchain) [22]. The key characteristics of blockchain are decentralization, transparency, and immutability. De-centralization refers to the fact that blockchain operates without a central authority, with data stored and managed by a network of participants (nodes), eliminating single points of failure and increasing security [23]. Each network participant has access to the entire transaction ledger, which provides a high degree of transparency [24]. Once recorded, the data in the blockchain cannot be changed or deleted without compromising the integrity of all subsequent blocks, ensuring their security and reliability [25].

There are several types of blockchain networks – public, private, and consortium.

- Public blockchain is open to anyone who wants to join and participate in the network, like Bitcoin and Ethereum [23];
- A private blockchain is controlled by a single organization that determines who can access and participate in the network and is often used in corporate environments [24];
- Consortium blockchain is a hybrid model where management is performed by a group of organizations, providing partial decentralization and increased security [25].

Blockchain applications are software solutions that use decentralized and secure blockchain technology to manage, store, and exchange data. These applications take advantage of blockchain features, including transparency, immutability, decentralization, and high security, to provide innovative solutions across industries and fields. Currently, blockchain applications have been developed in various fields of human life, such as voting and e-governance (Voatz, Follow My Vote), healthcare (Medicalchain, BurstIQ), food industry in supply chain management (VeChain, IBM Food Trust), digital identity management (Civic, uPort, Sovrin), real estate sales (Propy, RealT), entertainment and media by providing copyright management and direct payments to content creators (Audius, Mycelia), transaction finance with cryptocurrency, international transfers, and asset management (Bitcoin, Ethereum, Ripple), etc.

3. The Process of Creating Blockchain Applications

Regardless of the area in which the blockchain application will be used, its creation goes through the stages shown in Fig. 3.

In the first stage of developing a blockchain application, the main needs and problems that the application must address are identified. This may include specific business needs, user requirements, or technical challenges. Market research is also conducted to determine the competitive landscape and identify existing solutions and their limitations. The target users of the application are identified and how their needs can be met through blockchain technology is assessed. It is essential to define success metrics that will be used to measure the performance of the application after its implementation.

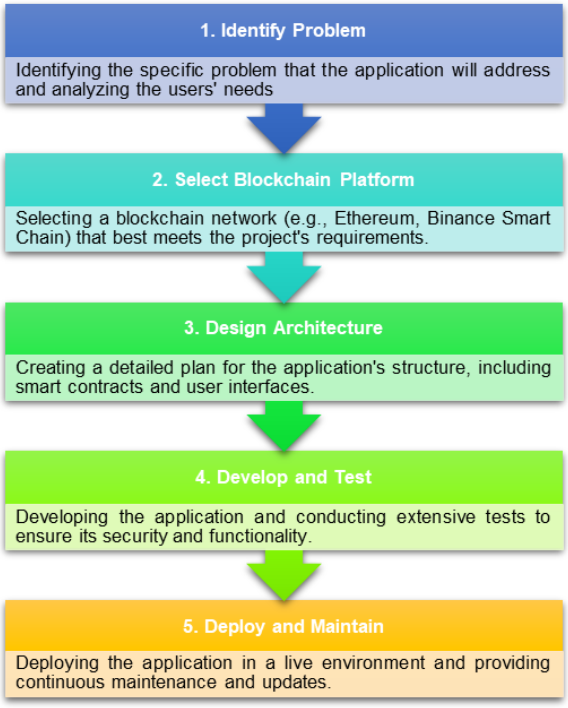


Figure 3. Stages of Blockchain Application Development.

The second stage focuses on choosing an appropriate blockchain platform. Choosing the right blockchain platform is a critical stage that can significantly affect the success of the project. This choice should be based on the specific requirements and goals of the application. Key factors to consider include performance (scalability, transaction speed, and network capacity), security, presence of an active developer community, and compatibility with required tools and protocols. Scalability refers to the platform's ability to handle an increasing volume of transactions. Transaction speed, which is the time it takes to process transactions, is a key factor for user-centric applications. Network capacity is the total

number of transactions the network can process per second (TPS). Security measures of the blockchain platform refer to its protection against vulnerabilities and attacks and the type of consensus algorithms (Proof of Work, Proof of Stake, etc.) that affect the level of security and efficiency. The requirement for an active developer community ensures that there are more resources, tools and libraries, documentation, and support materials available to developers that can make the development process easier and faster. Compatibility is primarily related to the platform's ability to integrate with existing systems and technologies, as well as the required protocols and standards for the application.

In the second stage, a blockchain platform is selected. Some of the most popular blockchain platforms are:

- Ethereum is a public, open platform that allows developers to create and execute smart contracts without the involvement of third parties. It is known for its ability to support smart contracts and decentralized applications (dApps). Ethereum has the largest developer community and broad support, which means more resources and educational materials are available (Ethereum White Paper). The platform offers numerous development tools and libraries that can speed up the development process (Cointelegraph). However, Ethereum requires payments to make transactions and execute smart contracts through fees known as "gas". These fees are paid in ether (ETH), the platform's cryptocurrency. Gas is needed to offset the computing resources used by the network to validate and execute transactions and smart contracts;
- Hyperledger Fabric is a private blockchain project supported by the Linux Foundation designed for enterprise solutions. It offers a modular architecture that allows various components, such as consensus and membership, to be plug-in systems. It is suitable for applications requiring high confidentiality and data control;
- EOS.IO is a blockchain platform that focuses on performance and scalability. It uses a Delegated Proof of Stake (DPoS) consensus algorithm that enables high transaction speeds. This platform can process thousands of transactions per second and is suitable for large decentralized applications;
- Binance Smart Chain (BSC) is a blockchain platform created by the Binance crypto-currency exchange. It is compatible with Ethereum, allowing easy migration of dApps from Ethereum to BSC. It supports smart contracts and decentralized applications and offers lower transaction fees compared to Ethereum;
- Cardano is a blockchain platform developed with a scientific approach and partnerships with academic institutions, focusing on research and evidence. It uses the Ouroboros consensus algorithm, which is a proof-of-stake (PoS) system. The platform was built with an emphasis on security and scalability
- Polkadot is a blockchain platform that allows different blockchain chains to interact and exchange information with each other. It uses a system of parallel chains called parachains. It supports multiple parachains, which improves performance and scalability.

When developing blockchain applications (e.g. on the Ethereum platform), costs can be reduced by using testnets. They allow free development and testing of smart contracts using test ether that has no real value. Test ether can be obtained from so-called faucets that distribute free test coins. Rinkeby, Ropsten, Kovan, and Goerli are some of the better-known test networks.

It is also possible to use local blockchain or layer 2 solutions, which are important components for developing and optimizing blockchain applications. They provide opportunities to improve performance, reduce costs, and test applications in a more controlled environment. Some popular native blockchains include Ganache, which allows developers to simulate a blockchain network on their computer and is suitable for testing and developing Ethereum smart contracts; Hardhat, which is an Ethereum development environment that includes a built-in local blockchain for testing and is suitable for testing complex smart contracts and integrations in decentralized applications; Hyperledger Fabric Local Network, which is a local blockchain network for enterprise solutions; and Parity Substrate, a modular framework for building custom blockchains. Among the better-known Tier 2 solutions are Polygon, which provides scalable and low-cost transactions for Ethereum and is suitable for applications requiring high transaction throughputs, such as games and decentralized finance (DeFi) platforms; Arbitrum, which is used in DeFi platforms and applications requiring fast data processing, as it allows the use of optimistic aggregations to improve scalability; and Optimism, which is suitable for applications that want to reduce transaction fees and improve the user experience.

The third stage of developing a blockchain application is dedicated to designing its architecture. This includes defining the structure of the application, including the front and back ends, the user interface design, and determining how the blockchain

components will interact with other parts of the system. It is also necessary to define smart contracts, i.e., to develop their logic and determine what functions they will perform and how they will interact with the blockchain.

During the fourth stage, application components are created and integrated using appropriate technologies and programming languages. Functional testing is conducted followed by stress testing to ensure that the system can handle workloads, performance analysis, and user testing. The most commonly used programming languages for the development of blockchain applications are listed in Table 1.

Table 1. The most commonly used programming languages for the development of blockchain applications.

Programming language	Description
Solidity	Used for programming when creating blockchain applications and writing smart contracts in Ethereum. It has a JavaScript-like syntax and is specifically de-signed to run on the Ethereum Virtual Machine (EVM). It supports inheritance, libraries, and user-defined complex types, and has a large developer community
Vyper	Alternative to Solidity. It has a stricter and simpler syntax compared to Solidity, which makes it more secure and reduces the risk of vulnerabilities. It also has fewer features but is easier to audit
Rust	Systems programming language known for its speed and security. It is widely used to develop smart contracts on platforms such as Solana and Polkadot
Golang (Go)	Golang is a powerful and efficient language used to write smart contracts and modules in Hyperledger Fabric
JavaScript/ TypeScript	Widely used to create a front-end for decentralized applications. Their main advantages include a huge ecosystem of libraries and tools for building user interfaces, as well as easy integration with blockchain through libraries such as Web3.js and Ethers.js
Python	Multipurpose language used for automation, data analysis and blockchain prototyping. It is easy to learn and use, with powerful support for blockchain integration libraries. Its main applications are related to blockchain application prototyping, blockchain data analysis and automation

When developing blockchain applications, some of the most commonly used tools and libraries are Truffle Suite, Hardhat, Web3.js, Ethers.js, and Brownie (Table 2).

Table 2. The most commonly used tools and libraries for the development of blockchain applications.

Tool/ library	Description
Truffle Suite	Complete framework for developing Ethereum applications, including tools for compiling, deploying, and testing smart contracts. Its main components are the Ganache platform and Drizzle - a state management library for dApps
Hardhat	Hardhat, as mentioned earlier, is a professional Ethereum development environment offering powerful tools for debugging and testing smart contracts
Web3.js	JavaScript library that enables interaction with the Ethereum blockchain from front-end applications. It is used to perform transactions, read data from the blockchain, and interact with smart contracts, allowing easy integration with dApps
Ethers.js	Lightweight library for interacting with Ethereum, known for its simplicity and powerful features. It provides tools for creating wallets, managing transactions, and interacting with smart contracts
Brownie	Python-based framework for developing and testing Ethereum smart contracts. Supports automated testing and deployment management. It is used to develop and test Ethereum smart contracts using Python

In the final fifth stage, the completed blockchain application is deployed on the chosen blockchain platform with the necessary configurations. Application performance monitoring is done to promptly detect and resolve any issues. It is recommended to offer updates and improvements to the blockchain application, as well as provide support to users in case of questions or problems.

4. Blockchain Authentication

4.1. Crypto wallet-based authentication

Crypto wallets are digital tools that allow users to store, manage, and transact with cryptocurrencies. They provide an interface for accessing blockchain networks while ensuring the security of private keys necessary for signing transactions. Crypto wallets can be categorized into two main types: "hot wallets" and "cold wallets," depending on whether they are connected to the internet.

Hot wallets are connected to the internet, offering easier access and convenience for daily transactions. Examples of such wallets include MetaMask, Trust Wallet, and Coin-base Wallet. These wallets are popular due to their ease of use and integration with decentralized applications, particularly within the Ethereum ecosystem.

Cold wallets, on the other hand, are not connected to the internet and offer a higher level of security,

reducing the risk of hacking attacks. They are preferred for storing larger amounts of cryptocurrency over the long term. Popular examples of cold wallets include Ledger Nano S and Trezor. These hardware devices store private keys offline, providing an additional layer of protection against cyber-attacks.

The use of crypto wallets is essential for securing cryptocurrencies and managing digital assets. According to research by Satoshi Nakamoto, the creator of Bitcoin, crypto wallets play a crucial role in decentralized security by providing control over private keys and eliminating the need for trust in third parties [22].

For the created example, the MetaMask crypto wallet was selected due to its wide-spread use and ease of integration with dApps.

MetaMaks is a tool for interacting with decentralized applications on blockchain networks such as Ethereum. It provides a simple interface for managing Ethereum ac-counts, sending and receiving cryptocurrencies, and signing transactions and messages directly from the browser. MetaMask operates as a browser extension, allowing users to interact with the blockchain without the need to install full nodes or specialized software. A full node refers to a computer that fully validates transactions and blocks in the block-chain. Currently, MetaMask is compatible with some of the most popular browsers, including Google Chrome, Mozilla Firefox, Brave, Microsoft Edge, and others. It should be noted that automatic installation of MetaMask as a browser extension is not possible due to security and privacy concerns. Browsers do not allow websites or scripts to install ex-tensions automatically without the user's consent.

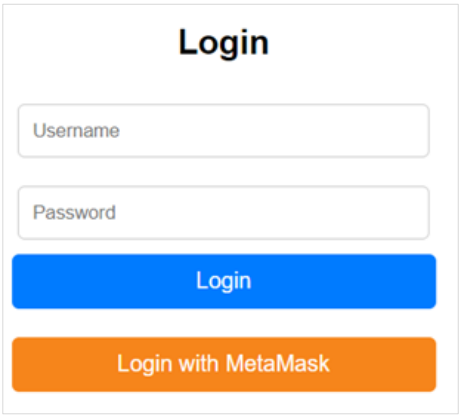
Once MetaMask is installed, it needs to be configured. This involves creating a new account or importing an existing one. Creating a new account requires setting a strong password, as well as a 12-word secret phrase that serves as a backup for account recovery. In case of loss of access to MetaMask or your computer, the wallet can be restored using this phrase. After the secret phrase is confirmed, the setup process is complete, and the wallet becomes ready for use. This includes receiving and sending cryptocurrencies, signing transactions, and interacting with decentralized applications.

To demonstrate the integration process, a standard login form was created, similar to those used by most websites or web-based information systems. This example form is depicted in Fig. 4a.

The HTML code behind the login form is shown in Fig. 4b. The JavaScript function checks if MetaMask is installed and requests permission to access the user's MetaMask account, which can then be used to log in to the system.

Clicking on the "Login" and "Login with MetaMask" buttons initiates two distinct processes, as they use different methods for user authentication.

When clicking the "Login" button, the user enters their username and password into the corresponding fields and submits the form to the server for processing. Basic client-side validations, such as checking for empty fields or minimum password length, may be performed. Once the form passes all client-side validations, the data (username and password) is sent to the server via an HTTP POST request.



(**a**)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <title>MetaMask Authentication - Primer 1</title>
</head>
<body>
    <div class="container">
        <h2>Login</h2>
        <form>
            <input type="text" placeholder="Username" required><br>
            <input type="password" placeholder="Password" required><br>
            <input type="submit" value="Login">
        </form>
        <div class="metamask-btn" onclick="connectMetaMask()">Login wit
MetaMask</div>
    </div>

    <script>
        function connectMetaMask() {
            if (typeof window.ethereum !== 'undefined') {
                console.log('MetaMask is installed!');
                window.ethereum.request({ method: 'eth_requestAccounts'
                .then(accounts => {
                    console.log('Connected account:', accounts[0]);
                    alert('Connected account: ' + accounts[0]);
                })
                .catch(error => {
                    console.error('Error connecting to MetaMask:', erro
                    alert('Error connecting to MetaMask');
                });
            } else {
                alert('MetaMask is not installed. Please install MetaMa
and try again.');
            }
        }
    </script>
</body>
</html>
```

(**b**)

Figure 4. Standard login form and basic HTML code for a MetaMask integration and authentication options: **(a)** Standard login form with MetaMask integration; **(b)** Basic HTML code for a login form with standard and MetaMask authentication options.

The server first checks whether the username exists in the database. If the username exists, the server then verifies whether the entered password matches the hashed pass-word stored in the database. If both the username and password are valid, the server creates a session for the user and redirects them to the secure area of the website.

The user is provided with a session identifier (session ID), which is used for identification in subsequent requests to the server. If the username and password are correct, the user will be able to log into the system and gain access to protected resources. If an error occurs (e.g., incorrect password or non-existent username), the user will see an error message.

When clicking the "Login with MetaMask" button, a JavaScript function is triggered, which checks whether MetaMask is installed and available in the browser. The JavaScript code verifies whether window. ethereum is defined in the browser, indicating the presence of MetaMask. If MetaMask is not available, the user is prompted to install MetaMask. If MetaMask is available, the browser sends a request for account access using Ethereum. request({ method: 'eth_requestAccounts' }). MetaMask then displays a pop-up window where the user must confirm or deny access to their Ethereum account. If the user grants access, MetaMask returns a list of Ethereum account addresses associated with the application. Typically, the first account in the list is the user's primary address, which will be used for authentication. The application may send a message to the user for signing with their private key stored in MetaMask. This signature serves as proof of ownership of the account. The signed message is sent back to the server, where it is verified whether the signature corresponds to the account address (this step of the process is optional).

To validate the signature generated by the user via MetaMask, the server must utilize appropriate cryptographic libraries that allow verification of the signature's authenticity against the sent message and the user's public key. The validation process involves sever-al key steps.

First, the server receives the signature, the user's address, and the original message that was signed.

Second, then, using a cryptographic library such as Ethereum-util in Node.js, the server extracts the public key from the signature. With this key, the server can verify whether the signature was created by the private key associated with the address provided by the user. If the signature is valid, the server can confidently assume that the user who presented the signature indeed owns the corresponding Ethereum account.

Third, the server then creates a session for the user and redirects them to the secure area of the website. If the authentication process is successful, the user will be logged into the system without the need to enter a username and password, providing a more secure method of authentication. If there is an error during the connection process or if the user denies access to MetaMask, the process is terminated, and the user remains unauthenticated.

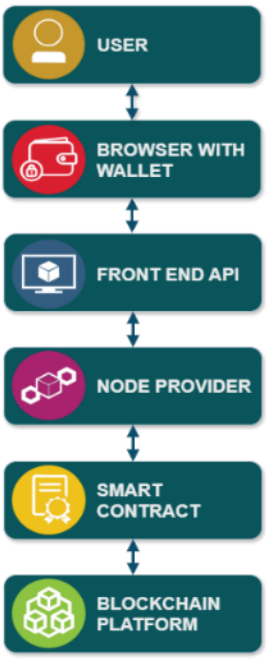
During the authentication process via MetaMask, various errors may occur, which must be properly handled to ensure the system's reliability and a good user experience. One of the most common errors is the user's refusal to grant access to their cryptographic key or to sign the message. In this case, the browser or the MetaMask extension itself generates an exception that must be caught by the application. Upon catching such an exception, the server or the client-side application may provide an appropriate error message to the user, explaining why the authentication cannot proceed. For instance, a message such as "Authentication was denied. Please try again and provide the necessary permissions." may inform the user of the actions they need to take. Additionally, the system should be prepared to handle other potential errors, such as network issues, browser incompatibility, or failures in sending data to the server, with all these situations being addressed through appropriate recovery mechanisms or error messages.

4.2. Smart Contract-Based Authentication

Reliable authentication can also be implemented using smart contracts. These are automated, self-executing programs that exist within a blockchain network and are triggered when predefined conditions are met. They are designed to facilitate, execute, and ensure trust in contractual agreements without the need for intermediaries. Authentication based on smart contracts provides a reliable and secure method for managing user identities and access control in a decentralized environment. These contracts contain the logic necessary for authenticating entities, ensuring that only authorized users have access to specific resources.

For this paper, a Smart Contract-Based Authentication application was developed, and its working diagram is presented in Fig. 5a. The authentication process is initiated by the user, who connects to a web application containing a login form. This application may be part of a decentralized application or an information system using blockchain technology for managing user identities. Simultaneously, the user must have a crypto wallet. For this specific application, MetaMask, a popular browser extension, was used. When attempting to log in, the web application establishes a connection with the crypto wallet through a library such as Web3.js or Ethers.js (in this case, Web3.js was used). The crypto wallet manages the user's cryptographic keys and

provides secure access to their crypto accounts. The user enters their password to unlock access to their account.



(**a**)

```
CONTRACT Authentication:

// Declare a variable to store registered users
DECLARE mapping(address => bool) registeredUsers

// Function to register a user
FUNCTION registration():
    IF registeredUsers[CALLER_ADDRESS] == false THEN:
        // Mark the current address as registered
        registeredUsers[CALLER_ADDRESS] = true
        RETURN "Registration successful"
    ELSE:
        RETURN "User already registered"

// Function to check user authentication
FUNCTION isAuthenticated(userAddress):
    // Returns whether the given address is registered
    RETURN registeredUsers[userAddress]

// Constructor (if necessary)
FUNCTION constructor():
    // Any necessary initial settings for the contract
    INITIALIZE any necessary variables or states

END CONTRACT
```

(**b**)

Figure 5. Smart Contract for Decentralized Applications working diagram and pseudocode: **(a)** Workflow of Smart Contract-Based Authentication; **(b)** Pseudocode of the Authentication Smart Contract for Decentralized Applications.

During the authentication process via MetaMask, various errors may occur, which must be properly handled to ensure the system's reliability and a good user experience. One of the most common errors is the user's refusal to grant access to their cryptographic key or to sign the message. In this case, the browser or the MetaMask extension itself generates an exception that must be caught by the application. Upon catching such an exception, the server or the client-side application may provide an appropriate error message to the user, explaining why the authentication cannot proceed. For instance, a message such as "Authentication was denied. Please try again and provide the necessary permissions." may inform the user of the actions they need to take. Additionally, the system should be prepared to handle other potential errors, such as network issues, browser incompatibility, or failures in sending data to the server, with all these situations being addressed through appropriate recovery mechanisms or error messages.

Once the user is authenticated in the crypto wallet, the Web3.js library acts as an in-termediary between the web application and the blockchain network. It sends a request to the crypto wallet containing the necessary information for authentication. The crypto wallet receives the request and presents the user with details about the transaction or message that needs to be signed. The user can confirm or reject the signing request. If the user confirms, the crypto wallet uses the user's private key to sign the data, which serves as proof of their identity and authorization to perform the requested action.

The signed data is returned to the library, which then forwards it to the node provider. The node provider (for example Infura, Alchemy, QuickNode, etc.) provides access to the blockchain network and sends the signed transaction to the relevant blockchain node for processing. When the signed transaction reaches the blockchain network, it is processed and passed to the smart contract deployed on the blockchain. The smart contract verifies the signature to ensure its validity and that it was executed by an authorized user. If the signature is valid, the smart contract executes the prescribed actions, such as granting access to protected resources or completing the transaction.

After the smart contract completes the requested action, the result is returned along the same path—from the blockchain network, through the node provider, to the Web3.js or Ethers.js library, and finally to the web application in the browser. The web application informs the user of successful authentication and grants access to the corresponding resources or services. This process demonstrates the interaction between the user, the crypto wallet, the blockchain interaction library, the node provider, and the smart contract, all of which play a key role in ensuring the security and efficiency of the system.

One of the critical stages is the development and deployment of the smart contract—the most crucial part of this type of cybersecurity application. The contracts are programmed, most commonly in Solidity. For this example, a smart contract in Solidity was also developed, and its pseudocode is presented

below (Fig. 5b). The smart contract for authentication functions as a basic system for registering and authenticating users within a blockchain environment. Its primary task is to provide a secure method for access management using decentralized infrastructure.

When executed, users have the option to register their Ethereum addresses directly on the blockchain via the registration() function. This function checks whether a given address is already registered. If the address is not found in the registry, it is added, and the user receives a confirmation of successful registration. In cases where the address already exists in the system, the function informs the user that they are already registered.

In addition to registration, the smart contract offers an isAuthenticated() function for verifying user authenticity. This function allows the application to check whether a given address is registered, returning a result that verifies the user's status. This way, the system can determine whether a user has the right to access certain resources or services within the decentralized application.

The smart contract may also contain a constructor() function, which is optional. It is executed only once—at the moment the contract is created—and cannot be called again thereafter. It is most commonly used for initializing variables but can also be used to set the address of the owner or administrator of the contract, define certain trusted addresses such as those of counterparties or other smart contracts with which the current contract will interact, automatically distribute a certain number of tokens to different addresses upon contract deployment, and more. If there are no specific tasks that need to be performed during contract deployment, the constructor() function can be omitted.

Lastly, while some node providers offer paid services, Ganache is used as a development tool in this study. Ganache allows developers to create a local blockchain network for developing and testing smart contracts and decentralized applications.

4.3. Comparative Analysis of Authentication Security

The comparison of authentication security between crypto wallets, particularly MetaMask, and smart contracts reveals different aspects of protection that are crucial for the reliability of blockchain-based systems. MetaMask, as a crypto wallet and browser extension, provides users with a convenient and secure way to manage their cryptographic keys and sign transactions directly from the browser. A key aspect of MetaMask's security is the local storage of keys on the user's device, which reduces the risk of centralized attacks. The user's involvement in approving each transaction adds a layer of protection, as every action requires explicit user consent. However, the security of Meta-Mask is directly linked to the security of the device on which it is installed, which may be vulnerable to physical or software attacks.

On the other hand, smart contracts offer a significantly different approach to authentication security. Smart contracts are automated, self-executing programs that operate within blockchain networks and provide decentralized management of identities and access. One of the primary advantages of smart contracts is their immutability after deployment, ensuring that once the rules and logic are recorded, they cannot be altered or manipulated. The cryptographic security of smart contracts provides reliable protection against data forgery, using proven algorithms for verifying digital signatures and identities. However, smart contracts are not immune to vulnerabilities; they can be exposed to specific risks such as re-entrancy attacks and logical errors, which can be exploited if the contract is not properly tested and audited.

In terms of authentication reliability, MetaMask offers a high level of security based on the local storage of keys and the need for user participation. This makes it a suitable solution for scenarios where user flexibility and control are necessary. However, Meta-Mask relies on the security of the device, which can be a vulnerability in certain contexts. Smart contracts, on the other hand, provide strong protection through decentralization and automation, making them highly suitable for systems requiring immutability and minimization of human involvement in the authentication process. Nevertheless, the security of smart contracts is directly tied to the quality of their code and development practices. Errors in the code can lead to serious vulnerabilities, requiring careful testing and auditing before deployment.

It can be said that the MetaMask wallet is more suitable for situations requiring active user involvement and flexibility, while smart contracts offer greater security and automation in decentralized applications. It is important to note that the security of these technologies depends not only on their built-in mechanisms but also on their proper implementation and maintenance in a real-world environment.

The security of smart contracts is critical to the reliability of blockchain-based authentication systems. Although smart contracts offer powerful possibilities for automation and decentralized management, they are also exposed to specific vulnerabilities that can be exploited if not properly addressed. One of the most well-known vulnerabilities is the re-entrancy attack, in which a malicious contract can re-invoke the original contract's function before the initial transaction is completed. This can allow the attacker to bypass identity verification logic,

potentially leading to unauthorized access to resources or the execution of unauthorized actions. For example, if a contract includes an authentication function that checks whether a user is authorized and then performs an action such as granting access or completing a transaction, a re-entrancy attack could allow multiple invocations of this function before the initial operation is completed. This can disrupt the authentication process and compromise system security.

To protect smart contracts from such attacks, it is essential to follow best practices in development and deployment. An effective strategy is to ensure that the contract's state is updated before making external calls or sending funds. This means that the authentication check should be completed before performing any action that could trigger an external call. Using modifiers like nonReentrant can also help prevent re-invocation of the function until the current transaction is fully completed. This ensures that even if an attempt is made to re-invoke, the contract will not execute the function again and bypass security checks.

Limiting the authentication logic to a single transaction, without relying on interactions with other contracts, can also reduce the risk of re-entrancy attacks.

If possible, the authentication logic should be kept simple, avoiding any interaction with external contracts that could be exploited.

Re-entrancy attacks pose a threat to the security of smart contracts, but with proper design and the application of best practices, their risk can be significantly reduced. In the context of authentication, protection against these attacks is crucial to ensuring the safety and reliability of decentralized systems.

5. Conclusion

This paper examined the security of authentication in the context of blockchain technologies, comparing crypto wallets like MetaMask with smart contracts. The study demonstrated that both approaches offer significant advantages in terms of decentralization, cryptographic protection, and reducing reliance on centralized intermediaries. However, each has specific challenges and vulnerabilities that must be carefully considered when selecting the appropriate technology for a given application.

One of the main conclusions from the analysis is that the choice between MetaMask and smart contracts depends on the context and specific requirements of the system. MetaMask provides a high degree of flexibility and user control, which is particularly useful in cases where user involvement is crucial. On the other hand, smart contracts offer strong protection through immutability and automation, making them highly suitable for systems requiring reliable and decentralized authentication.

Future research in this area should focus on a deeper analysis of vulnerabilities associated with the use of smart contracts, as well as the development of new techniques to mitigate these risks. Specifically, attention should be given to the integration of multi-factor authentication and the application of artificial intelligence to enhance the security and adaptability of these systems. Additionally, exploring the possibilities for standardizing smart contracts and cryptographic protocols used for authentication will be important to facilitate their implementation in real-world applications.

In conclusion, the findings highlight the importance of blockchain technologies in the advancement of modern cybersecurity. They offer new opportunities for creating decentralized, secure, and reliable systems for identity and access management. However, the successful implementation of these technologies requires a thorough understanding of their advantages and limitations, as well as the application of best practices in development, testing, and security auditing. While this study represents only an initial step in the process of integrating blockchain into the field of cybersecurity, the results demonstrate the potential of this approach to significantly contribute to the development of secure and reliable identity and access management systems.

References

[1] Patel, K. (2013). Incremental journey for World Wide Web: Introduced with Web 1.0 to recent Web 5.0–A survey paper. *International Journal of Advanced Research in Computer Science and Software Engineering, 3*(10).

[2] FBI. (2024, December 26). FBI's IC3 report: Losses from cybercrime surpass $12.5 billion—a new record. Retrieved from https://www.proofpoint.com/us/blog/email-and-cloud-threats/fbis-ic3-report-losses-cybercrime-surpass-125-billion-new-record

[3] FBI. (2023). *FBI IC3 Internet crime report 2023*. Retrieved from https://www.ic3.gov/AnnualReport/Reports/2023_ic3report.pdf

[4] CNBC. (2024, September 7). *Estimated cost of cybercrime worldwide 2018-2029*. Retrieved from https://blog.biocomm.ai/2024/09/07/cnbc-estimated-cost-of-cybercrime-worldwide-2018-2029-in-trillion-u-s-dollars/

[5] eSentire. (2024, December 26). *The state of cybersecurity threats*. Retrieved from https://www.esentire.com/

[6] Conti, M., Dehghantanha, A., Franke, K., & Watson, S. (2018). Internet of Things security and forensics: Challenges and opportunities. *Future Generation*

Computer Systems, 78, 544-546. https://doi.org/10.1016/j.future.2017.07.060

[7] Crosby, M., Pattanayak, P., Verma, S., & Kalyanaraman, V. (2016). Blockchain technology: Beyond Bitcoin. *Applied Innovation, 2*(6-10), 71.

[8] Precedence Research. (2024). *Blockchain technology market size, share and trends 2024 to 2034*. Retrieved from https://www.precedenceresearch.com/blockchain-technology-market

[9] Yli-Huumo, J., Ko, D., Choi, S., Park, S., & Smolander, K. (2016). Where is current research on blockchain technology? A systematic review. *PLOS One, 11*(10), e0163477. https://doi.org/10.1371/journal.pone.0163477

[10] Szabo, N. (1997, September 1). Formalizing and securing relationships on public networks. *First Monday, 2*(9). https://doi.org/10.5210/fm.v2i9.548

[11] Mahmood, S., Chadhar, M., & Firmin, S. (2022). Cybersecurity challenges in blockchain technology: A scoping review. *Human Behavior and Emerging Technologies, 2022*(1), 7384000. https://doi.org/10.1155/2022/7384000

[12] Villarreal, E. R. D., García-Alonso, J., Moguel, E., & Alegría, J. A. H. (2023). Blockchain for healthcare management systems: A survey on interoperability and security. *IEEE Access, 11*, 5629-5652. https://doi.org/10.1109/ACCESS.2023.3236505

[13] Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the Internet of Things. *IEEE Access, 4*, 2292-2303. https://doi.org/10.1109/ACCESS.2016.2566339

[14] Almadani, M. S., Alotaibi, S., Alsobhi, H., Hussain, O. K., & Hussain, F. K. (2023). Blockchain-based multi-factor authentication: A systematic literature review. *Internet of Things, 100844*. https://doi.org/10.1016/j.iot.2023.100844

[15] Gimenez-Aguilar, M., De Fuentes, J. M., Gonzalez-Manzano, L., & Arroyo, D. (2021). Achieving cybersecurity in blockchain-based systems: A survey. *Future Generation Computer Systems, 124*, 91-118. https://doi.org/10.1016/j.future.2021.05.007

[16] Buterin, V. (2014). A next-generation smart contract and decentralized application platform. *White Paper, 3*(37), 2-1.

[17] Ekblaw, A., Azaria, A., Halamka, J. D., & Lippman, A. (2016, August). A case study for blockchain in healthcare: "MedRec" prototype for electronic health records and medical research data. In *Proceedings of IEEE Open & Big Data Conference* (Vol. 13, p. 13).

[18] Guo, Y., & Liang, C. (2016). Blockchain application and outlook in the banking industry. *Financial Innovation, 2*, 1-12. https://doi.org/10.1186/s40854-016-0034-9

[19] Huh, S., Cho, S., & Kim, S. (2017, February). Managing IoT devices using blockchain platform. In *2017 19th International Conference on Advanced Communication Technology (ICACT)* (pp. 464-467). IEEE. https://doi.org/10.23919/ICACT.2017.7890132

[20] MetaMask. (2024, December 26). *MetaMask developer documentation*. Retrieved from https://docs.metamask.io/

[21] Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE International Congress on Big Data (BigData Congress)* (pp. 557-564). IEEE. https://doi.org/10.1109/BigDataCongress.2017.85

[22] Nakamoto, S. (2008). A peer-to-peer electronic cash system. *Bitcoin*. Retrieved from https://bitcoin.org/bitcoin.pdf

[23] Swan, M. (2015). *Blockchain: Blueprint for a new economy*. O'Reilly Media, Inc.

[24] Comert, O. (2020). *Blockchain revolution: How the technology behind Bitcoin and other cryptocurrencies is changing the world*.

[25] Antonopoulos, A. M. (2014). *Mastering Bitcoin: Unlocking digital cryptocurrencies*. O'Reilly Media, Inc.