BLVchain

By
Yahya Parvin Aghdam
2023-2025

Website: https://BLVchain.org

Email: YahyaAghdam.ir@gmail.com

Abstract

With the acceleration of digital transformation, public authorities and companies are facing growing challenges in terms of data security, efficiency and transparency.
Traditional systems often suffer from bureaucratic inefficiency, security gaps and a lack of trust between stakeholders.
A private (enterprise) blockchain offers a secure, scalable and tamper-proof solution that is tailored to the specific needs of organizations institutions.

Benefits of using blockchain

1. Data security & integrity

Government agencies manage sensitive data, including citizen data, financial transactions and legal documents. A private blockchain ensures this:

- Immutable records : Once recorded, data cannot be altered or deleted, preventing fraud and corruption.
- Advanced encryption : Sensitive information is protected by cryptographic hashing.
- Controlled access : Only authorized entities (e.g. government agencies) can validate and modify data.

2. Transparency & accountability

One of the biggest challenges in government institutions is lack of transparency. A blockchain-based system ensures:

- Real-Time Audits : Every transaction is logged and can be reviewed anytime.
- Public Trust : Ensuring fair and accountable governance.
- Tamper-Proof Decision Making : Policies, budgets, and approvals are recorded transparently.

4. Inter-departmental integration

Government entities often struggle with fragmented and outdated IT systems. A private blockchain:

- Enables Seamless Data Sharing : Departments can securely exchange verified data.
- Reduces Redundancy : Eliminates duplicate record-keeping.
- Improves Communication : Connecting government agencies in real-time.

5. Cost reduction & resource optimization

A private blockchain reduces expenses related to:

- Paper-Based Systems : Eliminating physical documents and storage costs.
- Middlemen & Intermediaries : Direct, trustless transactions lower operational costs.
- Cybersecurity Risks : Preventing data breaches and fraud saves millions in losses.

6. Identity & access management

Governments must protect citizens' identities while ensuring seamless access to services. Blockchain enables:

- Decentralized Identity (DID) : Citizens control their own digital identity.
- Fraud Prevention : Preventing identity theft and unauthorized access.
- Instant Verification : Reducing delays in document authentication.

Use cases of private blockchain in government & enterprises

BLVchain introduces a miner-free, lightweight and powerful architecture specifically tailored for enterprise applications.

Utilizing WebSocket-based node communication and gRPC-powered integrations, BLVchain ensures seamless connectivity with existing enterprise systems at high throughput and low latency. The Delium method, a unique and proprietary approach, increases security, efficiency and decentralization, setting a new standard in blockchain technology.

BLVchain is designed with Go for performance and MongoDB for optimized data storage to handle large volumes of data transactions while remaining lightweight and cost-effective. Unlike traditional blockchains that rely on resource-intensive mining, our approach enables faster processing of data transactions, lower operational costs and enterprise-grade scalability.

This whitepaper describes the technical architecture, benefits and real-world applications of BLVchain and provides a roadmap for enterprises and developers looking to harness the power of decentralized technology without the inefficiencies of legacy systems.

Some of private blockchain usages:

- **E-Government services:** Digitized and secure citizen records.
- **Voting systems:** Tamper-proof and verifiable election processes.
- **Supply chain & procurement:** Transparent tracking of public funds and contracts.
- **Healthcare records:** Secure and interoperable patient data.
- **Land registry & ownership:** Preventing fraud in real estate transactions.
- **Banking systems:** Building trust and customer loyalty

Introduction

Limitations of traditional blockchains

Blockchain technology has revolutionized the industry by introducing decentralization, transparency and security. However, traditional blockchain networks often suffer from scalability issues and high energy consumption due to their reliance on mining-based consensus mechanisms. Companies looking to integrate blockchain technology need efficient, cost-effective and scalable solutions that fit seamlessly into their existing infrastructures.

A miner-free, lightweight blockchain for enterprises

Our blockchain was developed specifically for companies that need a secure, scalable and lightweight distributed ledger solution. By avoiding mining with a special algorithm, our network achieves fast data transactions, low operating costs and improved energy efficiency.

Easy deploy, Easy connection

With gRPC technology, BLVchain can connect to any programming language at super high speed and communicate with other servers. Installing and running a blockchain node is also very simple with a json configuration file and running a Docker container.

The Delium Method

Introduction

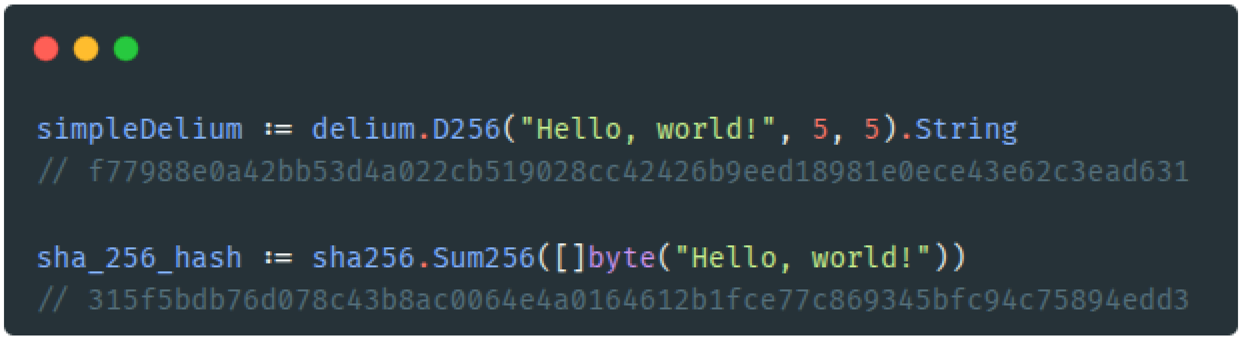
This method, inspired by the movie Oppenheimer, is derived from the concept of an atomic explosion. It works by imagining a string of characters encrypted with the SHA-256 or SHA-512 method as a certain number of atoms strung together. In an atomic explosion, almost all the atoms are destroyed, but our goal is to make the reverse engineering probability mathematically close to zero. Therefore, we remove several specific character sequences from the strings and then re-hack the resulting string using the SHA-256 or SHA-512 method. This cycle is repeated until we get a final hash value.
The ultimate goal of this method is for each blockchain organization to have its own specific hash method. Delium hashing method has two elements: 1- Delete steps 2- Repetition
So each final hash can change by changing these two elements, while the normal sha-256/512 method always has the same output with the same input.
Delium was developed specifically for BLVchain by Yahya Parvin Aghdam and is open source to use: https://github.com/blvchain/Delium

Delium is designed to turn a single input into a customized output based on configuration and create resistance to brute force attacks or quantum attacks. Also because of the deletion of miners, the blockchain must have a secure way to make the entire blockchain and data transactions and signatures secure and stable.
Two methods have therefore been developed: 1. Simple Delium 2.Complex Delium.
In the simple method, there is only the "Delete" and "Repeat" step, but in the complex method there is also the "Addon" string. In a cycle, the hashed string is added to another string and hashed again with delete step and repeat. The Complex method uses ‘path’ for this reason.
Therefore, the same input data can have an infinite output hash based on complex Delium.

Simple Delium

In the simple Delium we have only two elements: 1.delete steps 2.repeat. Let us take a look at how it works and what the difference is between sha-256 and D256
(We use D256 as an example, but D512 is also possible)

Delete step = 5
Repat = 5



```
simpleDelium := delium.D256("Hello, world!", 5, 5).String
// f77988e0a42bb53d4a022cb519028cc42426b9eed18981e0ece43e62c3ead631

sha_256_hash := sha256.Sum256([]byte("Hello, world!"))
// 315f5bdb76d078c43b8ac0064e4a0164612b1fce77c869345bfc94c75894edd3
```

What is happening in D256 method step by step:

1. Cycle 1: Hashing ***"Hello, world!"*** with sha-256 method
2. Cycle 1: Delete every 5th chars from the hash to use as new input data
3. Cycle 1: Hash with sha-256 the new data
4. Cycle 2: Delete every 5th chars from the hash to use as new input data
5. Cycle 2: Hash with sha-256 the new data

   .

   .

   .

11. Cycle 5: Hash with sha-256 the final data

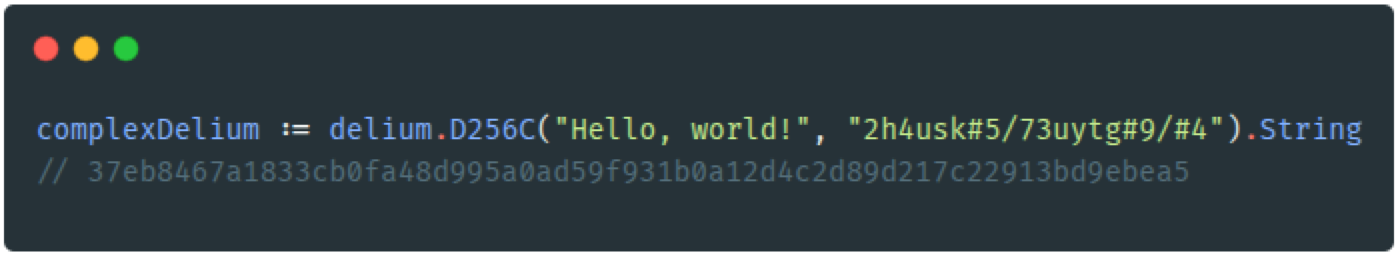
String output: *f77988e0a42bb53d4a022cb519028cc42426b9eed18981e0ece43e62c3ead631*

Complex Delium

This method was developed to simple Delium hashing along a specific path. So we have a path to simple Delium to hash the input data. We use D256C to get our complex Delium hash.

Part	***Explain***
/	Cycle identifier
#	Delete steps
<String>	Data will add in a cycle

Path example: ***"2h4usk#5/73uytg#9/#4"***
Explanation:
This path has 3 cycles/repeat based on slashes "/". The final cycle number is n+1
First cycle: Addon string is "2h4usk" and delete step is 5
Second cycle: Addon string is "73uytg" and delete step is 9
Third cycle: We don't have any addon string but we have delete steps that equals 4



```
complexDelium := delium.D256C("Hello, world!", "2h4usk#5/73uytg#9/#4").String
// 37eb8467a1833cb0fa48d995a0ad59f931b0a12d4c2d89d217c22913bd9ebea5
```

What is happening in D256C in here:
1. Cycle 1: Hashing "Hello, world!" with sha-256 method
2. Cycle 1: Add ***"2h4usk"*** to the end of hash
3. Cycle 1: Hash with sha-256
4. Cycle 1: Delete every 5th chars from the hash to use as new input data
5. Cycle 1: Hash with sha-256 the final output of cycle 1
6. Cycle 2: Add ***"73uytg"*** to the end of hash
7. Cycle 2: Hash with sha-256
8. Cycle 2: Delete every 9th chars from the hash to use as new input data
9. Cycle 2: Hash with sha-256 the final output of cycle 2
10. Cycle 2: Add **""** to the end of the hash
11. Cycle 2: Hash with sha-256

12. Cycle 2: Delete every 4th chars from the hash to use as new input data
13. Cycle 2: Hash with sha-256 the final output of cycle 3

String output: *f77988e0a42bb53d4a022cb519028cc42426b9eed18981e0ece43e62c3ead631*

Technical Overview

BLVchain was developed as a super-fast helper for enterprise servers to establish chained security for server data. Ensure data signing and verification for efficiency, transparency and scalability.
If a problem occurs on an organization's servers, BLVchain can help them recover lost data or remove and replace real data stored in blockchain databases, but all data in blockchain is immune.
Blockchain has a full log report where everything is stored in detail to check or fix security issues.
Also, BLVchain has a JavaScript library to create key pairs (private key, public key) based on a user password and make signature. Normal blockchains simply create the key pair with 12-24 mnemonic keys, but in organizations the situation is different. So a user can create a key pair based on a user-defined password, but the password MUST be strong. All key pair generation and data signing is done on a user's client, e.g. the browser. BLVchain does not receive/send private keys or passwords in any form. Therefore, the JavaScript library was developed to perform key generation and signing securely and in the user client.
BLVchain works as a powerful, secure and lightweight tool. It can therefore be used directly for clients or in parallel with the organization servers. The organization servers serve the clients as before, but they use BLVchain as a chain and communicate with it as a security check. We recommend using BLVchain as a parallel server to check and verify data. The servers can store all data as a string and read it in the blockchain.
Because of the deletion of miners, we use the Delium method. Also, we add data verification to each node so that every time data is added/read in the blockchain, each node verifies the data signature and then does the processing. So if wrong data is found in any way, the nodes will recognize it and boycott it forever.

Software and version

The core of BLVchain is written with Golang and uses Mongodb as database. It also uses websocket for node connections and gRPC for blockchain communication with other nodes. Docker is used to dockerize the core as a container so that it can be easily deployed and used.

Softwares and versions table

No.	Software	Version
1	Golang	1.23
2	Mongodb	8.0.4
3	Docker	27.x

How BLVchain works

BLVchain uses ECDSA (Elliptic Curve Digital Signature Algorithm) with the curve p-256 (also known as secp256r1 or prime256v1) as the elliptic curve for the signature procedure. ECDSA is a cryptographic method used for digital signatures that ensures authentication, integrity and non-repudiation in blockchain and other secure systems. It is based on elliptic curve cryptography (ECC), which offers high security with smaller key sizes compared to traditional algorithms such as RSA. It provides a 256-bit security level while ensuring efficiency and security.

ECDSA (Key Generation, Signing, and Verification)

A. Key Generation

In ECDSA, a key pair consists of:
Private Key (d): A randomly generated 256-bit integer.
Public Key (Q): A point on the elliptic curve, derived from the private key as:

$Q = d\,.\,G$

where G is the predefined generator point on the curve.
The public key can be shared publicly, while the private key must be kept secret.

B. Signing a Message

To create a digital signature for a message M:

1. Hash the data(message) using D256 (Delium-256bit) to get a **digest** of the data(message).
   $e = D256$(M)

2. Generate a random integer (k): Choose a cryptographically secure random number k ensuring it remains secret.

3. Compute the curve point (R):
   $R = k \, . \, G$
   Let r be the x-coordinate of R. If $r = 0$ choose another k

4. Compute the signature component (s):
   $S = k^{-1} . (e + d.r) \; mod \; n$
   where n is the curve order. If $S = 0$ choose another k

5. The final signature (r, s) is sent along with the message.

C. Verifying a Signature

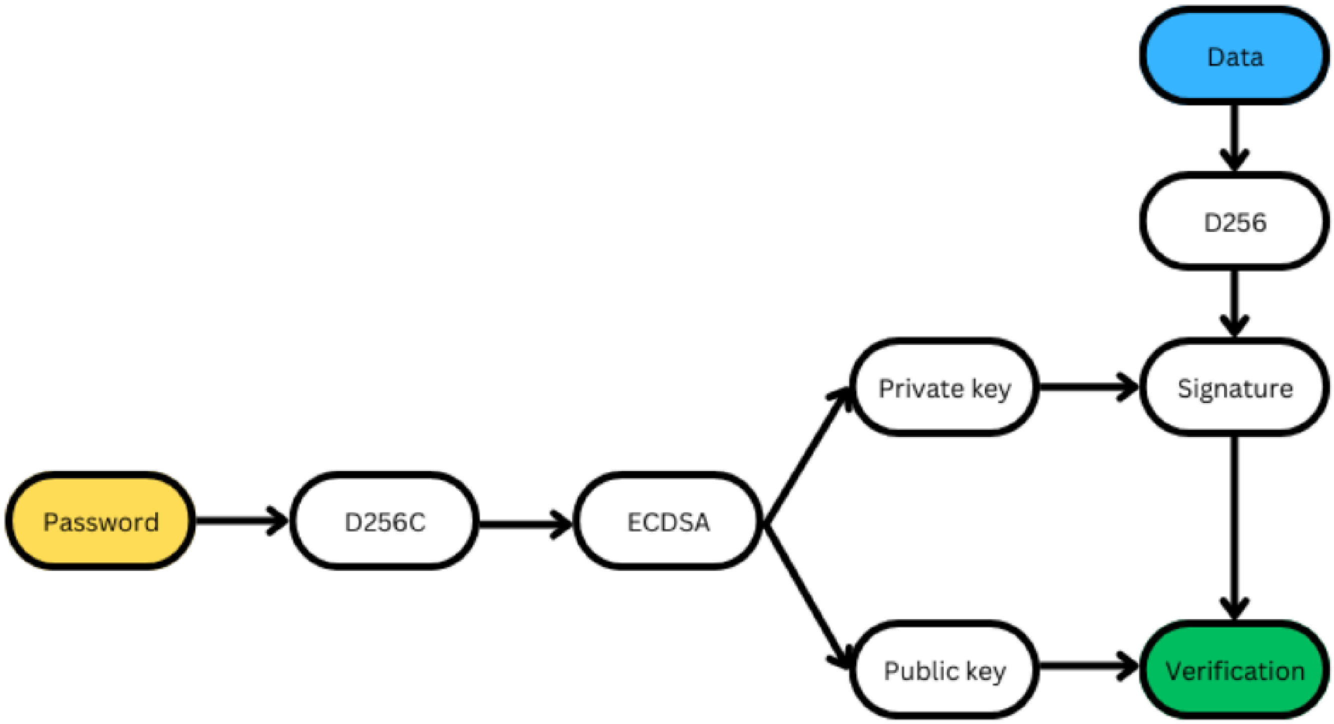
1. Hash the data(message) using D256 (Delium-256bit) to get a **digest** of the data(message).
   $e = D256$(M)

2. Compute signature inverses:
   $w = s^{-1} \; mod \; n$

3. Compute two elliptic curve points:
   $u_1 = e \, . \, w \; mod \; n \quad u_2 = r \, . \, w \; mod \; n$
   $P = u_1 \, . \, G + u_2 \, . \, Q$

4. Extract the x-coordinate of P and check:
   $r' = x\_coordinate \; of \; P$
   If $r' \equiv r \; mod \; n$ The signature is valid. Otherwise, it is invalid.

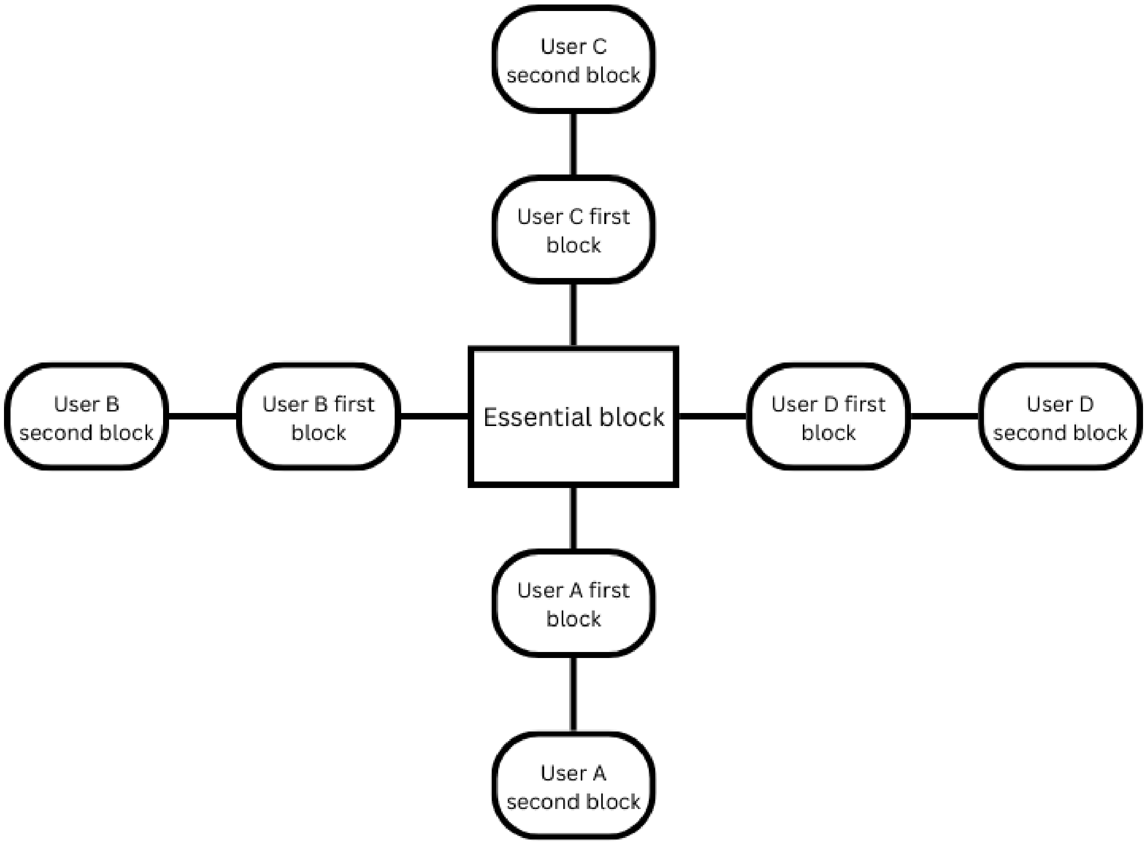
ECDSA diagram



Block

In contrast to conventional blockchains, where a lot of data is combined into a block and which consume a lot of energy and sources to generate a valid block, with BLVchain single data is assigned to a block so that we do not have to generate a block with a lot of data and find a valid hash first.
So the question is, how does BLVchain manage the block branches? The answer is that each node creates an essential block that has a hardened value. This is an essential block that every blockchain needs. After the essential block, there are two options for a user: 1 - First data transaction 2 - After the first data transaction. In the first data transaction, this block is chained to the essential block, but in the second option, the data is chained to the last block of the user who acts as the sender. So we have a structure like this:

Block sequence diagram

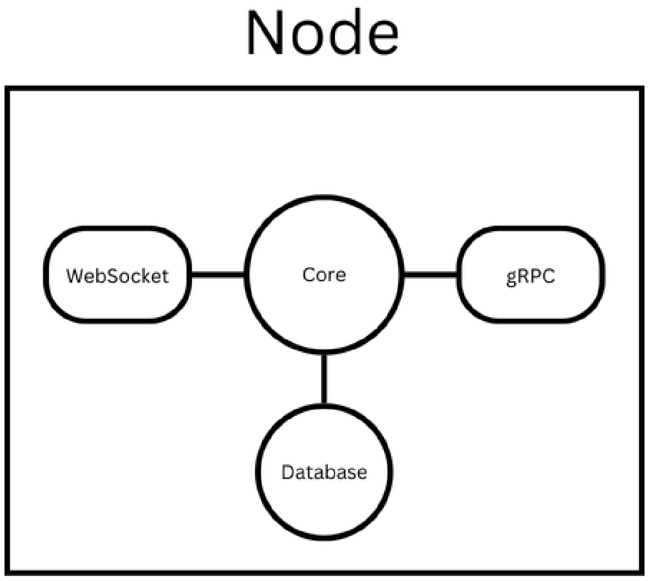


Node

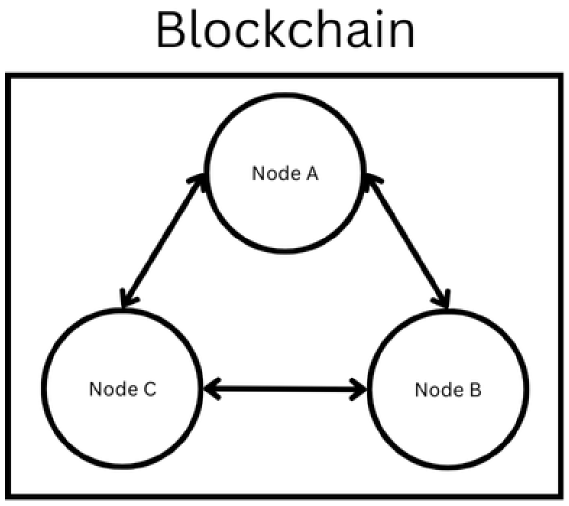
General explanation

A decentralized blockchain requires at least 2 nodes to run the blockchain. Each node has a configuration file that helps it to act based on this file. All configurations, whether Delium configuration and hashing or DNS seed, must be the same for all nodes. The inner connection between the nodes is done via the Websocket API for real-time connections and the outer connection with other applications is done via gRPC for cross-platform support and high speed.

Single node diagram



Blockchain nodes connection diagram



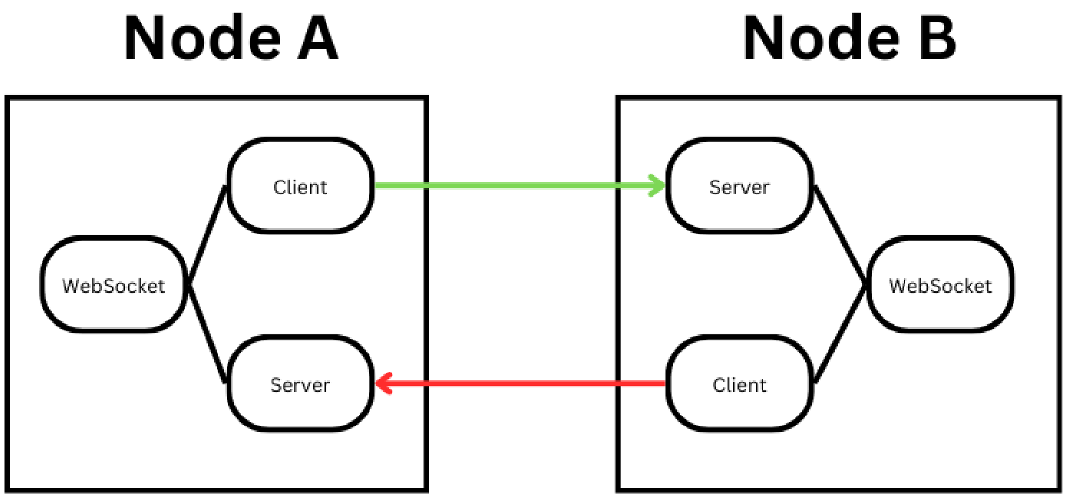
Communication with each other

BLVchain is designed for use in companies and therefore requires a fast online connection between the nodes. Therefore, each node uses websockets to connect to other nodes. Each node therefore has a server-side and a client-side websocket API and each node can find and connect to each other and transmit entire data or parts of it.
For internal security, all data sent/received must pass through the verification section and is then stored in the database. So the question is, what happens if incorrect data is stored in the DB? The answer is that there is no way to store incorrect data in BLVchain, but it is possible for someone who has access to the server to manually change the data in the database. If a node requests new data after this change, the incorrect data can be detected and the data will be boycotted forever in that database. If this incorrect data is stored in all nodes, it will continue to be boycotted in all nodes and there is no way to branch or use this block again.

Why Websocket?

- **Persistent & low-latency connection:** WebSocket maintains an open connection between the nodes so that no repeated handshakes are required as with HTTP, which reduces delays.
- **Efficient data propagation:** Data transactions or blocks are transmitted immediately and without polling, ensuring fast consensus and synchronization.
- **Lightweight overhead:** Compared to other protocols such as HTTP polling, WebSocket reduces network congestion and resource consumption.
- **Event-driven architecture:** The nodes listen for real-time updates and can thus react immediately to new data transactions and block additions.

Nodes connection with Websocket diagram



Communication with other apps

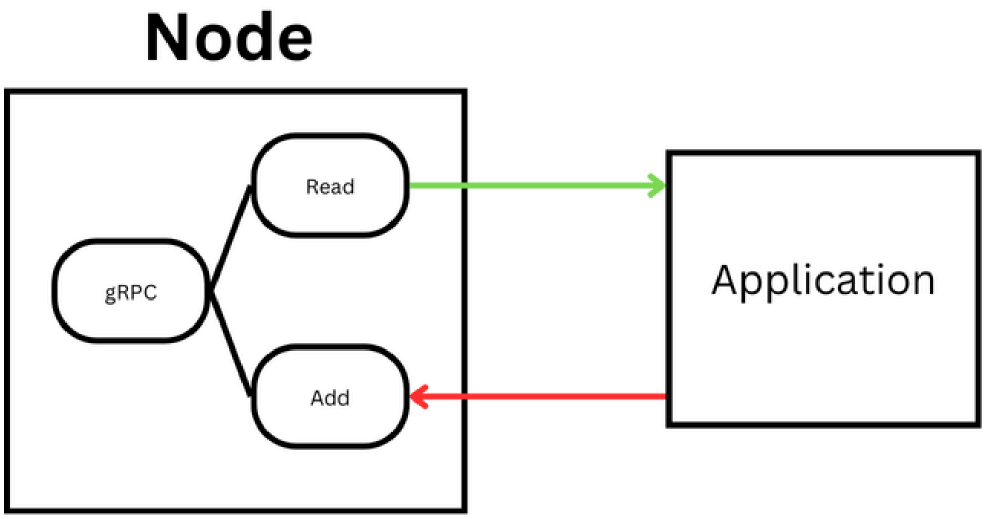
BLVchain uses gRPC to connect other applications written in any language. gRPC is a powerful API. It allows organizations, enterprise systems and third-party services to efficiently access blockchain data, send data transactions and integrate blockchain functionality into their own applications.

Why gRPC?

- **High-speed communication:** gRPC is optimized for low-latency, high-performance interactions, making it ideal for enterprise applications.
- **Cross-platform compatibility:** Supports multiple programming languages (Go, Node.js, Python, Java, etc.), ensuring easy integration with different tech stacks.
- **Structured data exchange:** Uses protocol buffers (ProtoBuf), which reduces the size of the payload and improves efficiency compared to conventional REST APIs.

BLVchain gRPC API consists of two gateways 1-Read 2-Add. The Read section allows you to retrieve data with a custom filter, limit and skip option and the Add section allows you to add new data transactions to the entire blockchain.

gRPC connection with other app diagram



Security issues

There is no software in the world that is 100% secure! There is no exception to this rule. Every software has its strengths and weaknesses in the area of security. When using BLVchain, i.e. for protected organizations, an attack from the outside is safe, but if someone gains access to the node servers in any way, there is a certain possibility, which we will explain, although this probability is very low.
So in this section we will clarify how someone can attack the blockchain and what we do to protect the data.

1. **Add wrong data from gRPC**
   Each node, before adding data from gRPC, checks all the data with the signature and, if it is valid, adds it to this node. After adding the new data to this node, it sends the new data to other nodes. The other nodes perform the same check and then add the new data as a new block.

2. **Add wrong data from Websocket**
   If someone can send false data from the websocket API in any way, the verification process will run on it and prevent false data from being added to the blockchain.

3. **Add wrong data directly in one/all database(s)**
   If someone has direct access to the node server and its database and adds false data to the database when a method, be it the add/read method, is called from gRPC or other nodes, the entire blockchain understands that there is a false block in the database and the entire blockchain will boycott that block forever.

4. **Edit data directly in database(s)**
   Here are two possibilities: 1. edit the last block 2. edit the middle block. With both options, this block is boycotted forever after verification and is useless like other blockchains in the world.

5. **Delete data directly in database(s)**
   There are also two possibilities here: 1.delete last block 2.delete middle block.
   If a block is deleted by n-1 nodes, a node that has the data of the deleted block helps the other nodes to recover the lost data. However, if a block is deleted by all nodes, the lost data must be added from outside the blockchain using gRPC or backup servers.
   If only the last block is deleted from all nodes, there is no problem and the blockchain continues to function, but if a middle block is deleted from the entire blockchain, the

branch of this chain is interrupted. If this happens to n-1 nodes, other nodes can reinstall the software to get completely new data, but if this happens to the entire blockchain, the branch must be boycotted forever to ensure the security of the entire blockchain.
This security issue can occur with any blockchain, but based on the private blockchain we have, this probability is very low.

All read/add data from all APIs is stored as a log in the log folder, and the operator can monitor and recognize warnings, errors and successful functions in each node.

Contributor

I am Yahya Parvin Aghdam, who designed and created this blockchain in two years. I always wanted to find the fastest and best way to secure data, and I finally did it.
This software is already private, but you can see the usage and tests on BLVchain youtube:

I would be happy if someone wants to invest in this project. If you want to contact me, just send me an email:
yahyaaghdam.ir@gmail.com