BLVchain

By
Yahya Parvin Aghdam
2023-2025

Website: https://blvchain.org

Email: yahyaaghdam.ir@gmail.com

Abstract

As digital transformation accelerates, government agencies and enterprises face increasing challenges in data security, efficiency, and transparency. Traditional systems often suffer from bureaucratic inefficiencies, security vulnerabilities, and lack of trust among stakeholders. A private (enterprise) blockchain provides a secure, scalable, and tamper-proof solution tailored to meet the unique needs of organizations and governmental institutions.

1. Data Security & Integrity
Government agencies manage sensitive data, including citizen records, financial transactions, and legal documents. A private blockchain ensures:

- Immutable Records : Once data is recorded, it cannot be altered or deleted, preventing fraud and corruption.
- Advanced Encryption : Sensitive information is protected using cryptographic hashing.
- Controlled Access : Only authorized entities (e.g., government departments) 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.

3. Efficiency & Automation with Smart Contracts
Manual processes in government services, procurement, and financial operations can be slow and error-prone. Blockchain enables:

- Smart Contracts : Automates administrative tasks (e.g., fund disbursement, document verification).
- Faster Approvals : Reducing paperwork and speeding up decision-making processes.
- Error Reduction : Minimizing human mistakes and inconsistencies.

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Government entities often struggle with fragmented and outdated IT systems.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.

7. Use Cases of Private Blockchain in Government & Enterprises

- 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.

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

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

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

This whitepaper outlines the technical architecture, advantages, and real-world applications of BLVchain, providing a roadmap for businesses and developers seeking to harness the power of decentralized technology without the inefficiencies of legacy systems.

Introduction

Limitations of Traditional Blockchains

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

A Miner-Free, Lightweight Blockchain for Enterprises

Our blockchain is designed specifically for businesses that need a secure, scalable, and lightweight distributed ledger solution. By eliminating mining with a specific algorithm, our network achieves high-speed data transactions, low operational costs, and improved energy efficiency.

Easy deploy, Easy connection

With gRPC technology BLVchain can connect to any programming language with super high speed and communicate with other servers. Also installing and running a node of blockchain is easy just with a json config file and running a Docker container.

The Delium Method

Introduction

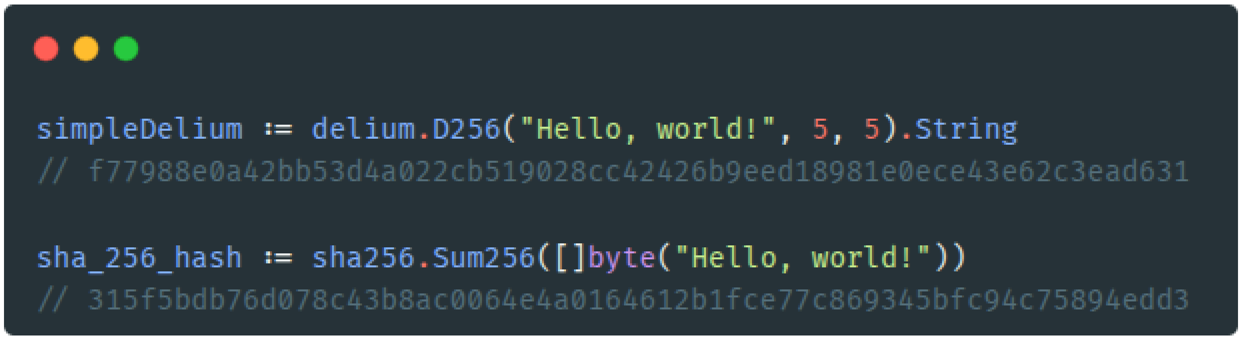
This method, inspired by the film Oppenheimer, is derived from the concept of an atomic explosion. It works by visualizing a string, encoded using the SHA-256 or SHA-512 method, as a specific number of atoms placed together. In an atomic explosion, almost all atoms are destroyed, but our goal is to mathematically make the reverse engineering probability almost zero. Therefore, we remove several specific character sequences from the strings and then re-hash the resulting string using the SHA-256 or SHA-512 method. This cycle will repeat until we have a final output hash.
The ultimate goal of this method is every organization blockchains have their own specific hashing method. Every delium hashing method has two elements: 1- delete steps 2- repeat
So, every final hash can change by changing those two elements while the normal sha-256/512 method always has the same output with the same input.
Delium developed especially for BLVchain by Yahya Parvin Aghdam and it is open source to use https://github.com/blvchain/delium

Delium developed to make custom output from single input based on config, and creating resistance against brute force attack or quantum attacks. Also because of deleting miners, the blockchain must have a secure way to make the whole blockchain and data transactions and signatures safe and stable.
So it developed in two methods 1.Simple delium 2.Complex delium.
In Simple method we just have delete step and repeat, but in Complex method in addition to that we have addon string too. In a cycle hashed string will addon with another string and will hash again with delete step and repeat. Complex method uses 'path' for this reason.
Therefore the same input data can have an infinite output hash based on delium config.
This method can give reassurance that every organization has a unique hashing method.

Simple Delium

In simple delium we just have two elements 1.Delete steps 2.Repeat lets see how it works and what is the difference between sha-256 and D256
(We use D256 for example but D512 also accessible)

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

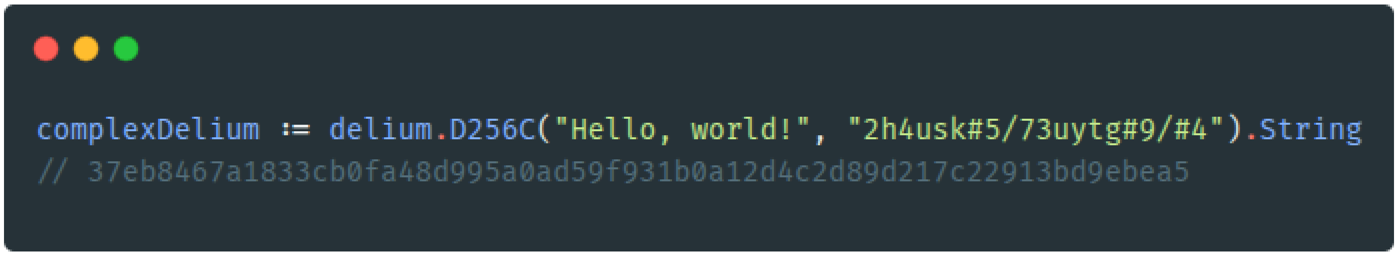
String output: *f77988e0a42bb53d4a022cb519028cc42426b9eed18981e0ece43e62c3ead631*

Complex Delium

This method is designed to direct delium hashing to a specific path. So we have a path to guide delium to hash 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 is designed to be super fast as a helper for organization servers to make chained safety for servers data. Make sure that data signature and verification makes efficiency, transparency, and scalability.
If any issue appears in organization servers, BLVchain can help them to recover lost data or remove and replace true data that are stored in blockchain databases, but all data in blockchain are immune.
Blockchain has a full Log report that saves everything with detail to check security issues or debug.
Also BLVchain has a JavaScript library for creating key pairs (private key, public key) based on user password. Regular blockchains just make the key pair with 12-24 mnemonic keys, but in organizations the situation is different. So, a user can make a key pair based on a custom password, but the password MUST be strong. All key pair generation and data sign will be in the client on a user such as the browser. BLVchain does not receive/send any private key or password in any way. Therefore the JavaScript library was created to make key generation and signature safe and in user client.
BLVchain operates as a strong, safe and light tool. So, it can be used directly for clients or can be used parallelly next to the organization servers. Organization servers will serve clients like 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. Servers can save any data as String and read it in the blockchain.
Because of deleting miners, we use Delium method. Also we add data verification to every node in such a way that every Add/Read in any data of blockchain, node(s) will verify the data signature and then will do the process on it. So if any wrong data in any way is found, node(s) will detect and boycott it forever.

Software and version

Core of BLVchain is written with Golang and uses Mongodb as a database. Also it uses Websocket to node connections and gRPC to communicate blockchain with out of nodes. Docker used to dockerize core as a container to easily deploy and use.

Softwares and version 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 p-256 curve (also known as secp256r1 or prime256v1) as an elliptic curve for the signature process. ECDSA is a cryptographic method used for digital signatures, providing authentication, integrity, and non-repudiation in blockchain and other secure systems. It is based on elliptic curve cryptography (ECC), which offers strong security with smaller key sizes compared to traditional algorithms like RSA. Offering 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

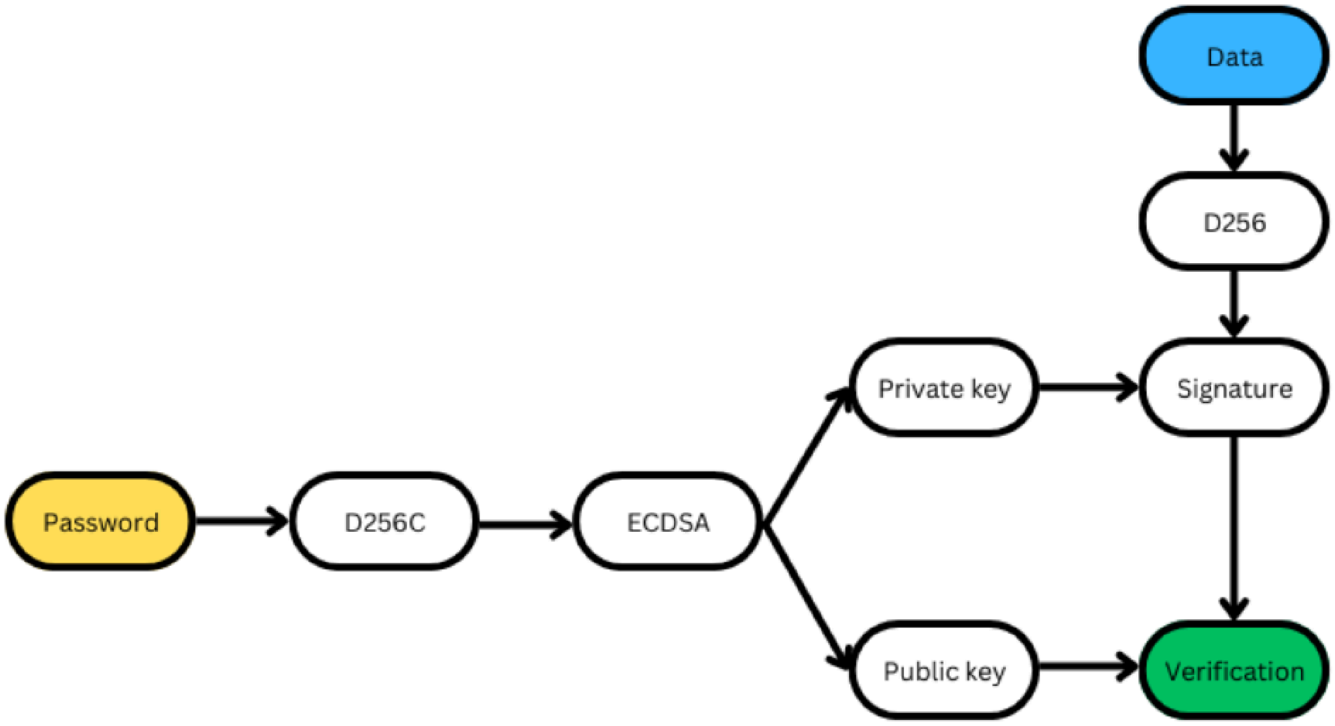
1. Hash the data(message) using D256 (Delium-256bit) to get a **digest** of the data(message).
   $e = D256$(M)

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

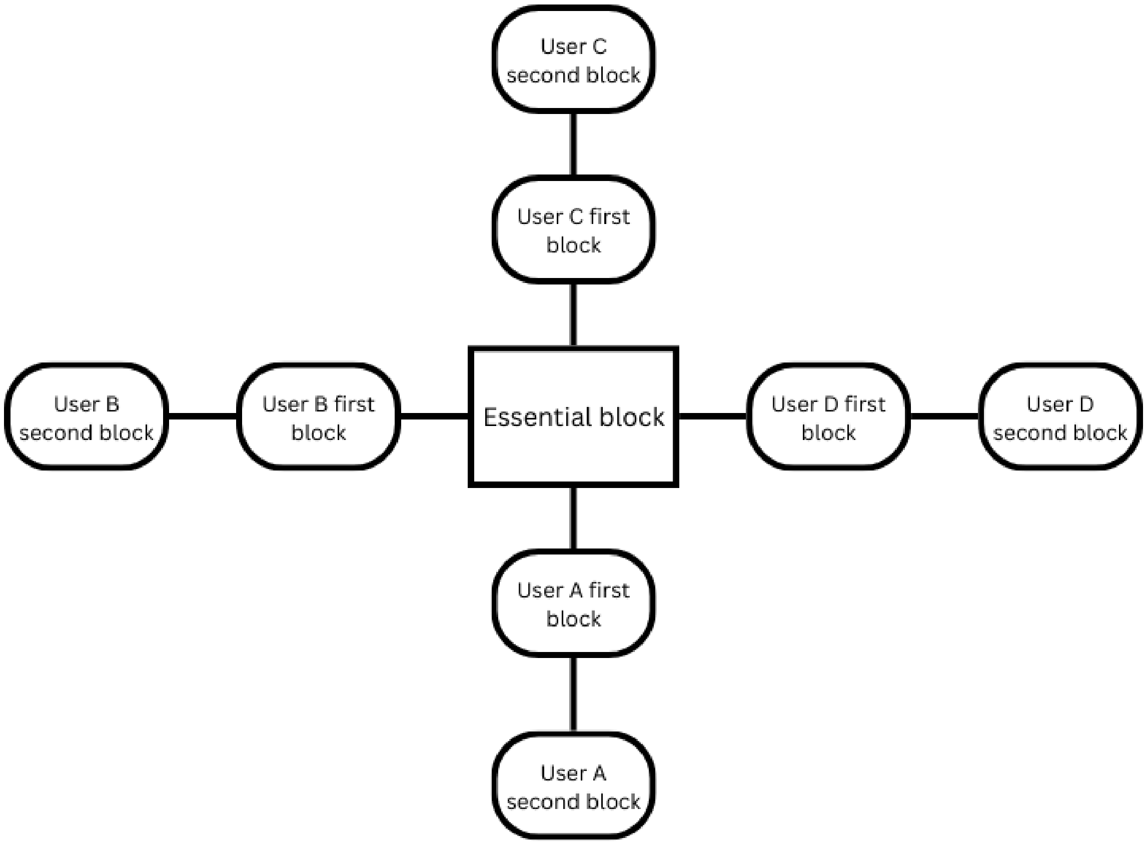


Block

Unlike traditional blockchains that put a lot of data to make a block and pay so much energy and source to make a valid block, BLVchain assigns every data to one block, so we don't need to make a block with a bunch of data and find a valid hash. Every data transaction will save as a block after signature validation.

So the question is how BLVchain manages the block branches? The answer is every node makes an essential block that has hardened value. This is an essential block that every blockchain needs. After the essential block, There are two possibilities for a user 1-First data transaction 2-After first data transaction. In the first data transaction this block will be chained to the essential block, but in second possibilities data will be chained to the last block of the user that is sender in it. So, we will have an structure like this:

Block sequence diagram

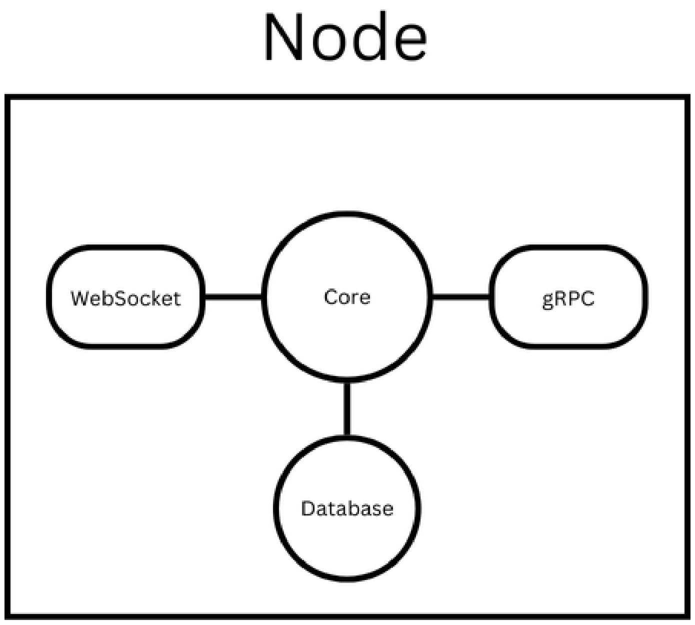


Node

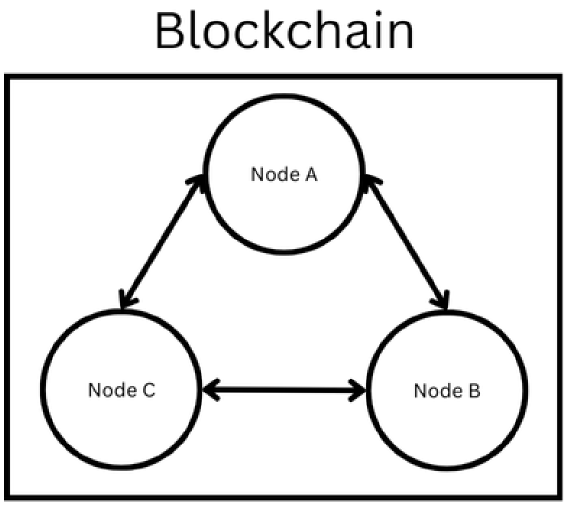
General explanation

In decentralized blockchain, having above 2 nodes is necessary. So, we need at least 2 nodes to run blockchain. Every node has a config file that helps it to act based on it. All configurations whether Delium config and Hashing or DNS seed must be the same in all nodes. Inner connection between nodes is with Websocket API for real-time connection and outside connection with other applications is with gRPC for cross-platform and high-speed support.

Single node diagram



Blockchain nodes connection diagram



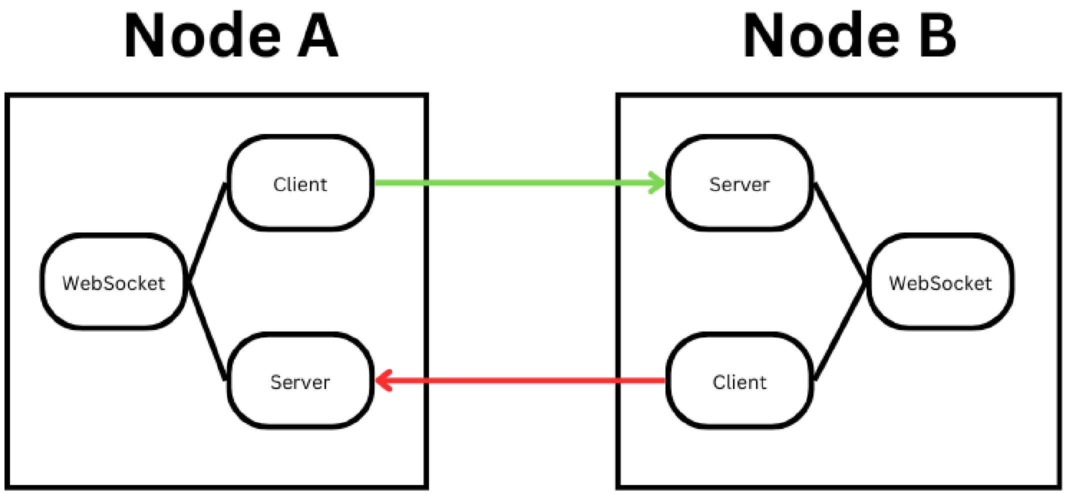
Communication with each other

BLVchain is designed for organization use, so it needs to have fast and online connection between its nodes. Therefore every node uses Websocket to connect to other nodes. So every node has a server side and client side Websocket API and every node can find and connect each other and communicate whole or part of data.
For inner security every single data that is sent/received has to pass the verification section, then will be saved in the database. So, the question is what will happen if a wrong data in any way is saved in DB? The answer is there is no way to save wrong data from BLVchain but it is possible to someone has access to server, change manually the data of database. After this change if one node requests new data or whole data can detect the wrong data and that data will be boycotted forever in that database. If this wrong data is saved in all nodes, it is still boycotted in all nodes and there is no way to branch or use that block again.

Why Websocket?

- **Persistent & Low-Latency Connection:** WebSocket maintains an open connection between nodes, eliminating the need for repeated handshakes like HTTP, reducing delays.
- **Efficient Data Propagation:** Data transactions or blocks are broadcast instantly without polling, ensuring rapid consensus and synchronization.
- **Lightweight Overhead:** Compared to other protocols like HTTP polling, WebSocket reduces network congestion and resource usage.
- **Event-Driven Architecture:** Nodes listen for real-time updates, allowing immediate reaction to new data transactions and block additions.

Nodes connection with Websocket diagram



Communication with other apps

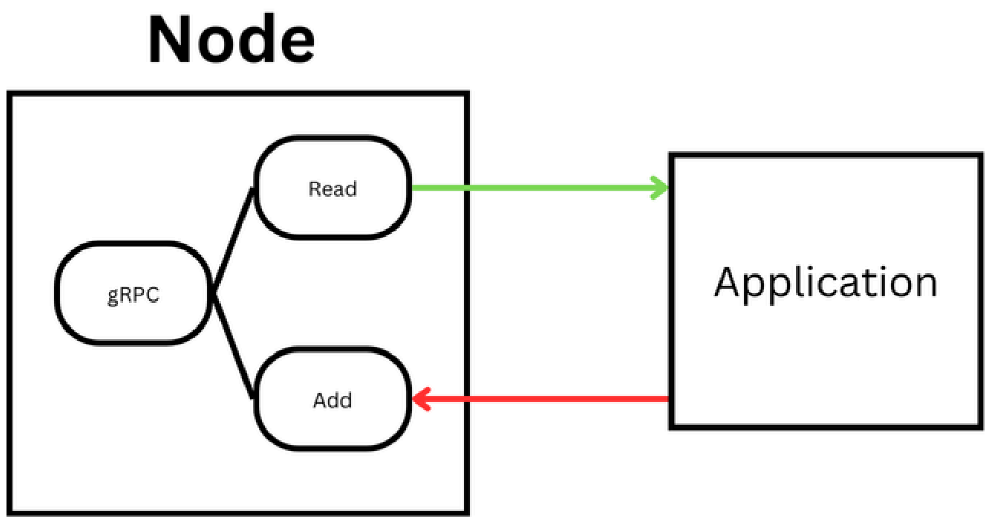
BLVchain uses gRPC to connect other applications written with any language. gRPC is a high performance API. This allows businesses, enterprise systems, and third-party services to efficiently access blockchain data, send data transactions, and integrate blockchain functionality into their own applications.

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 diverse tech stacks.
- **Structured Data Exchange:** Uses Protocol Buffers (ProtoBuf), reducing payload size and improving efficiency compared to traditional REST APIs.

BLVchain gRPC API consists of two gateways 1-Read 2-Add. With the Read section you can get data with a custom filter, limit and skip option and in the Add section you can add new data transactions to the whole blockchain.

gRPC connection with other app diagram



Security issues

There is no software in the world that is 100% safe! There is no exception to this rule. Every software has its strengths and weaknesses in the field of security. In BLVchain usage that is for protected organizations, an attack from outside is safe, but if someone gets access to node server(s) in any way, there is some possibility that 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 defend data protection.

1. **Add wrong data from gRPC**
   Every node before adding data from gRPC, will check the whole data with the signature and if it is valid, it will add to this node. After adding this new data to this node, it will broadcast new data to other nodes, other nodes will do the same verification and then add new data as a new block.

2. **Add wrong data from Websocket**
   If someone in any way can send wrong data from Websocket API, the verification process will be executed on it and avoid adding wrong data to the blockchain.

3. **Add wrong data directly in one/all database(s)**
   If someone has direct access to node server and its database and add wrong data to database, if a method whether add/read method called from gRPC or other nodes, the whole blockchain understand there is a wrong block in database, and whole blockchain will boycott that block for ever.

4. **Edit data directly in database(s)**
   Here are two possibilities: 1.Edit last block 2.Edit middle block. In both of these possibilities after verification this block will be boycotted forever and it is useless like other blockchains in the world.

5. **Delete data directly in database(s)**
   Here are two possibilities too: 1.Delete last block 2.Delete middle block.
   If a block is deleted from n-1 nodes, a node that has the deleted block data will help other nodes to recover lost data. But a block delete from all nodes, lost data must be added from out of the blockchain with gRPC or backup servers.
   If just the last block is deleted from all nodes, there is no problem and the blockchain will continue to work but if a middle block is deleted from the whole blockchain, the branch of

that chain will be broken. If this happens in n-1 nodes, other nodes can reinstall software to get whole new data, but if this happens to the whole blockchain, the branch must be boycotted forever or deleted from the database for security of the whole blockchain.
This security problem can happen for every blockchain, but based on the private blockchain we have ,this probability is very low.

All read/add data from all APIs is saved as a log in log folder and the operator can monitor and detect warning, error and successful functionality in every node.

Conclusion

Progress has never been about sticking to what's familiar, it's about embracing what's smarter, faster, and more efficient. In an era where instant access, seamless connectivity, and secure data transactions define success, organizations that adapt quietly gain the edge while others struggle to keep up.

After launching this blockchain you will be sure about Digital Signature Authenticity, Information transparency and Security of whole data at the lowest cost.

Contributor

I am Yahya Parvin Aghdam who designed and created this blockchain in two years. I always wanted to find the fastest, least expensive, and best way to secure data and finally I managed to make it.
I’ll be so glad if you want to have this blockchain on your system. So, you can have a demo version of blockchain with 20 blocks limit to test it. If you want to test this software feel free to send an email to me: yahyaaghdam.ir@gmail.com