

Comparative Analysis of Public and Encrypted Genesis Blocks in Blockchain Systems

Krisztián Bálint

Obuda University, Keleti Károly Faculty of Business and Management, Budapest, Hungary,
balint.krisztian1@uni-obuda.hu

Abstract In the accelerating digital economy, business communication increasingly depends on the fast, secure, and reliable exchange of information between organizations. Traditional communication systems often struggle with limited transparency, fragmented data management, poor traceability, and dependence on centralized intermediaries. Blockchain technology offers a novel approach to these challenges by providing a decentralized, tamper-proof, and auditable communication infrastructure. Trust, transparency, and security are key factors in blockchains, as blockchain acceptance continues to divide people to this day. The aim of the research is therefore to create two blockchains with different properties that can convey trust to users. The first of these is a public blockchain, in which the data is accessible and readable by everyone, but cannot be modified without authorization. The second blockchain is a private blockchain that uses complex data encryption solutions to protect user data, and access to it is strongly restricted. The research compares the Genesis blocks of the two blockchains created due to their different specificities. It can be stated that both blockchains enable secure decentralized data storage, and it is advisable to select them according to usage needs to ensure effective business communication.

Keywords: Public blockchain, private blockchain, data encryption.

Introduction

A blockchain has undergone a huge evolution, drawing upon various disciplines including mathematics, cryptography, and computer science. Central to its operation is decentralization, which enables peer-to-peer transactions, coordination, and collaboration facilitated by features such as timestamps, distributed consensus, data encryption, and economic incentives within the distributed system. These features offer a compelling solution to the inefficiencies and security concerns associated with centralized data storage [1]. Blockchain offers an innovative approach to storing information, executing transactions, performing functions, and establishing trust in an open environment. Besides technological characteristics, user trust, cybersecurity awareness, transparency, and effective communication are

also essential factors influencing blockchain adoption and acceptance [2]. Although blockchain has received growing interest in both academia and industry in recent years, the security and privacy of blockchains continue to be at the centre of the debate when deploying blockchain in different applications [3]. To examine the technology inherent in blockchain, private and public Genesis blockchains are created, with the aim of securely storing data using both solutions.

The research is structured according to the following structure:

- Creating and examining the Genesis block of a public transactional blockchain,
- Creating and verifying the Genesis block of a private, encrypted blockchain.

1 Public transaction blockchain

The characteristic of the blockchain is that the network operates on top of a traditional computer network, such as the internet, and cannot function without it. This is because the blockchain protocol does not propose a new method of connecting different devices but uses the Internet Protocol (IP) for all the transactions between different nodes. Therefore, the nodes cannot communicate without a common network [4]. With a distributed ledger technology, blockchain allows information to be stored in shared databases and protects information from removal. Blockchain also operates with fully shared and unchangeable information in a decentralized system, with the same level of certainty [5]. This condition is also true for the public blockchain. The Genesis block is the starting point of the blockchain, to which all subsequent blocks are logically linked, and therefore forms the basis of the integrity of the system. Its characteristics:

- One of the most important features of the Genesis block is that it has no history, so its previous hash field does not point to a real block, but contains an initial fixed value,
- The index number is usually 0,
- It contains the timestamp. This shows when the blockchain was created,
- In the case of a transaction's blockchain, it may contain a starting transaction, but this field can also be left blank,
- PreviousHash is 0, as there is no previous block,
- Own hash value, which is a unique imprint of the block.

```
using System;  
using System.Collections.Generic;
```

```

using System.Security.Cryptography;
using System.Text;
class Transaction
{
    public string FromAddress { get; set; }
    public string ToAddress { get; set; }
    public decimal Amount { get; set; }

    public Transaction(string fromAddress, string toAddress, decimal amount)
    {
        FromAddress = fromAddress;
        ToAddress = toAddress;
        Amount = amount;
    }
}
class Block
{
    public int Index { get; set; }
    public DateTime Timestamp { get; set; }
    public List<Transaction> Transactions { get; set; }
    public string PreviousHash { get; set; }
    public string Hash { get; set; }
    public Block(int index, DateTime timestamp, List<Transaction> transactions, string previousHash)
    {
        Index = index;
        Timestamp = timestamp;
        Transactions = transactions;
        PreviousHash = previousHash;
        Hash = CalculateHash();
    }
    private string CalculateHash()
    {
        using (SHA256 sha256 = SHA256.Create())
        {
            string rawData = Index.ToString() +
                Timestamp.ToString("o") +
                PreviousHash;
            foreach (var tx in Transactions)
            {
                rawData += tx.FromAddress + tx.ToAddress + tx.Amount.ToString();
            }
            byte[] bytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(rawData));
            StringBuilder builder = new StringBuilder();
            foreach (byte b in bytes)

```

```

        builder.Append(b.ToString("x2"));
        return builder.ToString();
    }
}
}
class Blockchain
{
    private List<Block> chain = new List<Block>();
    public Blockchain()
    {
        chain.Add(CreateGenesisBlock());
    }
    private Block CreateGenesisBlock()
    {
        var genesisTransactions = new List<Transaction>
        {
            new Transaction("0", "NyilvanosHalozat", 0)
        };
        return new Block(
            0,
            DateTime.UtcNow,
            genesisTransactions,
            "0"
        );
    }
}
}

```

Figure 1.
Public blockchain Genesis block code

2 Private encrypted blockchain

The private blockchain is only accessible to registered members with permission, offering instant, shared, and fully transparent information [6]. Before creating the Genesis block of the private blockchain, it is necessary to determine what security solutions we want to apply and what we want to protect the blockchain from. The first step is to consider the possibility of encryption, whether its use is necessary. The purpose of incorporating encryption into the blockchain is to prevent unauthorised reading of your data. In the case of the Genesis block, it is advisable to provide the following data:

- Organisational data belonging to the founder of the blockchain,
- Configuration data,

- Defining initial permissions and
- Private metadata.

Encryption itself does not prove that the data is authentic; it only hides it from unauthorised persons. Security can be increased by adding a digital signature. A digital signature answers the following questions:

- Who created the Genesis block?
- Has the content of the block not changed?
- Does the block come from a credible source?

Blockchains are characterized using hash algorithms. However, in the case of key-based integrity protection, it is advisable to use HMAC-SHA256 (Hash-based Message Authentication Code). This is a method where a plain hash function is used together with a secret key. Its purpose is not only to detect data modification, but also to ensure that only those who know the secret key can create a valid verification value. If the blockchain uses only the SHA-256 hash algorithm, the attacker can overwrite sensitive data and then create a new hash for this new content, so the system may not notice the fraud. In the case of HMAC, the attacker would also need to know the secret key to generate a new, valid HMAC. If they don't know this, the modified data can be easily discovered. In the case of integrating HMAC into a blockchain, network participants use a shared secret key.

Passwords can be used to gain access to specific data, an account, a computer system or a protected space [7], [8]. Weak password protection and insufficient security awareness significantly increase the risk of unauthorized access to sensitive information [8]. There are two different kinds of possibilities to get other users' passwords: stealing and guessing [9]. In the past few years, several security incidents have become known that were somehow related to passwords [10]. For additional data protection of the private blockchain, it is also recommended to use a salt, which makes it difficult to guess the password. A salt is a randomly generated additional data that is usually used in key derivation. This avoids the problem of two people entering the same password, as without a salt, the same key would be created. The danger of this is that it makes it easier for an attacker to recognize identical passwords. After that, it would presumably use pre-computed tables, which would make password cracking easier.

```

using System;
using System.Security.Cryptography;
using System.Text;
class GenesisBlock
{
    private int index;
    private DateTime timestamp;
    private string blockchainName;
    private string creator;
    private string networkType;
    private string version;
    private string encryptedData;
    private string previousHash;
    private long nonce;
    private string hash;
    public GenesisBlock(
        int index,
        DateTime timestamp,
        string blockchainName,
        string creator,
        string networkType,
        string version,
        string encryptedData,
        string previousHash,
        long nonce)
    {
        this.index = index;
        this.timestamp = timestamp;
        this.blockchainName = blockchainName;
        this.creator = creator;
        this.networkType = networkType;
        this.version = version;
        this.encryptedData = encryptedData;
        this.previousHash = previousHash;
        this.nonce = nonce;
        this.hash = CalculateHash();
    }
    private string CalculateHash()
    {
        using (SHA256 sha256 = SHA256.Create())
        {
            string rawData =
                index.ToString() +
                timestamp.ToString("o") +

```

```

        blockchainName +
        creator +
        networkType +
        version +
        encryptedData +
        previousHash +
        nonce.ToString();
    byte[] bytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(rawData));
    StringBuilder builder = new StringBuilder();
    foreach (byte b in bytes)
    {
        builder.Append(b.ToString("x2"));
    }
    return builder.ToString();
}
}
public void PrintBlock()
{
    Console.WriteLine("=== Extended Private Genesis Block ===");
    Console.WriteLine("Index: " + index);
    Console.WriteLine("Timestamp: " + timestamp.ToString("u"));
    Console.WriteLine("Blockchain Name: " + blockchainName);
    Console.WriteLine("Creator: " + creator);
    Console.WriteLine("Network Type: " + networkType);
    Console.WriteLine("Version: " + version);
    Console.WriteLine("Encrypted Metadata: " + encryptedData);
    Console.WriteLine("Previous Hash: " + previousHash);
    Console.WriteLine("Nonce: " + nonce);
    Console.WriteLine("Hash: " + hash);
}
}
class AesEncryptionHelper
{
    public static string Encrypt(string plainText, string password)
    {
        byte[] salt = GenerateRandomBytes(16);
        using (var deriveBytes = new Rfc2898DeriveBytes(
            password,
            salt,
            100000,
            HashAlgorithmName.SHA256))
        {
            byte[] key = deriveBytes.GetBytes(32);
            using (Aes aes = Aes.Create())

```

```

    {
        aes.Key = key;
        aes.GenerateIV();
        aes.Mode = CipherMode.CBC;
        aes.Padding = PaddingMode.PKCS7;
        using (ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV))
        {
            byte[] plainBytes = Encoding.UTF8.GetBytes(plainText);
            byte[] cipherBytes = encryptor.TransformFinalBlock(plainBytes, 0, plainBytes.Length);
            string saltBase64 = Convert.ToBase64String(salt);
            string ivBase64 = Convert.ToBase64String(aes.IV);
            string cipherBase64 = Convert.ToBase64String(cipherBytes);
            return saltBase64 + ":" + ivBase64 + ":" + cipherBase64;
        }
    }
}

private static byte[] GenerateRandomBytes(int length)
{
    byte[] bytes = new byte[length];
    RandomNumberGenerator.Fill(bytes);
    return bytes;
}
}

class PrivateBlockchain
{
    private GenesisBlock genesisBlock;
    public PrivateBlockchain()
    {
        genesisBlock = CreateGenesisBlock();
    }
    private GenesisBlock CreateGenesisBlock()
    {
        string sensitiveMetadata =
            "Validator1;Validator2;Validator3;Consensus=PoA;Permissioned=True";
        string password = "StrongPassword123!";
        string encryptedMetadata = AesEncryptionHelper.Encrypt(sensitiveMetadata, password);
        return new GenesisBlock(
            0,
            DateTime.UtcNow,
            "KGK Private Blockchain",
            "Obuda University KGK",
            "Private Permissioned Blockchain",
            "1.0",

```

```

        encryptedMetadata,
        "0",
        1001
    );
}
public void ShowGenesisBlock()
{
    genesisBlock.PrintBlock();
}
}
class Program
{
    static void Main(string[] args)
    {
        PrivateBlockchain blockchain = new PrivateBlockchain();
        blockchain.ShowGenesisBlock();
    }
}

```

Figure 2.
Private blockchain Genesis block code

3 Comparison of private and public blockchain Genesis blocks

The starting point of the operation of blockchain systems is the Genesis block, which, as the very first element of the chain, not only performs a technical initialization function, but also determines the integrity and trust basis of the entire system. In accordance with the research goal, two blockchains with different properties were developed, such as:

- A public, transaction-oriented blockchain, and
- A private, encrypted data management blockchain.

The primary role of the created public blockchain is security, openness, and transparency, which greatly support the transaction process for both the recipient and the sender.

The Genesis block of a private blockchain, on the other hand, is a much more complex structure. The blockchain is only accessible to authorized members, so special attention was paid to the security mechanisms built into the Genesis block. When creating a blockchain, it is advisable to record organizational data, configuration information, initial permissions, and private metadata in the Genesis

block. This indicates that the private Genesis block not only documents the start of the chain but also includes the initial state of the system's operating rules and security framework.

The private Genesis block is not based on a transaction list, but contains fields such as:

- blockchainName,
- creator,
- networkType,
- version,
- encryptedData,
- previousHash,
- nonce, and
- hash.

This Genesis block structure highlights that the Genesis block carries institutional and access control features even in the initial phase.

One of the most important differences between the two genesis blocks can be seen in the visibility and protection of data, since in the public blockchain, the data in the blocks is visible to everyone, while in the case of the private blockchain, access is highly restricted. The difference is also visible on a technical level. In the private Genesis block, the sensitiveMetadata content is stored using AES-based encryption, while the key is generated using PBKDF2-based derivation using SHA-256 and 100,000 iterations. During the encryption, a salt and an IV are also generated, which increases the security of the key derivation and symmetric encryption. Based on this, it can be concluded that the private Genesis block does not simply contain hidden data but implements a multi-layered protection approach that ensures data confidentiality at the boot level.

Public blockchains are characterized by complete decentralization, whereas private blockchains necessarily require some degree of access control and organizational management. The public genesis block, therefore, includes minimal organizational metadata.

In contrast, the private genesis block explicitly contains the institution name, network type, version, and also encrypted information referring to validators and PoA consensus, since in the private blockchain, the genesis block is not only a technical basic unit, but also a kind of regulated entry and operation framework.

Aspect	Public block	Genesis block	Differences between private and public blockchains
Basic goal	The chain is open, transaction-based data	The chain is protected, highly regulated	The public blockchain ensures transactional launch, the other ensures controlled launch
Data visibility	Visible to all participants	Only accessible to authorized actors	The public block is transparent, the private block contains encrypted metadata
Content structure	Index, timestamp, transactions, previous hash, hash	Index, timestamp, name, creator, network type, version, encrypted data, nonce, hash	The private block has an expanded security field structure.
Initial data type	Initiating transaction	Organizational and configuration metadata	The public block contains transaction. The private block contains encryptedData.
Data protection	It is not based on concealment, but on public verifiability.	Protects sensitive content with encryption	The private block uses AES + PBKDF2 + salt + IV.
Trust model	Trust based on transparency	Trust based on entitlement and protection	The public model is transparent; the private one is permission-only.

Table 1.
Comparison of private and public blockchain characteristics

Conclusions

Both public and private blockchains can store data appropriately and securely. Before using a blockchain, it is always recommended to examine your needs and select the blockchain that best suits your purpose. In the case of a public blockchain, the data contained therein is visible to everyone, while in the case of a private blockchain, it is not.

Before creating a private blockchain, it is advisable to examine the level of data protection required and, based on this, build the appropriate protection mechanisms

into both the entire blockchain and the Genesis block itself. When encrypting data, the best solution is to apply multiple encryption methods in parallel within the blockchain itself to increase security.

Public blockchains are characterized by complete decentralization, which is not entirely clear in the case of private blockchains. This problem stems from the structure of the blockchain itself, since with private blockchains, it is necessary to regulate who can access the blockchain and under what conditions they can use it.

In conclusion, proper communication is one of the most important factors when choosing a blockchain solution. If developers clearly communicate the characteristics, security mechanisms, and intended use of the blockchain, users can make more informed decisions and select the most appropriate solution for their needs. This observation is consistent with previous research emphasizing that user awareness, cybersecurity, transparency, and effective communication significantly influence the acceptance of digital technologies [2].

References

- [1] S. Dong, K. Abbas, M. Li & J. Kamruzzaman: Blockchain technology and application: an overview. *PeerJ Computer Science*, 9, e1705, pp 1-50, 2023.
- [2] Szikora, P., & Fehér-Polgár, P. (2025). Kiberbiztonság és felhasználói kihívások a digitális korban: párhuzamok a BYOD és az önvezető autók között. *MEB 2025 Konferencia kötet: Kiadványok III* (173–179. o.).
- [3] R. Zhang, R. Xue & L. Liu: Security and privacy on blockchain. *ACM Computing Surveys (CSUR)*, 52(3), pp. 1-34, 2019.
- [4] T. Ncube, N. Dlodlo & A. Terzoli: Private blockchain networks: a solution for data privacy. In *2020 2nd international multidisciplinary information technology and engineering conference, IMITEC*, pp. 1-8), 2020.
- [5] K. Bálint: Blockchain and Smart Contract Creation for Efficient and Secure Data Storage of Consumer Habits and Logistics Data. *Procedia Computer Science*, 253, pp. 49-58, 2025.
- [6] P. K. Paul, P. S. Aithal, R. Saavedra & S. Ghosh: Blockchain technology and its types—A short review. *International Journal of Applied Science and Engineering*, 9(2), pp.189-200, 2021.
- [7] K. Chanda: Password security: an analysis of password strengths and vulnerabilities. *International Journal of Computer Network and Information Security*, 8(7), pp. 23-30, 2016.
- [8] Ali, B., & Szikora, P. (2017). Információbiztonság az Y generáció körében. In *Tanulmánykötet – Vállalkozásfejlesztés a XXI. században VII* (pp. 24–40). A. Keszthelyi: About passwords. *Acta Polytechnica Hungarica*, Volume. 10, No. 6, pp. 99-118, 2013.

- [9] A. Keszthelyi: About passwords. Acta Polytechnica Hungarica, Volume. 10, No. 6, pp. 99-118, 2013.
- [10] A. Keszthelyi: Jelszavakról-iparági legrosszabb gyakorlatok= Passwords-worst practices in user authentication. Taylor, 7(3-4), pp. 261-268, 2015.