APOLLO CURRENCY

THE WORLDS FASTEST MOST FEATURE RICH COIN

TECHNICAL DOCUMENT 2.0

Apollo FINTECH

CONTENTS

INTRODUCTION

The current document contains the description of implemented solutions, goals and methods of development for the Apollo project, in accordance with the presented roadmap. Further development of the Apollo project and technologies are declared.
Applied blockchain protocols and supported services are numbered sequentially depending on new implemented functions during project development (in accordance with the declared roadmap).
The Apollo blockchain project is aimed at developing innovative technologies and properties. All technologies of the Apollo project are used and will be used to build different services, eventually uniting these services into a single Apollo ecosystem. At the same time, services can be provided using APIs for use by third-party projects and services

CURRENT STATE OF APOLLO CURRENCY

Olympus 3.0 protocol with IP Masking 2.0 function (support for TOR technology) has been launched. These provide support of web-wallets and desktop versions for different platforms.
Using TOR and new types of transactions (anonymous) leads the anonymity paths of data packets. This makes it impossible to monitor traffic inside the TOR network.
The current version of the block explorer, support of the market platform, voting system, assets, exchange, message forwarding and other functions are available. Combined, these functions favorably distinguished the Apollo system from other solutions, even at the initial stage.
Use of aliases, and coin mixing will return for use in a near update, after its long awaiting modified protocol following a complete blockchain re-factor.

APOLLO CURRENCY

The branches of the Apollo project development reflect parallel lines of development of different properties. Hermes protocol is speed, separation of various segments of the blockchain and additional functionality. Olympus protocol is anonymity, security, and services for parameters and hiding transactions, incl. for external projects.

HERMES PROTOCOL V1.0

The following modules have launched and have radically changed and improved the functioning of the detachment:

ChainID

is a function that allows the separation of different types of blockchain networks and its segments, to separate and protect against parasitic traffic, respectively, to make the operations faster and more stable.

ApolloUpdater

is a module that allows its developers to quickly and efficiently add new features, update the Apollo system, speed up the release of new versions, and to avoid blockchain “hard forks”. At the same time, the ApolloUpdater protects from unauthorized, hacking attempts to update the application. A new type of transaction is introduced that allows its developers to implement the update procedure automatically, securely, with verification of the cryptographic digital signature.

Block2sec

is a function which allows the creation of blocks within 2 seconds, which is much faster than most existing solutions. This function allows the blockchain to speed up the processing of a high number of transactions as well as speed up the system in general.

Additionally

the block size will also be increased, cryptographic functions will be refined and additional validations of the incoming packets will be performed. The digital signature during the development, installation and update of applications is fully implemented.

IP Masking 2.0

is a transport protocol that allows anonymous encrypted packet delivery. This function prevents the tracking of transaction paths, and is one of the elements of complete anonymity of transactions. The development of a proprietary protocol is justified by the recognition and blocking of TOR in some countries, as well as by the inability to configure data exchange parameters in the TOR network. To implement transport between nodes, a VPN channel is created, a tunnel through which all data is transmitted in the form of standard protocol packets used by standard web-browsers.

ApolloMixer

is a transaction-mixing system for hiding the sender, recipient and transaction amount. At the same time, transactions are encrypted during forwarding and in this form are delivered to the mixer. All transactions in the mixer are divided into 3-7 parts (the number of parts is selected randomly), sent randomly (3 to 10) one-time use wallets, which are destroyed immediately after the operation, finally these parts arrive in the wallet of the recipient. In this case,
the absence of the same parameters in transactions makes the 6 restoration of the original data almost impossible. Third parties can track open transactions, but the crypto mixer mixes transactions with many others, or exchanges funds for funds received from many other users. This is so that the funds received will no longer be associated with the original owners.
The Elliptic Curve Digital Signature Algorithm (ECDSA) is used to encrypt transactions inside the Apollo network. It is a public-key algorithm for creating a digital signature, defined not over the ring of integers, but in the group of points of an elliptic curve. The strength of the encryption algorithm is based on the problem of a discrete logarithm in the group of points of an elliptic curve. Unlike the problem of a simple discrete logarithm and the problem of factoring an integer, there is no subexponential algorithm for the discrete logarithm problem in the group of points of an elliptic curve.
The choice of key length and elliptic curve points corresponds to the US National Institute of Standards and Technology (NIST) recommendations. Key exchange uses the Diffie-Hellman protocol on an elliptical curve, a cryptographic protocol that allows two parties that have open/closed key pairs on an elliptical curve to obtain a shared secret key using an unprotected communication channel.

This secret key can be used both for the encryption of further exchanges, and for the formation of a new key, which can then be used for subsequent exchanges of information using symmetric encryption algorithms. This is a variation of the Diffie-Hellman protocol using elliptical cryptography.
The functions of content evaluation and storage time in services using third-party files are implemented. Storage of external content is carried out in a separate, independent repository. This increases the speed of data processing, data security, and the ability to delete unnecessary content by the owner.

Two Factor Authentication

The implementation of a two-factor authentication mechanism for vault wallet addresses will allow for increased security in regard to user identification. This additional security protocol would make the theft of a user's password pointless. Two-factor authentication is an enhanced authentication protocol which utilizes an access control method in which more than one type of proof is required to provide the user with access to data.

HERMES PROTOCOL V2.0

ApolloSharding

allows the blockchain to be split into segments, which positively affects the volume of the database of transactions, download speed, the speed of the blockchain, and the stability of the system. Multilevel sharding allows the system to flexibly adjust the size of the block and the number of transactions in the database, all depending on the region, time and other parameters. Sharding also allows transactions to be divided into parts, this permits each node in the network to process only the components of a block, not the entire block itself. This division allows the system to process transactions faster and facilitates sustainability for the Apollo network.

Adaptive Block Generation

is the mechanism for creating blocks only in the presence of transactions. This function affects the speed and volume of the database and allows for guaranteed transactions. Guaranteed transactions are a new type of prioritized transaction performed regardless of the progress of creating new blocks and new transactions.

OLYMPUS PROTOCOL V3.0

IP Masking 3.0

is an implementation of the zkSnark mechanism that is based on an anonymous transaction library. This function allows the system to hide the sender, recipient and amount sent by encryption, with the ability to verify the transaction without access to the source data (proof with zero disclosure). The goal of zero-disclosure proof is to ensure that the auditor can ascertain that the auditee has knowledge of a secret parameter, called a “proof ” that satisfies certain relationships, without disclosing the certificate to the examiner or to anyone else.

The essence of the proof with zero disclosure is a program denoted by C, which takes two parameters: C (x, w). The parameter x is a public value, and w is the secret value of the proof. The program returns a Boolean value, that is, either “true” or “false”. The goal is to get such a public value of x that you can verify that the verifier knows the secret value of w, such that C (x, w) == true.

Private Ledger

is a function which implements the creation of anonymous wallets.

Users can carry out anonymous transactions from these wallets using a full range of anonymous services. This offers users a 100% guarantee of anonymity on transactions, users, nodes and services.

HERMES PROTOCOL V3.0

FastApollo

is an instant transaction protocol that allows the system to conduct a transaction directly between two nodes of the Apollo network. This significantly reduces transaction time, adds guaranteed execution and enhances the ability to build distributed independent systems (for example, a distributed exchange). This protocol allows for instant transactions between participating nodes and is proposed as one of the solutions to the scaling problem. The Apollo network will consist of nodes and bidirectional payment channels which are installed between two nodes of the network. Each of the two nodes in the payment channel blocks a certain amount of funds for the channel in the blockchain. In the future, the bandwidth of the channel will be composed of the volume of funds blocked by the nodes.

ApolloDEX

is a decentralized exchange and exchange service which allows instant transactions between nodes and instant exchange of various currencies. It can withstand a large number of simultaneous transactions and works on the basis of the FastApollo protocol.
Decentralized exchanges differ from centralized exchanges, as they allow users to control their funds, managing their most important functions on the blockchain. ApolloDEX uses the technology of the applied cryptocurrency itself, thereby ensuring the security and transparency of the trade. ApolloDEX solves the main problems faced by cryptocurrency markets, since they do not have a single point of failure.

OLYMPUS PROTOCOL V4.0

ApolloMobile

involves the launch of a mobile application for wallets on various platforms. It is the implementation of security functions, identification, wallet management, cryptocurrency exchange services and other Apollo ecosystem services.

This protocol includes the development of a public application programming interface (API) for connecting third-party services and projects to Apollo system. This API allows developers to use readymade blocks for building the compatible applications.

HERMES PROTOCOL V4.0

SmartApollo

is the implementation of a smart contract mechanism. It is the ability to create complex data processing scenarios. Smart contracts enable the system to perform reliable and confidential transactions without the involvement of external intermediaries. In addition, such transactions are traceable, transparent and irreversible. Smart contracts not only contain information about the obligations of the parties and sanctions for their violation, but they automatically ensure the fulfilment of all terms of the contract.

ApolloPlatform

is a decentralized platform for executing an ICO on the Apollo blockchain. This toolkit facilitates generating tokens and raising funds for third-party projects, allowing users to collect funds using innovative blockchain technology and smart contracts.

The ApolloPlatform makes it possible to do the following: implement an ICO (primary placement), prepare for the initial placement of pre ICO, implement crowdfunding, provide other services related to public financing, the initial launching, preparation and placement of coins, and more.

The ApolloPlatform module includes an administrative panel for the ICO or event, a client dashboard, a mechanism for connecting third-party payment services, accepting payments in Apollo and the ability to connect KYC and AML services.

Terms and Definitions

This section is a digital document for the Apollo blockchain that contains an exhaustive list of obligations of the parties.
Decentralized Exchange (DEX) is a software complex for P2P exchange of digital assets.
Crypto account monitor (oracle) is a service that defines the current status of the cryptocurrency.
wallets. The order will be deleted by the Crypto account monitor if the funds in the User`s wallet are less than declared in the published order (and than necessary to complete the corresponding contract).
Crypto asset is any coin or token that can be traded at the Apollo DEX (as of September 2019 it is APL, ETH and PAX).
Foreign Blockchain Transfer Module (FBTM) is a service of assets transfer to a third-party blockchain.
Instant order is a direct order for the selected pending order.
Order is an instruction for the exchange (Apollo DEX) to buy or sell a digital asset.
Order book is a pool of pending orders.
Order Book Manager is an order management service that creates the required number of orders with simultaneous multiple exchanges.
Pending order (offer) is an entry in the Order Book for the exchange of N units of some crypto assets for another at the rate M.
Smart contract is a computer protocol (executable source code) in the blockchain that is intended to full execution of contract conditions by both parties.
Apollo DEX is currently using Ethereum smart contract. But in the future, the Apollo DEX will use smart contracts of other blockchains.
Transaction monitor (oracle) is a third-party blockchain monitoring service. It determines whether the transaction was executed in a third-party blockchain.

Abbreviations

API - Application Programming Interface.
APL - Apollo cryptocurrency.
DEX - Decentralized Exchange.
BTC - Bitcoin cryptocurrency.
ERC-20 - Technical standard that is used for smart contracts on the Ethereum blockchain for implementing tokens.
ETH - Ethereum cryptocurrency.
FBTM - Foreign Blockchain Transfer Module.
HTLC - Hashed Time-Locked Contract.
LTC - Litecoin.
PAX – Paxos standart, ERC-20 tokens.
UI - User interface.
P2P - peer to peer

Introduction to Apollo DEX

Decentralized Exchange (hereinafter - DEX) is a software package for the exchange of digital assets at p2p networks. Unlike traditional (centralized) exchanges, DEX does not have a single centre acting as a guarantor of the transaction. In fact, assets are transferred directly from the sender to the recipient under DEX excluding third parties. That gives DEX a number of the following advantages:

- Decentralization and separation of powers: There is no single centre responsible for all trading operations. That means that each member of the network supports the work of the exchange, ensuring its stable and uninterrupted operation. So, DEX is decentralized across the globe and available for trading 24/7.
- The nodes of the network interact according to a protocol that does not know the consequences associated with the assets involved. This ensures the correct processing of contracts by network nodes and eliminates the manipulation of data and assets.
- DEX allows operations with any type of asset or its digital counterparts.
- Reliability and security. User funds and the transaction are protected using modern cryptographic methods. An unauthorized party will not be able to access the user's assets or change the terms of the transaction. The Apollo DEX also proposes additional methods of user data and assets protection. That methods will be configured by Users for a balance of security and convenience.
- Fast. Apollo platform performs almost instant transactions with APL cryptocurrency. Thus, the speed of the Apollo DEX exchange can be very high and is determined only by the speed of transaction processing by third-party blockchains.
- Minimal fees. The absence of a third-party (like centralized exchange) leads to the minimization of contract fees and associated costs.

Hereinafter, let`s consider the Apollo DEX as an example of the modern and one of the most technologically advanced decentralized exchange.

Key Features of Apollo DEX

Apollo team focuses on making the Apollo DEX the most secure, efficient and easy to use.
Let's take a look at the main features of the Apollo DEX platform, which distinguish it from other similar systems.
Atomic swap is a modern technology that allows two parties to exchange crypto assets between two blockchains without intermediaries. Atomic swap solves the problem of performing related operations in unrelated blockchains (for example, Apollo and Ethereum) when both parties must complete their set of actions under the contract, otherwise, the whole transaction will not take place.
TradingView is a graphical representation of the change in the price ratio of two selected assets. Such a tool allows the User to quickly navigate in the current market situation and carry out transactions at the most favourable rate. The TradingView graph and tools will be included on the “Exchange” page of the Apollo web wallet: https://apollowallet.org/exchange
Own Ethereum nodes have been deployed for safe and controlled interaction with the Ethereum blockchain. Using own Ethereum nodes allows to not use third-party services to interact Apollo with Ethereum blockchain. Thus, communication with the Ethereum network becomes more reliable, safe and fault-tolerant.
Matcher is an automated algorithm that allows to find mutually complementary orders and carry out a transaction as soon as possible. Verification is carried out after placing each individual order.

Smart Contracts

Apollo DEX uses Ethereum smart contracts to create and execute orders with some additional conditions and rules. In particular, Ethereum smart contract allows to transfer funds with confirmation from the recipient and to set a time limit for the entire procedure.

The use of Ethereum smart contract in the framework of Apollo DEX is carried out according to the following scenario:
Step 1. The owner of an ETH/PAX account can transfer any number of ETH/PAX currencies. So, that cryptocurrency are blocked for a specified time. After the specified time, the amount is automatically returned to the user`s account.
The User can also transfer blocked funds or its part to another User using the "Send money" method.
Step 2. The owner of an ETH/PAX account can transfer any number of frozen funds to another User using the "Send money" method. When sending funds, the User must indicate the secret, recipient address, amount and time to confirm this transaction.
The recipient must confirm the transaction with a secret key within the specified time. Otherwise, the funds will return to Step 1.
In case of successful confirmation of the transaction by the User 2 (by the secret key), the recipient immediately receives the funds, and the secret key must be published and available for viewing.
Transaction confirmation mechanism:
secret hash = SHA256(secret key).
Also, see the section “Atomic swap” for details.

Apollo DEX: principles of work

Trading on the Apollo DEX is carried out by placing appropriate orders. The User has the opportunity to specify the desired conditions of the contract (exchange rate). This undoubtedly provides a benefit to the User.

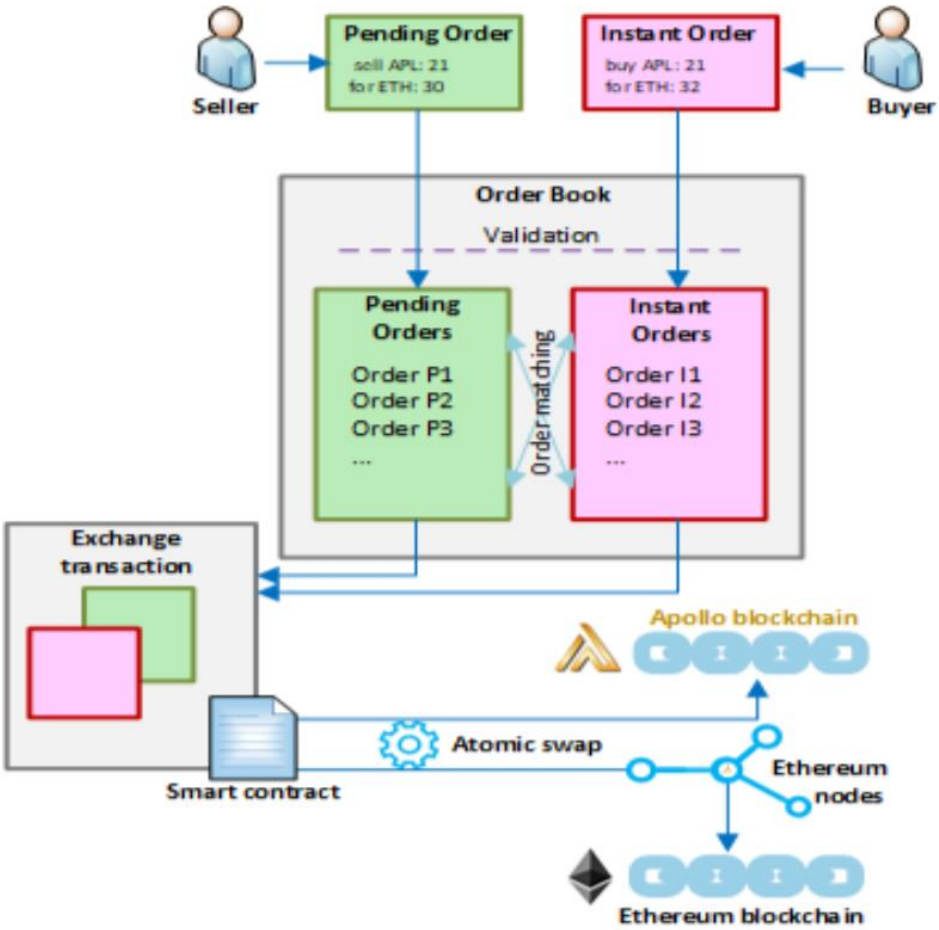


Figure 1. General scheme of Apollo DEX functioning.

The contract execution at the Apollo DEX includes the following sequence of actions:

1. 1. User 1 (named "Seller" in fig. 1) places a pending order to sell (exchange) some asset at the desired rate through a corresponding user interface (see “UI Description” section for details). This pending order is checked and placed in the order book.
2. User 2 (named "Buyer" in fig. 1) places an instant order to buy (exchange) another asset at the desired rate. This instant order is checked and also placed in the order book.
3. The order matching system searches for a suitable counter order. User orders 1 and 2 turned out to be the most comparable in the system.
   Note that the order matching system searches for a suitable counter order for each new order in the order book. A suitable order can be found by the order matching system immediately after placing the order by User 1. See the “Order matching system” section of the current document for details.

4. By the atomic swap mechanism, Apollo DEX forms a smart contract that reserves assets for an exchange on the users' accounts and waits for confirmation of the exchange from Users 1 and 2. See also the “Atomic swap” section for details.
5. After the mutual confirmation of the asset exchange, Apollo DEX exchanges crypto assets. Information about the transaction is entered into the blockchains of both cryptocurrency systems between which the exchange was carried out.

Purchase of APL for a third-party cryptocurrency will be carried out according to the following algorithm:

1. 1. User 1 (named as “Seller” in fig. 2) creates Pending Order SELL for XX APL coins with rate YY.YY ETH (or other available cryptocurrency).
2. The order is stored at OrderBook. XX APL including minimum fee (if the order will be cancelled) will be blocked at the account. It is impossible to execute any transaction if the account balance after the transaction becomes less than the blocked amount.
3. User 1 may cancel the order at any time if it has not executed yet.
4. If User 2 (named as “Buyer” in fig. 2) wants to buy APL for another cryptocurrency.
   To do this, User 2 creates an Instant Order that specifies the required parameters: the number of APLs that the User 2 is ready to buy and the rate at which the User 2 is ready to purchase APL. Crypto assets exchange will be executed in the presence of the appropriate counter-pending order.
5. Checking the availability of the necessary funds on the cryptocurrency wallet. Exchange transaction will be cancelled if there is not enough funds. If there is enough coins, the following steps are taken:
   - Transaction "CREATE_INSTANT_ORDER_BUY_APL_FOR_CRYPTO" with the corresponding parameters (amount in APL, rate) is created.
   - The required APL amount is sent to User 2 using the delayed transactions with hash verification. Transaction hashing is formed by Oracle Transaction Monitor for this order.
   - Asset transfer information is sent to external cryptocurrency blockchain.
   - The Transaction Monitor defines the transaction state at the external blockchain.

- After N confirmations (N is a configurable parameter) Transaction Monitor (on his own behalf) executes the transaction order (INSTANT_OREDR_APL_COMPLETE) with hash. The transaction of funds transfer is confirmed. Order Book Manager closes the Pending Order if it is executed 100%. In another case it sets the parameter filled (filling parameter for the order).

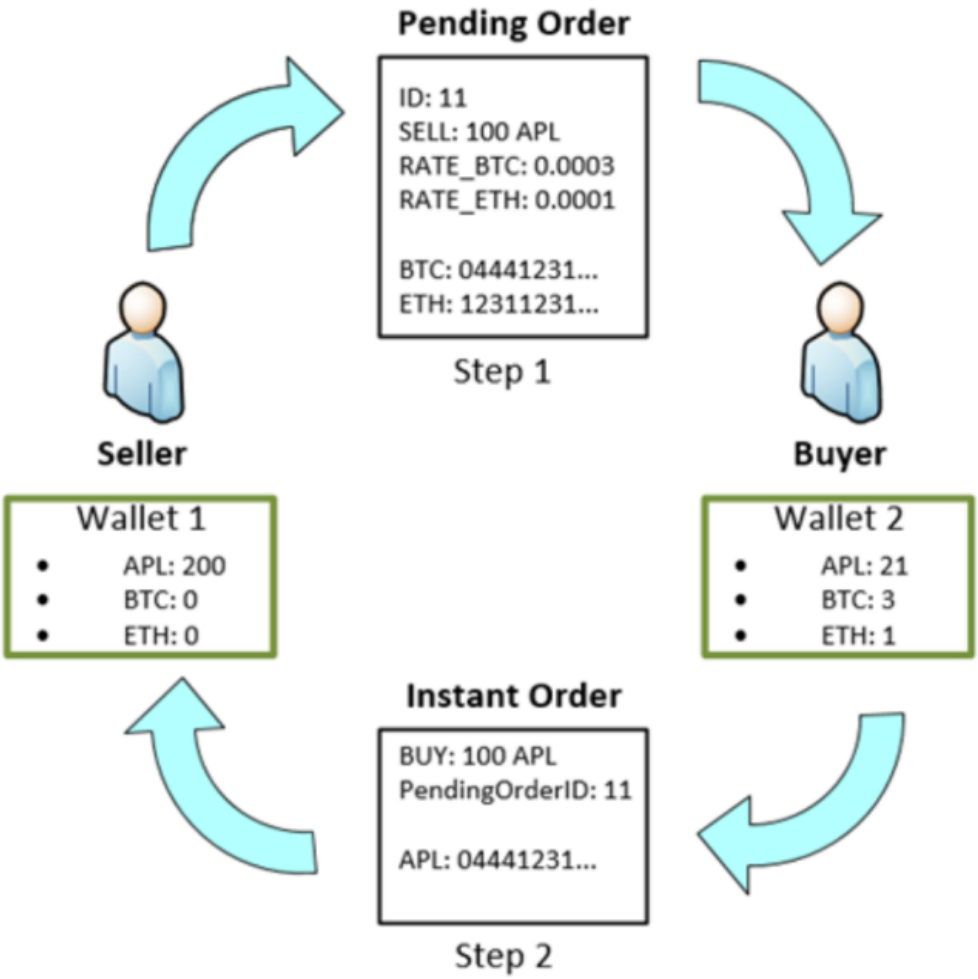


Figure 2. Scheme of buying the APL using another cryptocurrency.

Buying the APL using third-party cryptocurrency will be carried out according to the similar following algorithm:

1. User 1 wants to buy APL for another cryptocurrency. User 1 creates the Pending Order SELL for XX ETH (or another available cryptocurrency) with APL rate YY.YY.
   At the same time, the balance of User 1 account is checked whether enough to create an order. For the Order creation, commission is charged from the User 1 account for cancellation of the order.
2. User 2 creates an Instant Order that specifies the required parameters: The number of ETH (or another available cryptocurrency) that user wants to buy and the admired rate of needed currency if user wants to buy a cryptographic currency12 using an APL. In the presence of the appropriate counter-pending order exchange cryptocurrency:
   - Order Book Manager creates a transaction (INSTANT_ORDER_APL_EXECUTE) that executes the order. It transfers the required amount of APL to the User's 1 account using a delayed transaction with hash acknowledgment. Transaction hashing is formed by the Transaction Monitor.
   - FBTM transfers funds in a third-party blockchain or gives a signal to execute the corresponding smart contract (if a smart contract is used).
   - After the confirmation of the transaction by the Transaction monitor, the FOREIGN_BLOCKCHAIN_ORDER_COMPLETED transaction is executed, The hash of this transaction unlocks the Apollo transfer.

General logic scheme of the Apollo DEX operation is presented in fig. 3 (a, b). The scheme
is divided into 3 main steps, from the moment of creating an order to closing the contract,
including internal checks and atomic transfer.

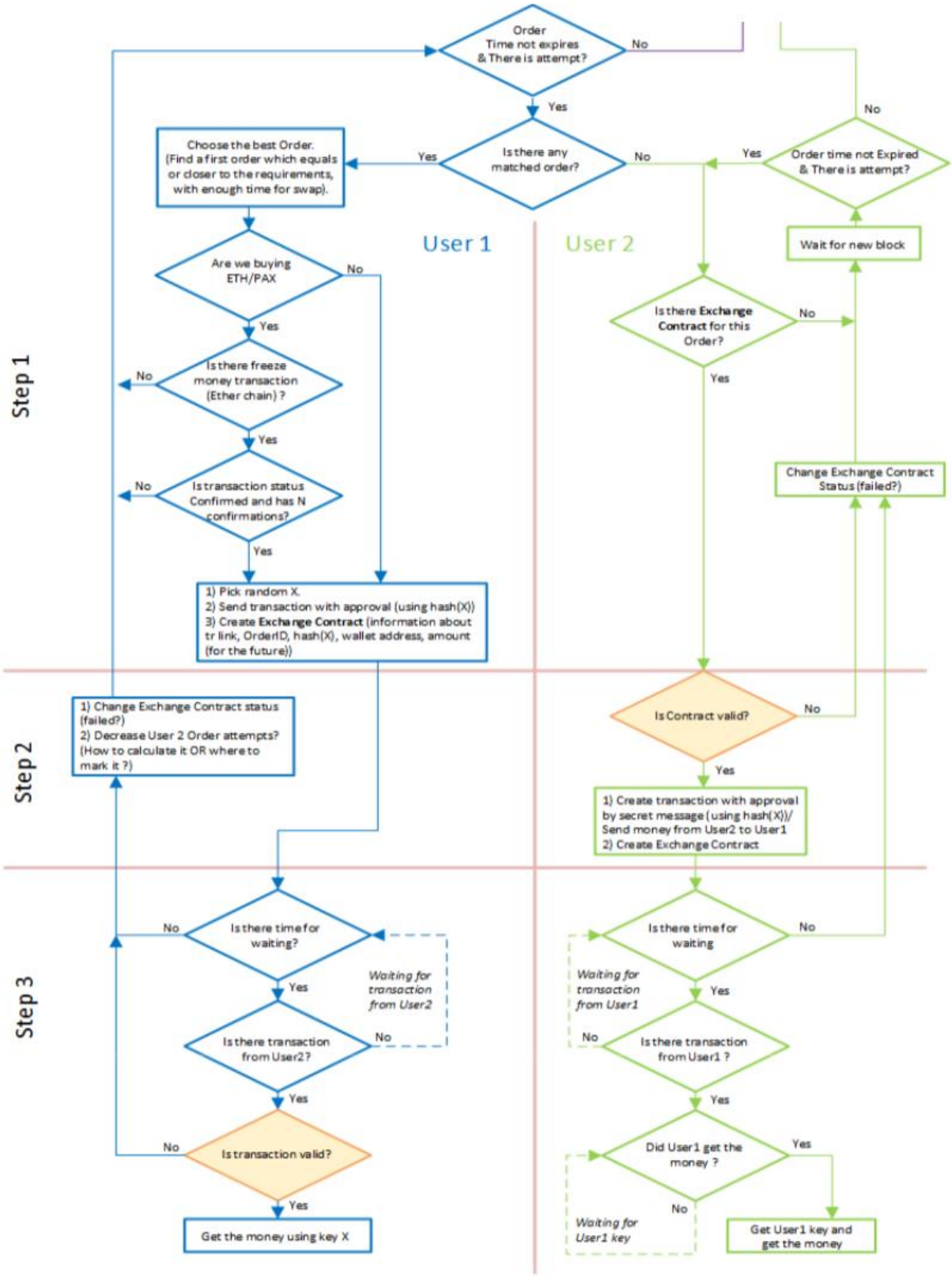


Figure 3 (a). Scheme of exchange transaction execution at the Apollo DEX

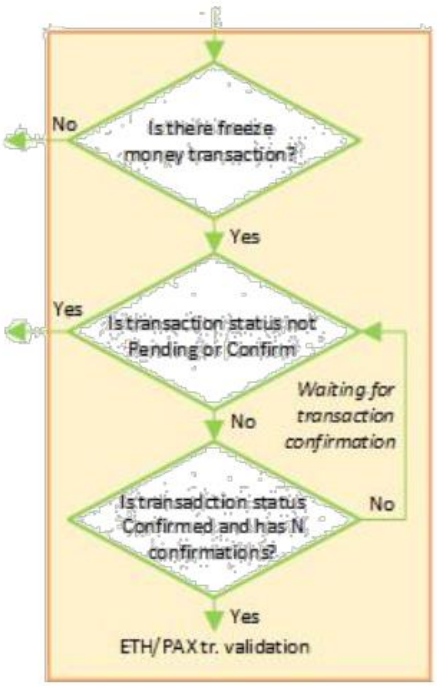


Figure 3 (b). Scheme of transaction validation.

Apollo Wallets

Apollo Wallet is available on various operating systems including Windows, Linux, OSX, Android, iOS, as well as users can access a web version of the wallet via any web browser. Apollo DEX is integrated into the Apollo wallets via DEX application programming interface (API).

The main functionality and principles of work of these types of wallets are identical. In this document the work of Apollo DEX using the web interface will be considered.

To access the Apollo DEX, the User needs to login in the Vault wallet. If the User does not have an account, it is necessary to sign up - to create a wallet (see “How to use Apollo DEX” section for details).

> *Note. The term "wallet" for Apollo should be understood as a set of wallets for various supported cryptocurrencies, united by one user account. Today “Apollo wallet” includes APL-, ETH- and PAX-wallet. The list of supported cryptocurrencies is expanding constantly*

There are two types of Apollo Wallet:

- Standard wallet
- Vault wallet.

Vault wallet has additional security mechanisms to protect unauthorized access to the user`s wallet. The Vault wallet uses an encrypted file with sensitive data. It is stored at the user's device in a secure place. In this case, the secret phrase is a key to decryption of this file, that contains the keys to wallets:

- Key to the Apollo DEX,
- Key to Ethereum,
- Key to PAX,
- The wallet key.

Apollo DEX services are available only for Users of Vault wallets. Therefore, consider the creation and usage of the Vault wallet. In the future, it is planned to provide access to the Apollo DEX for Users of Standard wallets.

Orders

There are following types of orders at the Apollo DEX:

- Pending Order. It is used as a posted offer on the exchange of assets at the specified rate. A pending order can be created or cancelled (deleted) by the User.
- Instant (Market) order. Instant order is the response of the User 2 to the selected pending order.

Today Apollo DEX operates with APL, ETH and PAX crypto assets. From this point of view, the Apollo DEX orders can be classified as follows:
BUY | APL-ETH (buy APL for ETH),16
SELL | APL-ETH (sell APL for ETH),
BUY | APL-PAX (buy APL for PAX),
SELL | APL-PAX (sell APL for PAX).
During the Apollo DEX development the list of available cryptocurrencies for the exchange will be significantly expanded.

UI Description

In this document, the Apollo DEX using the web interface, as declared above, will be considered. Note, that the main functionality and work principles of all Apollo wallets types are identical.

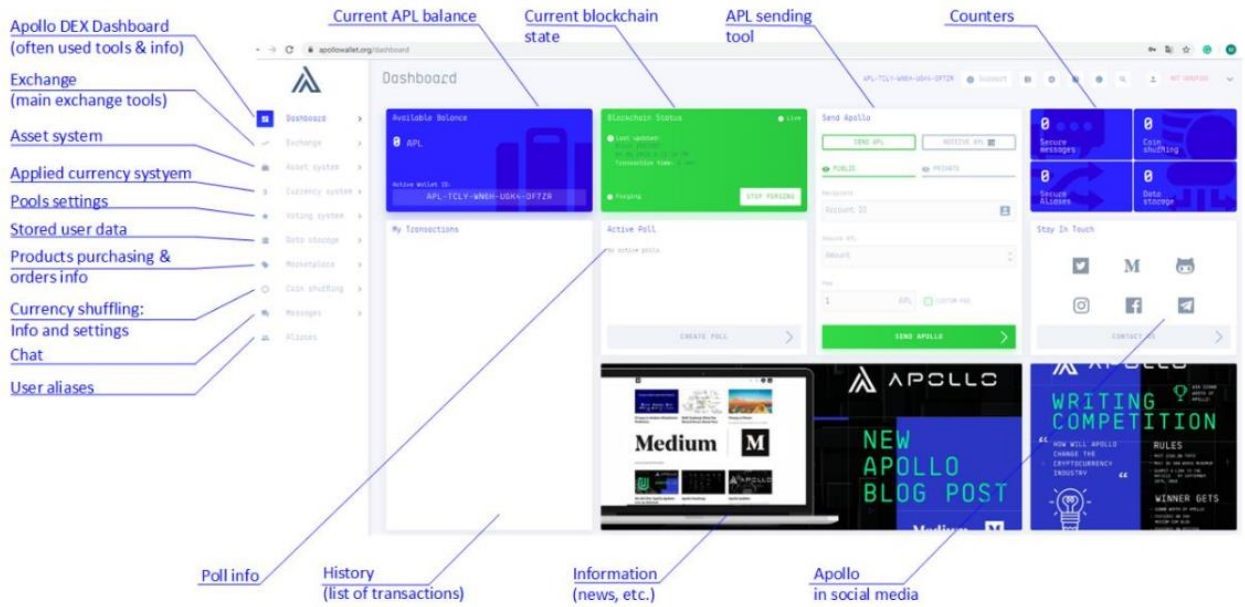


Figure 4. Dashboard page of Apollo DEX: view and main functioning elements.

Access to the wallet and Apollo trading tools is provided after the successful User authorization only. The login procedure is described in the "How to use Apollo DEX“ section below.

After successful login, the User will be redirected to the Dashboard web page.17 The Dashboard page allows the User to get basic information about his/her crypto assets (see fig. 4). The dashboard provides the following tools:

- Information on the current APL balance of the wallet;
- Transaction History;
- Event counters;
- Active Polls (the ability to create a new one);
- Information about the current state of Apollo blockchain (the ability to configure forging);
- A tool (webform) for sending APL;
- General information (news, etc.).

To navigate through the Apollo wallet/exchange web pages a user should use the main menu, that is placed on the left side of each page.

The main Apollo user tools for crypto assets trading are presented on the https://apollowallet.org/exchange web page (see fig. 5 for details).

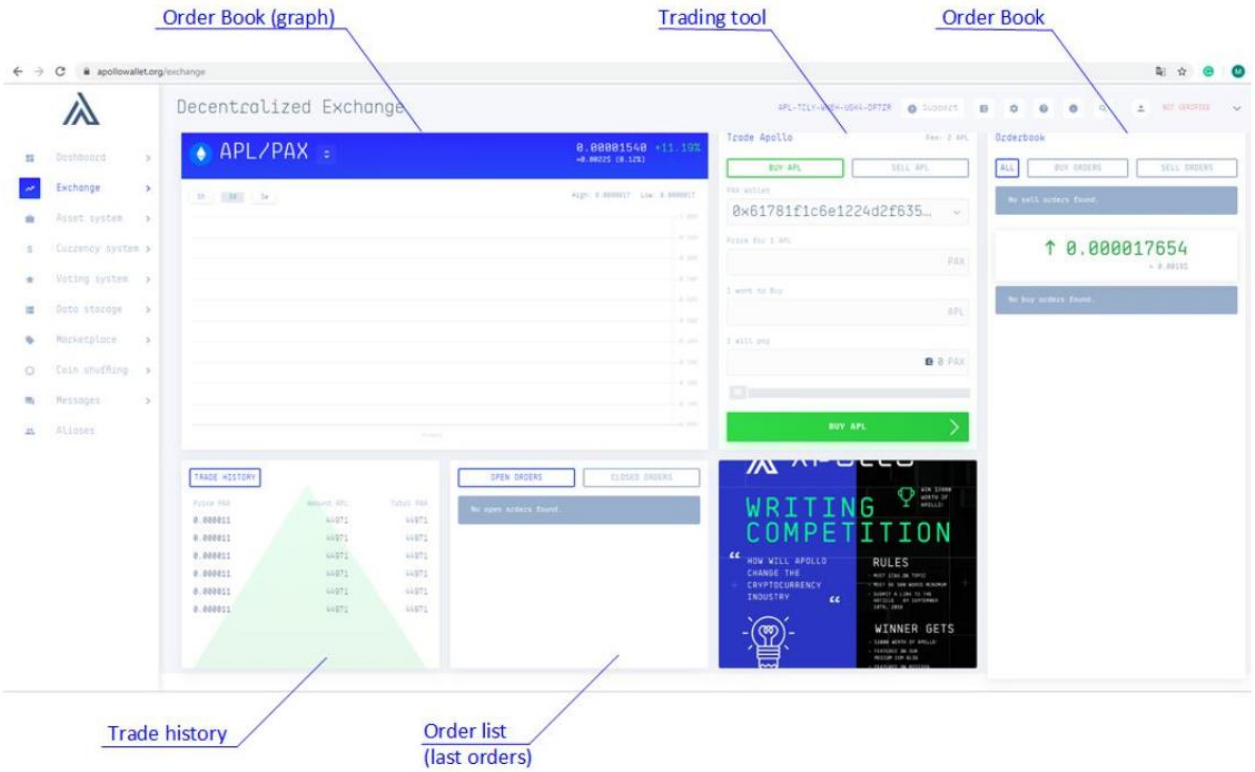


Figure 5. Exchange/Dashboard page of Apollo DEX: view and main functioning elements.

The Exchange web page allows to do the following:
• Tracking the dynamics of changes in the crypto assets ratio;
• Creating pending orders or cancel existing orders;
• Viewing a list of available pending orders;
• Creating an instant order for a suitable pending order;
• Viewing a list of own pending orders;
• Viewing own trading history (list of completed contracts).
How to execute main trading operations at the Apollo DEX is described in the "How to use
Apollo DEX" section below.

How to use Apollo Dex

To access the Apollo DEX, logging in the vault wallet is needed. Hereinafter the Apollo vault wallet and Apollo DEX services through the web interface will be considered.

Login

To login to the Apollo web wallet, it is necessary to visit https://apollowallet.org/login page first. There are two possible options how to login to Apollo Wallet:

- With account ID. User can access any account via ID. However, any transaction will require entering the secret phrase. That's why login using secret phrase is more convenient.
- With a secret phrase. User will access his/her wallet already authorized if the secret phrase is used for the login.
  Important: Login with the secret phrase only works for the standard wallets. If User has Apollo Vault wallet, it is needed to enter the account ID first, and then enter the secret phrase.

Import/export the wallet

User can import the Vault wallet in one of the following ways:

- Secret key. The secret key is a string that was used in the previous version of the Apollo wallet. At the moment, login with secret keys can be used by those Users who still have them saved. After the login, the wallet is automatically upgraded to the new version that uses a secret file with several keys in it, including Ethereum and PAX.
- Secret file. In the current version of the Apollo web interface, it is possible to import the Vault wallet by uploading the corresponding wallet file.

To logging in with the secret file the following steps should be completed:

1. Visit the https://apollowallet.org/login web page.
2. Click "Import Vault Wallet".
3. Click "SECRET FILE".
4. Enter a secret phrase.
5. Upload the wallet file.
6. Click "RESTORE ACCOUNT".

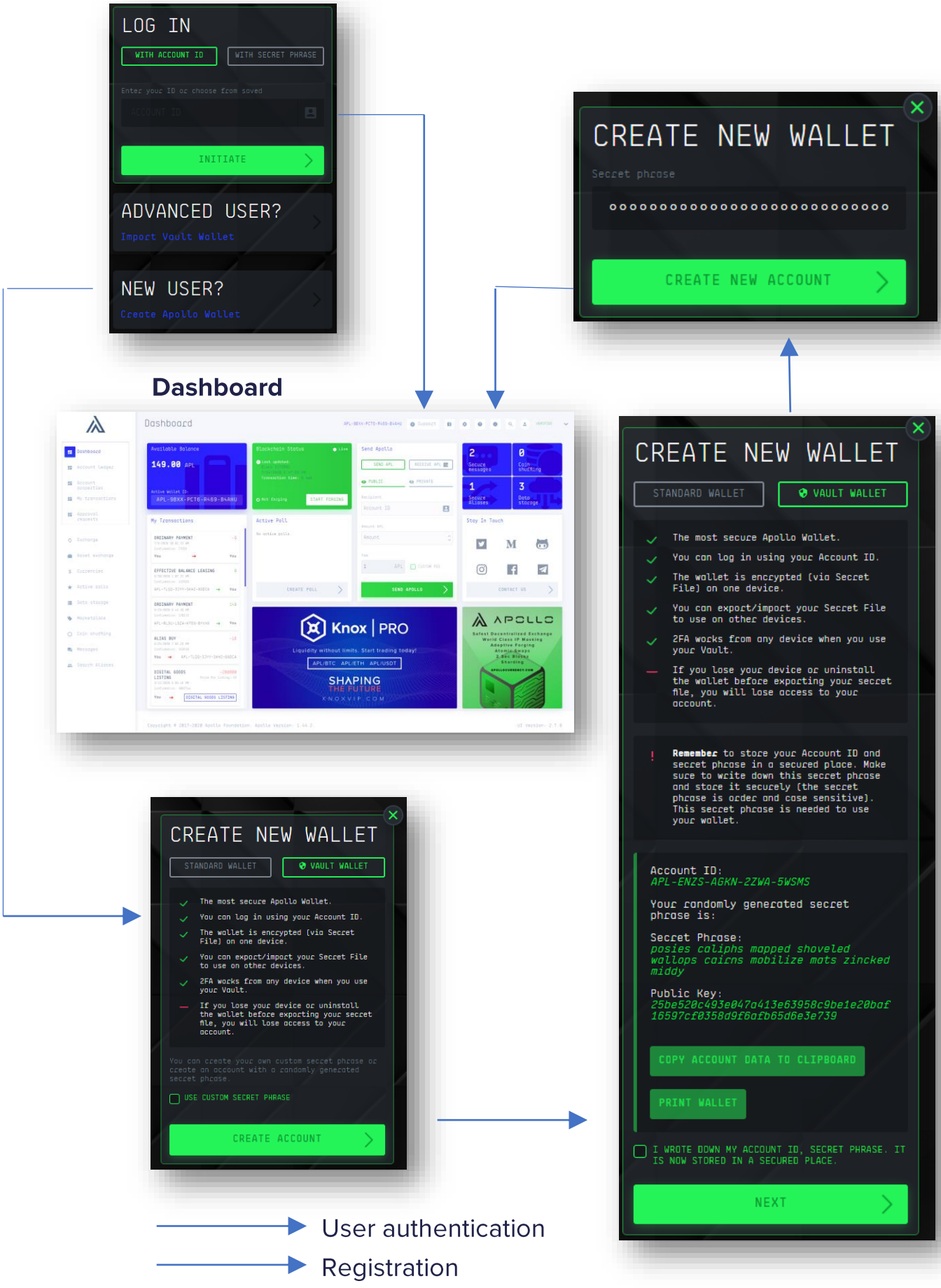


Figure 6. Scheme of logging in and vault wallet creating

Wallet creation

To create a Vault wallet it is necessary to do the following steps (see fig. 6 for details):

1. Visit the https://apollowallet.org/login web page.
2. Click “Create Apollo Wallet”.
3. Choose “VAULT WALLET” as wallet type.
4. (Optional) to make it easier to access the wallet in the future, specify the secret phrase. This phrase will be used for authorization instead of the wallet key, which might be difficult to memorize. Also, the secret phrase will be the key for the wallet file, which will include the key for user`s account. If someone steals a wallet file, it will be impossible to access it without this secret phrase, and vice versa.
5. Select the “USE CUSTOM SECRET PHRASE” checkbox, and then enter the secret phrase in the corresponding text field.
6. Click “CREATE ACCOUNT”. Important: Make sure to save the account ID and secret phrase. It cannot be restored.
7. Click “Next”.
8. Enter the secret phrase, and click “CREATE NEW ACCOUNT”. After that User will be redirected to the Apollo wallet Dashboard page.

Depositing

"Depositing" term means sending ETH to the Apollo wallet.

To deposit the Apollo wallet, the following steps are needed:

1. In the MetaMask account, click "Send".
2. Specify the Apollo ETH wallet address and amount of Ethereum cryptocurrency in the "Send ETH" dialog window.
   The actual Apollo wallet address for the ETH currency is presented at the “Exchange -> Wallets” web page.
3. Click "Send" again.

After the transaction is processed, the ETH will appear in the specified Apollo wallet.

Note. *The Apollo wallet may be replenished by ETH in any other way. Only the actual Apollo wallet address for the ETH currency is needed. MetaMask usage is just one of the possible ways of depositing*

Trading

To trade (exchange APL for another crypto asset), User must perform the following steps*:

1. 1. Go to the “Exchange -> Dashboard” page using the main menu for website navigation.
2. 2. In the “Trade Apollo” section (see fig. 7), choose “BUY APL” or “SELL APL” order type as needed.
3. Select the ETH wallet that will be used for this transaction.
4. Specify a price for 1 APL or ETH.
5. Specify the amount of desired cryptocurrency. Total receiving value (upon sale) or the payment size (upon purchase) of the cryptocurrency will be calculated automatically.
   It is possible to set the amount of selling transaction as a percent of the total wallet amount (for “SELL APL” transaction only).
6. Click “BUY APL” (or “SELL APL”) button at the end of the “Trade Apollo” webform.
   The corresponding order will be generated and added to the Orderbook automatically.

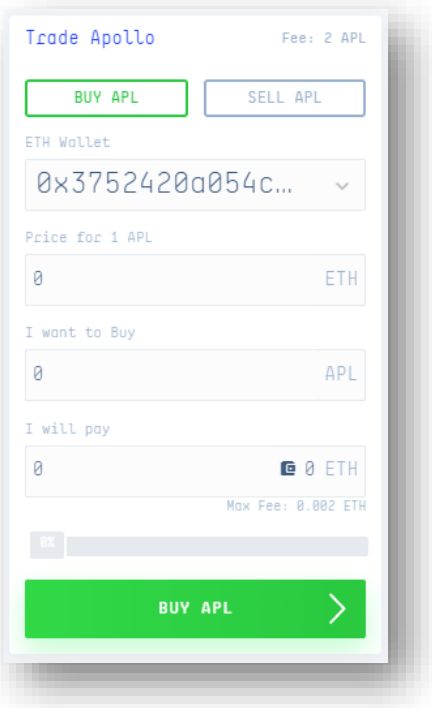


Figure 7. Apollo DEX trading web form.

Order Cancellation

An open order can be cancelled by the owner (a User who create it) in the following cases:

- Before the end of the order lifetime (24h by default) or
- Before an order is closed.

To cancel the pending order User must perform the following actions:

1. Go to the “Exchange -> Dashboard” web page.
2. In the OPEN ORDERS section, click the needed order.
3. Click CANCEL.

Withdrawls

Withdrawal is a crypto assets transfer from the Apollo wallets. This is the opposite of depositing (as described above). The following steps are needed (see fig. 8) to withdraw cryptocurrency from the Apollo wallets:

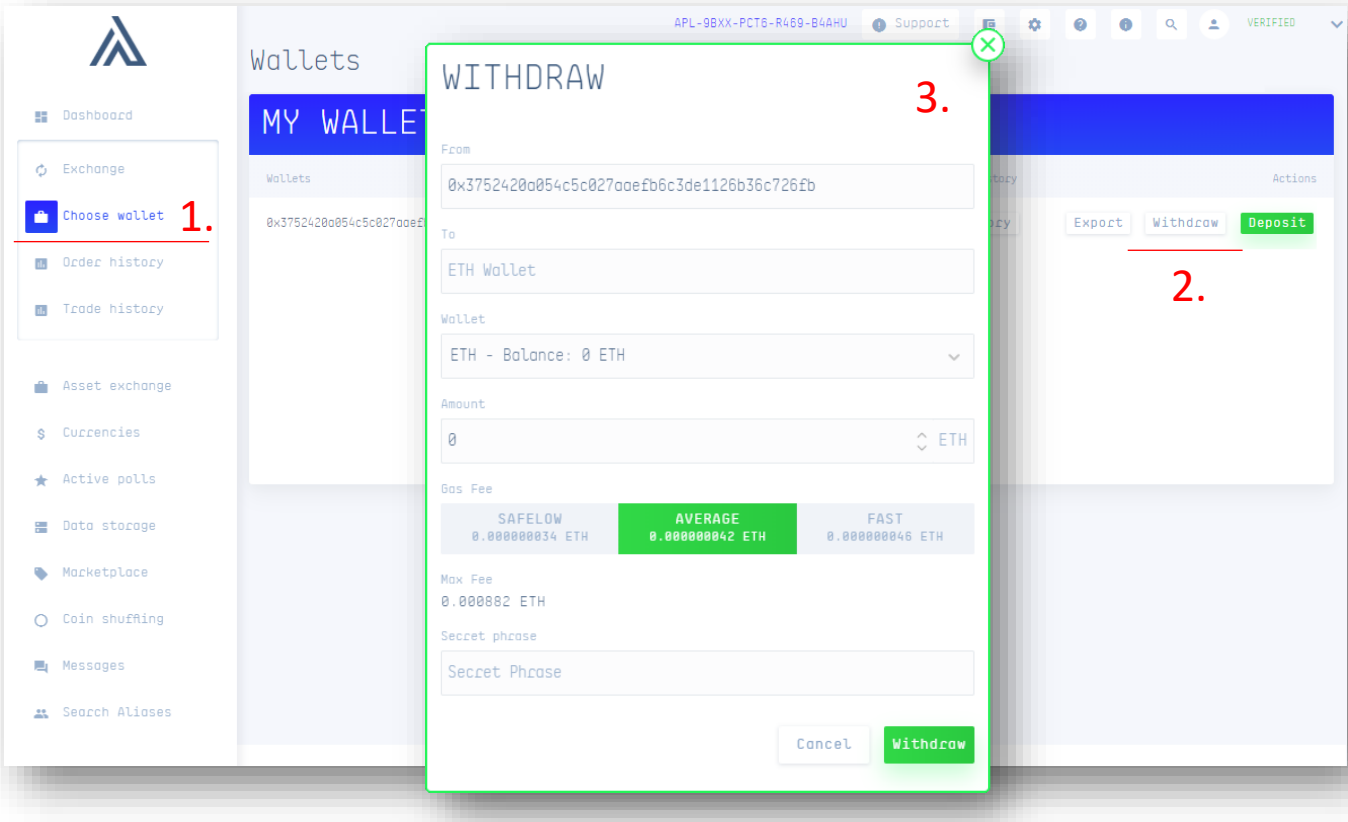


Figure 8. Scheme of crypto asset withdrawing.

1. Go to the “Exchange -> Wallets” web page using the main menu for site navigation.
2. Click the “Withdraw” button opposite the needed wallet.
3. Set the following data to the “WITHDRAW” webform (see fig. 8):
    - Address of wallet for Withdrawing (will be specified automatically);
    - Address of target Wallet;
    - Type of target wallet;
    - Amount of transaction.
4. Choose a gas fee that influences the transaction speed (the higher the commission, the faster the transaction).
5. Click the “Withdraw” button to transfer the funds or “Cancel” button for cancellation.

APL sending

The APL transfer from a user wallet to another wallet can be made using the "Send Apollo" webform (see fig. 9).

To transfer APL cryptocurrency, it is necessary to perform the following steps:

1. Go to the Dashboard page.
2. Select the type of transaction - click the "SEND APL" button (it is chosen by default).

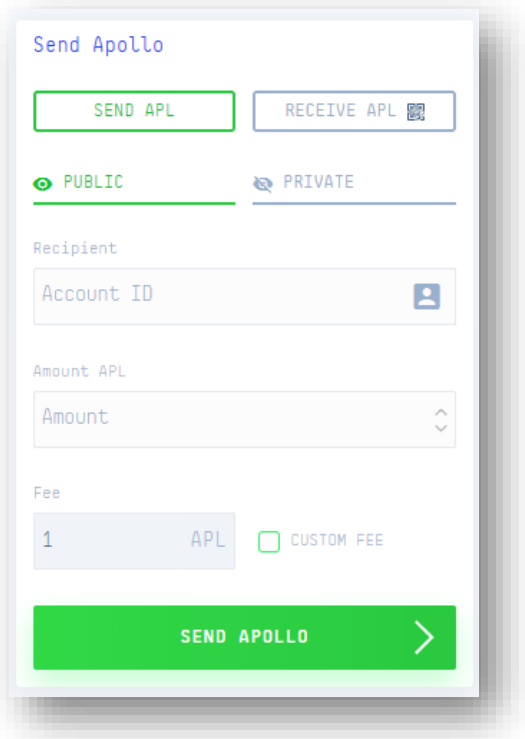


Figure 9. Webform for APL sending.

1. Specify the type of transaction: private or public.
2. Fill in the following data fields at the "Send Apollo" webform:
   - Recipient wallet ID;26
   - Transaction amount;
   - Fee (standard or custom value).
   - Click the "SEND APOLLO" button.

TradingView©

For the users' convenience, the visualization of the current state of the market (see fig. 10) will be implemented. It will be a valuable tool for traders and ordinary users.

Figure 10. TradingView graph and tools.

Visualization trading graph will be based on the commercial TradingView© library (see official site https://www.tradingview.com for details).

In general, the functionality of the trading view is dependent on exchange system functionality. The Apollo DEX trading view will be implemented as a market data graph with the following characteristics:

- The source of information - satisfied orders/offers (closed down both in terms of BUY and SALE)
- Charting technique - Japanese Candlesticks Visualizing.
- Information for depicting - typical for stock market "High Low Open Close" data. The following information will be presented:
  - the price at the beginning of the period (“Open”);
  - the price at the end of the period (“Close”);
  - the maximum price during the period (“High”);
  - the minimum price during the period apart from that, except prices (“Low”);
  - floating funds via period (the whole amount of currency) (“Vol” or “Volume”);
  - MA (Moving Average) Indicator (MA7, MA25). It’s the averages calculated back over the past x days. For example, MA 7 is calculating the average price for now and the past seven days.
- Specification of periods (typical for any tradingview) - 1, 5, 15, 30, 60, 240, 1440 min and week.
- All the necessary indicators should be available (depending on the components used). Usually, such indicators are calculated from the candlesticks.

Order matching system

The order matching system allows to find mutually complementary orders and carry out a transaction. The order matching system finds suitable pairs for closing orders (offers/offers) for the purchase or sale of foreign cryptocurrency.

The order matching system has 2 main goals:

1. Contract execution with the maximum benefit for all participants of the contract. The system offers the User a counter order that exactly matches the conditions or order with the most similar conditions in the direction of benefit for the User*.
2. The fastest close of the orders and implementation of the contract. For this purpose, the order matching system searches for a suitable pair each time after creating a new order. If a suitable counter order exists**, it will be offered to the User immediately***.

**Note1. Currently, the order matching system ascertains a match in the case of FULL compliance of two orders. So, the values "Buy XXX for YYY" and "Sell YYY for XXX" in two verified orders must match exactly.*
In future releases of the Apollo DEX, the matching conditions will become more flexible. The order matching system will propose a counter order with a similar, but at a more profitable rate. Users will be able to sell crypto assets more expensive or to buy it cheaper if a more favourable counter order exists.

***Note2. A new order will be placed in the list of orders if a suitable counter order was not found by the Order matching system.*

****Note3. In future Apollo DEX releases, the possibility of concluding and executing a contract at offline mode will be included*

A unique feature of the Apollo order matching system is multi-matching mechanism. It ensures the correct and simultaneous matching of several orders (including complex orders for multiple exchanges). The multi-matching mechanism ensures the matching correctness and resistance to possible attacks. So, the order matching system implements contracts execution as quickly as possible, taking into account the maximum economic benefits for all participants and relies on the basic principles of the economy

Atomic Swap

The atomic swap technology ensures the execution of transactions between unrelated blockchains without the participation of intermediaries. At the same time, the transaction is ensured only under the condition that each of the parties fulfils its part of the contract.

Atomic swaps attract each payment party to a transaction with a contract, a separate contract for each blockchain. A contract with the terms of the transaction is created in the Apollo blockchain and the corresponding smart contract is created in the Ethereum blockchain.

The general algorithm of atomic swap usage can be represented using the following example (see fig. 11):

Two parties (User 1 and User 2) agree on the exchange. User 1 pays User 2 1 APL, and User 2 pays User 1 10 ETH. The parties agreed to pay the blockchain commissions on their own.

Step 1. User 1 sends to User 2 an AA address in the Ethereum blockchain, User 2 transfers User 1 an AB address in the Apollo blockchain.

Step 2. User 1 generates the secret key K and its one-way hash KX. User 1 creates a transaction (smart contract) TA with a cost of 1 APL. TA falls into the Apollo blockchain at the time of BTA.

Step 3. User 1 informs User 2 of the details of the TA, from which User 2 learns the KX hash, and also checks the contract.

Step 4. If everything is in order, User 2 creates a TB transaction (smart contract) worth 10 ETH. TB falls into the Ethereum blockchain at the time of VTB. User 2 informs User 1 of the TB details.

Step 5. User 1 checks the TB contract. User 1 makes sure that the “Recipient address” is set to his/her ETH address, which was generated by User 1 in the first step and sent to User 2. The cost of the contract must correspond to the original agreement, that is, have a value of 10 ETH. The value of the “Secret hash” field must be equal to the hash of the KX secret key that User 1 sent to User 2 in step 2. It is also necessary that the TB contract is still valid, that is, the Locktime set should allow the exchange to complete.

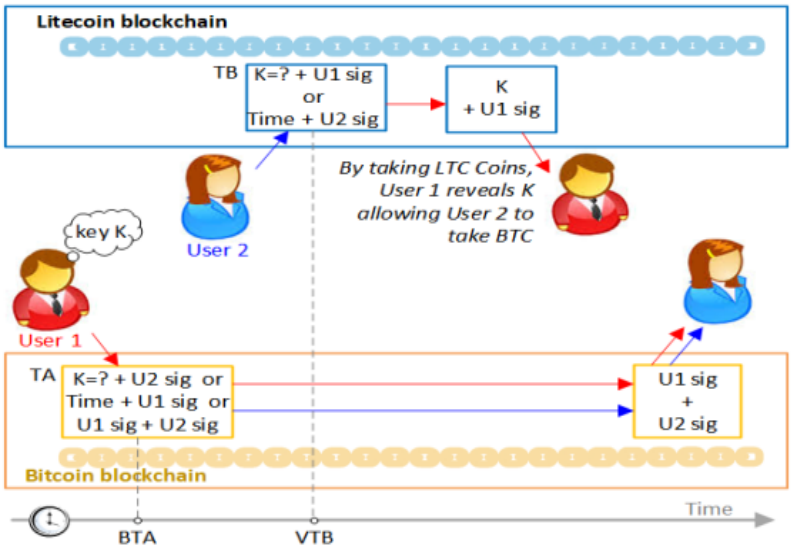


Figure 11. General scheme of the Atomic swap procedure.

Where:

AA is a new temporary wallet address of User 1 for expected payment (ETH in the example above).

AB is a new temporary wallet address of User 2 for expected payment (APL in the example above).

K is a secret key of User 1.

KX is one-way hash of secret key K.

TA is a payment transaction from User 1 to User 2 (APL transfer in the example above).

TB is a payment transaction from User 2 to User 1 (ETH transfer in the example above).

BTA is the date/time of creation the transaction TA by User 1.

VTB is the date/time of creation the transaction TB by User 2.

If everything is in order, User 1 transfers 10 ETH from the TB transaction to another address of his wallet. By this action, User 1 tells User 2 the key K. Now that User 2 has learned the key. So, User 2 has access to the transaction TA. User 1 transfers 1 APL from transaction TA to another address of his wallet. This completes the exchange process.

Terms of use of funds on the TA contract: User 2 can spend the money if learns the key K, in the period (BTA, BTA + 48 hours). After the BTA + 48 hours, User 1 can also spend unused funds.

Terms of use of funds on a TB contract: User 1 can spend money using his key K in a period (VTB, VTB + 24 hours). After the VTB + 24 hours, User 2 can also return unused funds.

As a result, User 1 received 10 ETH minus ETH transaction fees, User 1 spent 1 APL plus APL transaction fee. User 2 has 1 APL minus APL transaction fee. After the transaction, User 1 spent 10 ETH plus ETH transaction fee.

Fees

Today the Apollo DEX sets the minimal fees for all actions using the Apollo DEX. This makes it economically attractive in comparison with existing competitors.

Action	Fee on Apollo Exchange	Notes
Deposit	0	This values mean a zero fee action into the Apollo network. The transaction fee of the Ethereum network is charged when depositing money on the Apollo ETH wallet as well as withdrawing money from the Apollo ETH wallet.
Withdrawal	0	
Order creation (Buy/Sell)	2 APL	
Order cancellation	1 APL	
Contract (transaction) creation	2 APL	
Currency transfer	3 APL	These values mean a fee for transferring APL funds to the Apollo network. External blockchain (Ethereum) fees must also be paid for these actions. These fees for external currency (ETH) transfer is defined by the corresponding external blockchain (Ethereum).
Transfer Confirmation	2 APL	

Table 1. Fees values for various operations.

Note. The size of the commission on the Ethereum network is not constant. It depends on the level of the network load at transaction time.
The Apollo system integrated tools to determine the current state of the Ethereum network. This allows informing Users about the approximate amount of fees for ETH transactions.
Declared fees may be changed to provide the stability, security, and convenience of the Apollo DEX.

Integration with Apollo project

To integrate the DEX into the Apollo platform, the Apollo blockchain has been significantly upgraded. New key entities have been added, such as “Order”, “Contract” and “Smart contract”.

“Order” means an instruction for the Apollo DEX to buy or sell a digital asset. An order is formed by the Apollo DEX at the user's request for the sale or purchase of a crypto asset.

“Contract” is a digital document that contains a comprehensive list of obligations of the parties. The contract is formed in the case of finding two mutually corresponding orders.

The contract is recorded to the Apollo blockchain. The authenticity and integrity of the contract are ensured by the digital signature.

“Smart contract” is a computer protocol (executable source code) intended to full execution of contract conditions by both parties. The smart contract is created at the same time when contract created (see "Contract" above). The smart contract is stored at the Ethereum blockchain. A smart contract key entity for transactions synchronization between different blockchains (see "Atomic swap" section for details).

ETH integration

Apollo own ETH nodes were deployed for the Apollo interaction (and Apollo DEX correspondingly) with the Ethereum network. Integration with the Ethereum network is carried out using Apollo`s own Ethereum nodes, as declared above. Such interaction with Ethereum allows to do the following:

- carry out operations with the ETH (funds transfer, exchange);
- perform operations with ERC-20 tokens, like PAX (funds transfer, exchange);
- check the balance of the specified cryptocurrencies and tokens based on ERC-20 standard;
- use ETH smart contract to place contracts and conclude transactions on DEX.

Note: *that synchronous execution of transaction between the Apollo blockchain and Ethereum blockchain (or any other external blockchain) is provided by the implemented Atomic swap mechanism (see "Atomic swap" section for details). Apollo's interaction with ETH is designed to make all operations and transactions as fast and reliable as possible. At the same time, commissions, and, as a consequence, user costs in APL are minimized.*

In the near future, it is planned to create the BTC-to-APL and APL-to-BTC exchange at the Apollo DEX. To synchronize payments with Bitcoin blockchain, the Hashed TimeLocked Contract (HTLC) will be implemented. HTLC is a scheme of cryptographic confirmation of the legality of the actions of participants, while the actions themselves are separated in time. As the name implies, this tool combines two mechanisms for blocking the exit of a transaction: by time (time lock) and by a secret number, the hash of which is recorded in the blockchain (hash lock). HTLC ensures the reversibility of the operation when one of the parties of the transaction does not perform the required part of the actions. HTLC is a proven tool that is used, in particular, in the Lightning network.

Further plans

Further development of Apollo DEX will be carried out in the following directions:

1. It is supposed to integrate Apollo with other cryptocurrency systems. Thus, the list of cryptocurrencies that is available in the Apollo DEX user account will be significantly expanded (currently, only the APL, ETH, PAX are available). As a result, new trading pairs of crypto assets will be available at Apollo DEX.
2. According to the plan of refactoring of the Apollo platform, Apollo software will migrate to a truly modular architecture. As a result, the Apollo software will become a flexible kit for quickly and conveniently build any desired software configuration depending on current tasks and needs. Since the Apollo DEX is a part of the Apollo platform, it will be separated as a single module that is interacted with the Apollo platform.
   Since the Apollo DEX is a separate software module, it may not be included in the build or replaced with another software module by User needs. This can be a critical moment when deploying a node to devices with limited hardware resources.
3. Also, the development of the Apollo DEX is planned in the field of increasing the users' convenience. It is planned to introduce new tools for trading, such as the announced TradingView, and some others.

Conclusion

The main aspects of the Apollo DEX cryptocurrency exchange like functionality, technical features, development prospects were considered. The Apollo team convinced that a combination of all these aspects will make the Apollo DEX a powerful and popular tool for traders, investors, crypto enthusiasts and ordinary Users.

APOLLO SHARDING

What is Sharding

To use a distributed ledger for online transactions (for example, as a payment system), the database should work very fast.
Sharding is a concept that's widely used in databases, to make them more efficient. A shard is a horizontal portion of a database, with each shard stored in a separate server instance or separate "database file" instance (as in the case considered below). This spreads the load and makes the database more efficient.
In this section, we will consider the sharding implementation for the Apollo blockchain network and its technical aspects.
The Apollo wallet application version 1.38.7 (or later) as software with sharding mode support for Apollo nodes is considered in the current document.

Sharding: role and benefits for blockchain

In the case of the blockchain as a special type of distributed ledger technology (DLT), the following aspects should be considered:

- Scalability issues,
- Latency issues,
- Low throughput.

Each blockchain node will have only a small part of the data on the blockchain, but not the entire information when sharding is implemented. Nodes with enabled sharding mode maintain information only on that shard in a shared manner, so within a shard, the decentralization is still taking place. However, the node doesn't load the information on the entire blockchain. Such an approach ensures the scalability of the project.
Sharding implementation makes blockchain more scalable with higher transaction throughput.
Apollo is a fast developing blockchain platform. Increasing the number of transactions leads to a constant increase in the Apollo database. This process goes faster with shorter block times. So the database could become enormously big after some observable time.

Blockchain by its nature requires trusted access to ledger database from the very beginning to the present in order to be able to verify every fact/transaction. But this is not a task that is performed every time.
To carry out transactions in a distributed network, as a rule, a complete database is not required. Just a small part of the database is needed for this.
So, the main idea of sharding features of the Apollo blockchain consists of the following parts:

1. Split ledger database into time-based (or blockchain height based) chunks. This split can be made on any number of nodes (zero or all) in consensus. For instance, each node "knows" that split should be done on each 750 000 confirmed blocks of acknowledgment.
2. Add information to each chunk/shard that allows blockchain operations without queries to the full ledger database.
3. Optimize blockchain processing code to use the last chunk for most operations.
4. Make each chunk "trusted" by hashing it's summary tables.
5. Make full blockchain downloading optional/delayed. Download the latest chunk of the ledger and start processing

Such an approach will give the following benefits for Apollo distributed network:
1. Faster processing;
2. Faster node startup time;

Fewer resources (memory or disk consumption) for transaction and block processing.

Shard Creation

Any node can be launched in one of three following sharding modes (see Fig. 1):
Start node for downloading data, keeping full blockchain inside only one full main database. A full node without shards.
Starts node for downloading data, keeping full blockchain inside several shards + main database. Full node with sharded databases.

Starts node for downloading data using the latest available ‘shard archive’, keeping the next (after latest shard) blockchain data in several shard(s) + main database (or inside single main db file). Limited data on the node is available because all previous data (before imported shard archive) is not available on that node.

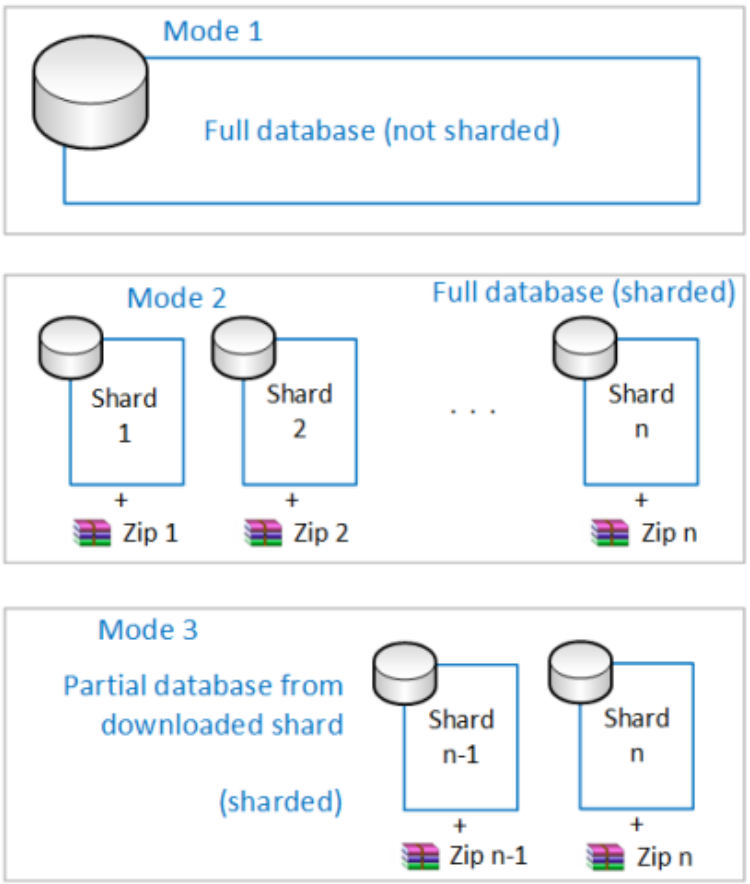


Figure 1. Scheme of data exchange during the shard creation procedure

Next shard creation is possible only for nodes with enabled sharding (sharding modes 2 and 3 only).
For node operation, the formation of a new shard implies the following data transfer procedures:

1. Makes sure that the blockchain height is greater than shard height for enough blocks which make rollbacks impossible.
   The node will start the shard formation at the following height:

   ShardHeight = (CurrentHeight - MaxRollback) / ShardFrequency,

   where:
   ShardHeight is blockchain height when node will start the shard creation;
   CurrentHeight is the current height of the blockchain;

   MaxRollback is the maximum number of consecutive blocks after which rollback of transactions is impossible. Currently, theMaxRollback is set to 21,000.
   ShardFrequency is periodicity (number of blocks) of shards formation. This parameter can be set for each node (see the “Launching the Apollo application with/without sharding option” chapter of the current document for details).

2. Stops current blockchain processing.
3. Locks all external P2P communications.
4. Backups the old database (or leaves as is in the case of SQL server).

Note that steps 2-4 are executed before each shard formation procedure.

5. Checks if the previous sharding process has started but not finished yet. Recovers previous unfinished shard attempt and continues it.
6. Creates a new shard database with the necessary structure. It is shard_NNN database where NNN is the number of the new shard.
7. If it is shard 1 then start from genesis block with empty shard_NNN database. If shard number >1 then gets previous shard super-block and fill database by importing next blockchain-related data.
8. Starts moving blockchain data from the main database to the new shard and fills the shard_NNN database with data as it goes by normal "scan" of blockchain records.

9. When the data transfer is finished, fills the main database with required additional data.
10. Exports the shard_NNN snapshot data to CSV files with well-defined pursuable header lines. These header lines of each file must contain data to identify and verify exported data.
    The data structure of the shard is described in the “Data in the shard archive” chapter of the current document.
11. When all data are exported, creates a zip file with date value equal to shard number (1L, 2L, ...).
12. Calculates SHA-256 hash of resulting zip file, timestamp, other control and technical parameters of this created shard. This hash value is used to compare shards among all nodes at the downloading process. This value should be exactly the same for all nodes with the right blockchain.

    Sample of computed data for data shard is presented below:


```
{"shardId":12,"shardHash":"5fdcb0530cfaa0e8ad24c963789b41b9
ac52e3b0a023d4c58679acf9d897f85b","shardState":100,"shardHe
ight":576000,"coreZipHash":"f10158dcbef3857b549d961d6de742c
f81e4b46d40fdf4aa0bb3cd5cea6762b7","prunableZipHash":null,"
generatorIds":"[9211698109297098287, 9211698109297098287, -
4010905505099140128]","blockTimeouts":"[0, 0,
0]","blockTimestamps":"[53297484, 53297480, 53297475]"},
```


13. Stores this data safely and stays ready for uploading to other peers.
14. Attaches shard_NNN database as additional to the main blockchain database.
15. When finished, unlock external P2P communications and goes to regular mode.

Data in the shard archive

The CSV file with shard data will contain all necessary snapshot data from the Apollo database that corresponds to the current shard only. That corresponds to the data from (PreviousShardHeight till next ShardHeight blocks. The list of database tables is presented below (can be clarified later):

1. account_asset.csv
2. account_control_phasing.csv
3. account.csv
4. account_currency.csv
5. account_info.csv
6. account_property.csv
7. alias.csv
8. alias_offer.csv
9. ask_order.csv
10. 1asset.csv
11. asset_delete.csv
12. asset_dividend.csv
13. asset_transfer.csv
14. bid_order.csv
15. block.csv
16. block_index.csv
17. currency.csv
18. currency_supply.csv
19. currency_transfer.csv
20. dex_offer.csv
21. exchange.csv
22. exchange_request.csv
23. goods.csv
24. phasing_poll_result.csv
25. poll.csv
26. poll_result.csv
27. public_key.csv
28. purchase.csv
29. purchase_feedback.csv
30. purchase_public_feedback.csv
31. referenced_transaction.csv
32. shard.csv
33. shuffling.csv
34. shuffling_participant.csv
35. tag.csv
36. tagged_data_timestamp.csv
37. tagged_data_timestamp.csv

38. trade.csv
39. transaction_shard_index.csv

NOTE 1. Currently, when shard database exists it includes BLOCK and TRANSACTION tables mainly.

NOTE 2. To ensure the reliability and fault tolerance of the Apollo network, control nodes are created and launched. Control nodes store a complete shard set of the Apollo database.

Control Blocks

The node will start the shard formation at the height ShardHeight(see above). Every shard starts from the snapshot block.

Snapshot block - blocks that start next shard (like genesis block but start just next shard). After completion sharding procedure every new shard is started with the first snapshot block.

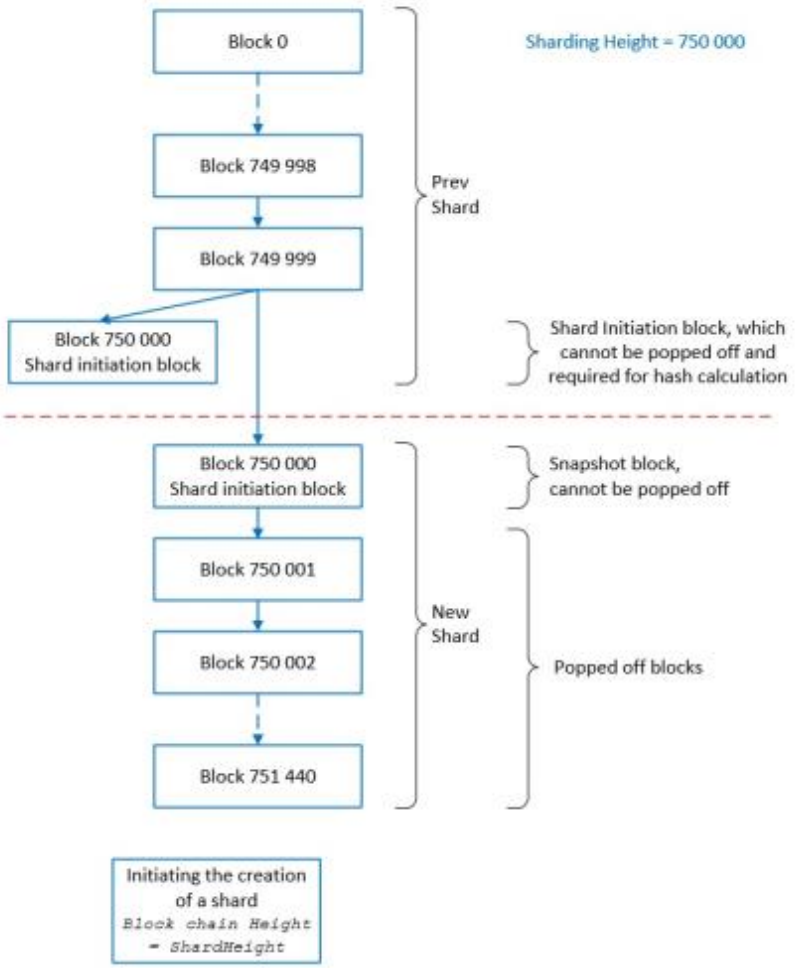


Figure 2. Scheme of blocks selection for shard formation. Sharding Height is set as 750 000.

The procedure of snapshot block formation includes the following steps (see fig. 2 for details):

1. When shard height has reached the system is ready to initiate sharding. The node saves its shard initiation block and waits until it becomes a block, which cannot be popped off (roll-backed). Also, the node should have an ability to pop Off the next block (right after the shard initiation block).
2. Then the node makes pop Off down to our shard initiation block (not including it) and saves all popped Off blocks (right after it) into temporary storage/tables. At this moment its previous shard is entirely completed. The node should calculate the hash of all blocks. Merkle tree is not required, because the node does not need to identify which block is not valid.
   Note, that the hash calculation procedure can take a lot of time. It defines by shard size (ShardFrequency parameter) and hardware nodes.
3. The node copies all necessary tables to the new shard DB and pushes the shard initiation block as our snapshot block.
4. When the snapshot block is pushed, the node pushes all previously popped Off blocks.
5. As a result, after the above items, the node can work with the new shard.

API related to sharding

To get a list of available shards from some Apollo node, it is possible to send the following GET (HTTP) request:

```
.../rest/shards
```

The requested node will return a list of available shards and their parameters as a response.
Example:
Request: `http://localhost:7876/rest/shards`
Response:


```
[{"shardId":1,"shardHash":"9370ce0ddb3648decdee2b94e3926093114a0939
bd08c1c3624567ee604c4ab9","shardState":100,"shardHeight":2250000,"c
oreZipHash":"7a3db9513e70e0ae5aaee08b23bc3035bc2862dd61c790980dea60
61e1d6f2e1","prunableZipHash":null,"generatorIds":"[-
1836812095473256537, 8638689226260340876,364975977827208006]","
blockTimeouts": "[6, 5, 9]","blockTimestamps":"[47545988, 47545978,
47545968]"}]
```


Note that the presented response includes the information about one shard. Generally, it may contain the data about several shards or may not have data at all.

Shard downloading and import

Any node can be started in mode (see above) for downloading from some shard archive. Usually, this is done once during the operation of the node. The algorithm for choosing the method of data loading by the node is presented in Fig. 3.

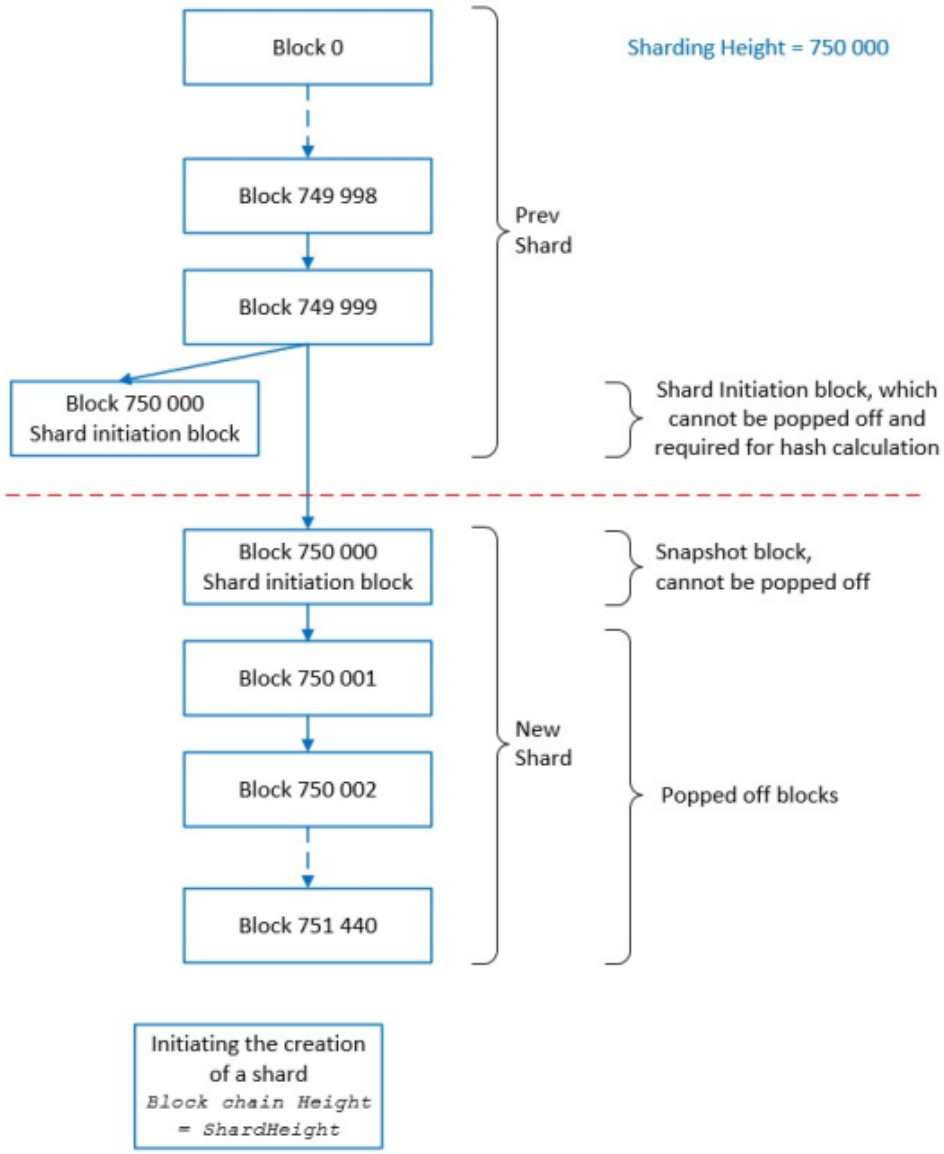


Figure 3. General scheme of data downloading during node startup.

NOTE 1. Shards downloading will be executed only if node sharding is switched on.

NOTE 2. Sharding is switched on by default in the Apollo wallet application. Latest special data are known as shard archive from 0-height till the last is available for download by default. To configure node settings, see chapter “Launching the Apollo application with/without sharding option” of the current document.

Features of the shards downloading procedure depend on the node settings, available peers, etc. In general, the shards loading procedure includes the following steps:

1. Connect as much peers as possible (from the well-known list that available on every node).
2. Randomly choose peer and try to connect.
3. If connection successful request shard related info.
4. Choose the next random peer and repeat 2-3 until there are enough peers. The number of peers should be 10% of known or at least 20.
5. Sort randomly connected peers by the latest shard block hash value. In a normal situation, almost all peers should give the same value of this hash. When the node has a small number (2-3) of peers with different hash values it means that those peers created shard incorrectly (or maybe on the fork), the node can ignore that peers. In case of a bigger number node has troubles with blockchain and should take further steps to investigate the situation. As a result in ordinary terms, the node has a set of peers with "right" shards. In case the correct number of trusted nodes with the latest trusted shard archive were not found/computed, node fallbacks into mode starting downloading from 0 height.
6. Get an array of CRC sums for the shard archive file from one of peers. This array should be identical for each "right" peer because the shard archive file is byte-by-byte the same.
7. Start downloading of shard archive file chunks from chosen peers in parallel until the entire file will be in place with the right chunks' CRC. Use random file access to write chunks in a single file.
8. Check the integrity by calculation of shard master hash.
9. Start the shard data import process (see below).

After a node has downloaded latest shard archive (as zip-file), it can import shard archive data to local database in the following way:

1. Unpack zip with CSV shard data to a temporary folder.
2. Zip should include snapshot block + transactions, block_index, transaction_index, shard table, operational derived tables.
3. Find derived table from DerivedTablesRegisty and retrieve a proper file.
4. Read CSV file as result set and import into specific tables all data sequentially.
5. Start normal node operation when shard archive importing is finished.

Launching the Apollo application with/without sharding option

A node (as an Apollo wallet application) can start working on one of the three following modes:

1. Download all data blocks (the entire database) without sharding, starting from the genesis block. Sharding is disabled.
2. Load the entire database in the form of shards, starting from the lowest height. Sharding is enabled.
3. Load only the latest shard archive*, sharding is enabled.

* This implies a network-accessible shard that is statistically reliable.
Sharding is switched on by default in the Apollo wallet application. All shards from 0- height till last available will be downloaded by default (mode 2 as declared above).
To configure node settings user can set the needed command line parameters when starts the Apollo wallet application via the command line (terminal).

Key	Description
--no-shards-create	Global value. Do not create shards even if it configured to do so.
--no-shards-import	Start from Genesis block, do not try to import last shard archive and continue working from it's height

Command line startup examples (service mode of the Apollo node):

```
>java -jar apl-exec-1.38.8.jar -s -d 3 --net 0 --no-shardsimport true
--no-shards-create true
```

This command runs the full node without sharding, starting from 0 height.

```
>java -jar apl-exec-1.38.8.jar -s -d 3 --net 0 --no-shardsimport true
```

This command runs the full node with sharding, starting from 0 height

```
>java -jar apl-exec-1.38.8.jar -s -d 3 --net 0 --no-shardsimport false
```

This command runs a limited data client node. It should try to download the latest trusted
shard archive and continue from it with next shards creation.

The sharding settings of the Apollo wallet application can be changed using the following configuration files of the application:

Parameter	Description	Notes
apl-blockchain.properties file		
apl.sharding.backupDb	Backup DB to zip before creating any new shard DB and deleting old data from main DB.	Bool (true or false)
apl.noshardimport	Disabling the latest shards archive search, download and importing. The node will download all database starting from genesis block.	Bool (true or false
apl.noshardcreate	Global parameter. Enable/disable splitting blockchain data into shard databases.	Bool (true or false)
chains.json file		
shardingSettings	BlockchainProperties is a structure that describes sharding procedure for the current height configuration	
enabled	Sharding ability, can be specified additionally and turn OFF sharding when it's ENABLED globally. It CAN"T turn ON (overwrite) global value though.	Bool (true or false) "true" = enabled "false" = disabled
frequency	Frequency of partitioning (sharding)	Integer number

Note that the keys to launching the Apollo wallet application have a higher priority over the corresponding parameters in the configuration files

As of July 14th 2020, the Apollo blockchain has successfully performed a shard at each of the below block counts.
The blockchain performs these shards at 750,000 block intervals, with the next shard due to take place at block 5,250,000.

Shard #	Blocks	Date of shard
Shard 5	750,000	Next Shard @ Block 5,250,000
Shard 4	750,000	30th Apr 2020 (UTC) @ Block 4,500,000
Shard 3	750,000	2nd Feb 2020 (UTC) @ Block 3,750,000
Shard 2	750,000	4rd Nov 2019 (UTC) @ Block 3,000,000
Shard 1	2,250,000	19th July 2019 (UTC) @ Block 2,250,000

REFACTORING REPORT

Introduction

The Apollo project is growing and developing constantly. But, like any construction work, the development of an IT project requires a reliable basis and a healthy ecosystem. The current document describes a number of enhancements to the Apollo platform, collectively named "Refactoring". At this stage, the refactoring has been finished, and all code improvement that was done prior to this point is called refactoring, whereas all future updates to the Apollo core to support new features will be called adaptation. All these enhancements are designed to ensure the harmonious development of the Apollo project in the future, its modernization and expansion.

The fast development of the market requires prompt development of new features to satisfy the growing needs of the market participants. From the experience that we gained in 2019, it was clear that successful movement further with the old codebase is almost impossible, so it should be replaced with a well structured and modular design. This step allows us to develop new features faster, as the core is based on modern technologies.

If we compare the state of the project at the beginning of 2019 and the current state, we'll see a massive leap from hard to extend and support entangled custom software towards a truly industrial modern software project.

Apollo's 2020 roadmap includes several innovations and unique features. For these ambitious plans to be implemented on time, the refactoring of the software platform was executed; otherwise, further development of the Apollo wouldn't be effective. Furthermore, for the successful development of Apollo in the future, the code base will be adapted when needed to support new features.

Modularity, as a part of the refactoring process, has been finished. It includes moving from a monolith code structure to a modular code. The dapp modularity that will bring constructor possibility to the Apollo project will be introduced after the launch of Apollo Smart Contracts.

Current State

The development of Apollo in 2018 by our small team showed serious limitations and difficulties related to the NXT code quality. In one sentence, simple feature implementation required significantly more time and effort than the average modern project usually requires. Refactoring was a necessary measure for the effective implementation of the project. According to it we've started the refactoring process. First of all, the project structure and architecture didn't allow to change software parts independently due to high code inter-connectivity and entanglement. This code is a classic example of the notorious “spaghetti code”. Secondly, the project is based on 20-years old approaches and technologies with a huge amount of plumbing code that could be replaced with simple annotations or just one statement using a modern approach.

To be more exact with project problems, here is a shortlist of approaches and technologies from the late 90s in the project before refactoring:

•

The script-based manual build system for the whole project that does not allow automated modular builds with CI/CD. But continuous integration is a “must-have” project structure element.

- Monolithic design without separation of even such parts as frontend and backend. This forces the whole project update on any change in independent parts, and other inconveniences in the development process.
- Wide usage of static code that makes unit testing impossible and creates additional complexity.
- “Spaghetti” interconnection of all code and code entanglement. These facts make code very fragile when a small change in one place causes unpredictable effects in several other places.
- “Manual” processing of pseudo-REST queries scattered over different parts that bloat code and makes it hard to understand and fix.
- JSON processing with field-by-field coding and a lot of hardcoded strings. This fact also makes code over-bloated and hard to change.
- Pure servlet-based request processing with manual Object instance creation for processing classes and a lot of hardcoded request mappings. This also makes any small change a difficult task.
- Huge classes (up to 10,000 lines) with a lot of plumbing code that is hard to read, understand, extend and support,
- Enormous usage of inner classes (up to 100) in the parent class that also makes code extremely hard to extend and support.
- Overused threading with pure synchronization all over the code. That causes instability and hard to find bugs.

- All other possible violations of object-oriented design and the SOLID principle that complicates and entangles code.hard.

All things above are tabooed 30 years ago by the computer science community because such an approach leads to an exponential growth of unnecessary code complexity, cost and time of development and causes problems in the project development and support as a result. So it is not effective to continue development in such a way and it is clear why the NXT project stuck.
The Apollo project must evolve, so all those drawbacks must be resolved to allow healthy project growth and development. We conventionally call this process “Refactoring”, but in reality, it is a massive change of source code, project structure, and infrastructure.

The high-level goals of the changes:
1. Stable and steady project development based on industrial development process models.
2. Cost and time effective usage of human resources, automation of routine tasks.
3. The possibility of effective development of new functionality.
4. The possibility to work on a project with a growing team and develop several parts in parallel.
5. To use best practices in development, modern tools and libraries.
6. To allow the project to catch up with the modern Java ecosystem and contemporary computing environments in general.

Therefore, work during the first quarter of 2019 and later was an effort to make the project closer to modern development practices and allow new features implementation on a modern basis. Firstly, we took organizational steps to build a healthy project development process and project infrastructure.

So following organizational steps were done:

1. Source code was split into several build modules. That allowed independent changes in modules and made the source code development process more effective and productive.
2. Modular Apache Maven-based builds created. This step allowed us to automate the build process with CI/CD infrastructure.
3. JavaScript GUI parts moved out of the main project to separate project. This step allowed us to have a separate independent UI development team.
4. Jenkins build server deployed and CI/CD tasks created for modules and installation artifacts (continuous integration infrastructure).

5. Artifactory repository server deployed for support of independent module and subsystems builds. Now we have stable builds of all project libraries such as our cryptographic library, utilities, UI, and others available to all developers. There's no need to download and build from source every time.
6. Automatic installer builds on CI/CD pipeline created for all supported platforms. Now we have sustainable builds of all artifacts needed for QA testing and for releases.
7. Development of source code was switched from a chaotic model to strict GitFlow procedures with well-defined source code branching, stabilization, and release rules.
8. Development, build, testing and release procedures were written, automated and followed by developers.
9. Testing plans and checklists are developed and followed. This is a big step to better software quality.
10. Additional testnets were created with well-defined purposes: stable code staging, development code testing, experimental code testing.

Secondly, the required changes to source code structure, build process and source code were accomplished:

- The main blockchain project is split into several build modules and is not monolithic anymore. This task required a lot of source code refactoring. This step stabilizes the interface of modules and allows independent fix or re-write of implementations according to the main principle of object-oriented design.
- Whenever possible static code is replaced with a service pattern. It is about 70 classes. This step allowed us to develop unit tests with modern testing frameworks.
- Huge classes refactored to 4-5 smaller classes. It is also about 70 classes. This step allowed us to develop more productively.
- Hundreds of inner classes moved outside of encapsulating classes to sub-packages. This step makes code more transparent and also allows us to develop tests.
- Tens of in-class threads streamlined and redefined in separate classes under thread scheduling service. This step minimizes random "Out of memory" failures and other random resource exhausting failures in runtime.
- Hundreds of static variables moved to the injectable property holder class. Now we have the possibility to configure applications in a clean and unified way.
- The dependency of packages normalized to the tree from circular. That step radically reduces unnecessary code complexity.
- Circular dependencies between classes inside of packages resolved. Now we can use dependency injection and also reduce unnecessary code complexity.

- Thousands of “isTestnet” statements and related code removed. Now we have exactly the same code with the same behavior on the main net and on test nets. This means that we test the same code that goes to production. As a result, we have a more stable code in releases.
- A default configuration and application data (Genesis accounts, JSON definitions, etc.) moved to application resources (into single jar file). This step reduces errors of misconfiguration and accidental changes in important application data.
- The main net and the test net resources are in a single jar file, so the same application can run with the main net and with any of test nets with a single command-line option. This step simplifies the testing process and reduces the time for testing.
- Location of databases and related files are standardized for server and user modes, which allow running the single installation with different test nets or with the main net. This step also simplifies the testing process and reduces the time for testing.
- Dependency injection container (CDI 2.0 standard) added to the project, most direct class Object instance creation replaced with dependency injection annotations. Now code is quite more manageable and transparent. This step reduces needs for “plumbing code” and speedups development.
- JAX-RS standard REST level added to the project. Now we can easily develop new API without a lot of buggy and time consuming “manual” coding.
- Request forwarding code added to support the seamless switch from old servlet-based “manual” request processing to JAX-RS standard.
- Open API OAS 3.0 (Swagger) added to support annotation-based API auto-documentation and testing. This step gives us an automatically generated API test web page instead of a time-consuming messy old test servlet. On the other hand, UI developers now have exact and precious documentation on API “on the fly” without any additional work on the backend.
- Java-JSON mapping (Jackson) added to replace “manual” JSON processing with automatic POJO mapping. This step radically reduces errors that could be made by “manual” JSON parsing into Java objects and the generation of JSON from objects. This feature speedups development and makes code more stable.
- REST API are-defined as POJO with OAS 3.0 documentation annotations. This feature allows more effective development of client software by UI team and third parties using a wide range of automation tools and programming languages.
- REST API moved to a separate module for pure integration testing. Integration tests that use API only help to reduce manual testing time and allows developers to check changes without time-consuming manual testing by the QA team.

- Other numerous fixes of inappropriate coding and design to reduce unnecessary code complexity and entanglement, make code more transparent and easy to understand, and to make development and testing less time and labor-consuming.

As a result, in 2019 Apollo project was shifting from monolithic to modular design with components based on CDI 2.0 and message-driven programming. All new development is done on those modern technologies where existing code allowed us to do so. The transfer to full code modularity has been finished, and the team is ready to move forward with development of dapp modularity.

Just one simple but convincing example of the changesə effectiveness:

As a result of the work described above we have a separate Apollo UI project and UI development team that works independently from backend developers. UI team now has:
Well documented and testable REST API of backend because we have now JAX-RS 2.1 based REST API with automatic documentation and test page based on OAS 3.0 standard.
Possibility to generate client stub code with OAS/Swagger tools for React.js.
Independent build and test system that exactly fits their needs.
Independent versioning of UI project and own workflow.

Development, testing, and release procedures are now well-defined and strictly followed by all team members. Furthermore, the team has automated build processes and created continuous integration infrastructure. Apart from mentioned, a huge unit test suite and integration test suite were developed by our QA department.

Dividing the Apollo project into modules gave us the opportunity to separately and independently update individual components of the project: web, desktop and mobile. It was executed into three following ways: deploy to the server, update the transaction with auto-update of the desktop and server application and, finally, manual installation. On the contrary, the NXT project was developed as a monolith.

As a result of such architecture, the updating of individual modules does not seem so flexible, because during any update it is necessary to include all the user and server part code in it.
The modular structure of the Apollo platform makes it possible to separately update these parts and include in the release both together and separately.

Simulation modelling

Besides the scope of main application development, we have many questions that require mathematical analysis or computer simulation. In the case of a known mathematical solution - we use it. But some questions do not have a strict solution.

A bright example of such questions is the statistical convergence of Apollo ledgers in distributed networks. Basic NXT consensus works by choosing the longest chain with the biggest cumulative difficulty. So network forms consensus or “common view” of blockchain ledger statistically, not in a determined way, by trying chains from different “forgers” and rollbacking to better chains.
We have changed the NXT block generation algorithm with developed adaptive forging algorithm to speed up transaction processing in the Apollo network and released it to the main network in 2018, as NXT’s solution didn’t satisfy the needs for successful Apollo development. As those changes were hard to implement, we had to find the right solution that included strict mathematical improvement and parameter variation for these algorithms.

We’ve developed a simulation modelling software suite that allowed research and stabilization of Apollo features such as adaptive forging, general network stability, statistical algorithms of diagnostics and so on. The idea was to simulate millions of transactions and forgings on blockchain network models with hundreds of nodes in a few seconds and see what parameter causes one or other result. Then we can build diagrams, run parameter optimization algorithms and find optimal parameter combinations for stable network behaviour.

The issue of blockchain forking in case of the temporarily disconnected network parts was solved using simulation. Adaptive forging parameters optimized also using this modelling software.

The simulation software suite will be used in the future for different consensus algorithms modelling and other tasks that require modelling and research before implementation and does not have direct mathematical solutions.

As a result of such architecture, the updating of individual modules does not seem so flexible, because during any update it is necessary to include all the user and server part code in it.
The modular structure of the Apollo platform makes it possible to separately update these parts and include in the release both together and separately.

2019 Refactoring process

Apart from the code refactoring, two big features were implemented: time-based sharding with quick blockchain start (Sharding) and decentralized exchange (DEX). All new features were developed using industry standards, best practices and new approaches prepared in previous steps. Both refactoring and implementation of new features were done in parallel.

The following tasks were accomplished during this period:

Backend tasks.

- New API is on JAX-RS REST standard, old API is re-written;
- Big preparation work has been done for future dapp modularity by switching to the CDI 2.0 standard and dependency injection approach instead of unacceptable static design; This work was done in all parts of code that are related to features mentioned above;
- Peer to peer level is optimized up to needed limits;
- Secure network transport is stabilized and is in use by Apollo network;
- El Gamal encryption is developed for secure information passing between frontend and backend;
- Crypto library developed in Java and JavaScript with a common test suite to unify all crypto operations in Apollo and avoid hard to find compatibility bugs;
- Data export/import module is implemented for sharding and other purposes;
- Predictable zipping with stable archive SHA-256 hash implemented;
- Merkle tree for blockchain quick diffs implemented;
- Mathematically correct statistical solver module developed to make a decision about data file and shard correctness on set of available nodes before downloading;
- Fast multi-threaded multi-source file downloader developed;
- Files, options and database locations unified for user mode and service mode;
- Command-line options interface created for the convenience of server deployments;
- Code covered with unit tests, the unit tests are part of the CI/CD build process;
- Integration tests suite developed;
- Performance/stress test suite has been developed, but it's a constant process that keeps going during the project lifetime.

Frontend tasks.

We have a new Apollo UI that already does not contain any legacy code and is based on React.js technology stack. It was an evolutionary process because the UI part had to be compatible with old backend parts.

- All jQuery-based legacy code is replaced with React.js framework based code. This allowed us to get rid of obsolete technologies and to structurize the code. Besides it, all AJAX calls to backend were replaced by a more effective and structured fetch/axios code with the usage of Promise class. This approach resolved the so-called “callback hell”;
- After transition to React.js project was split into logical components. This step allowed us to speed up development of UI significantly;
- All modal widows of UI were refactored following the “Don’t repeat yourself” principle. It was necessary for better code reuse and for simplifying new dialog windows development and existing dialog windows modification;
- Forms elements refactoring. All types of inputs, buttons, file uploads and so on were moved to separate reusable components. This allowed us to have all UI in a single consistent style and reduced possible user interface bugs;
- Tables refactored to use a single unified component that allows flexible and quick paginated table view development;
- ElGamal crypto algorithm of directional encryption was developed and used for secure passing of secret phrases to backend and for secure transfer of other sensitive data;
- New secure login procedure and UI with new style were developed for convenient and secure login;
- Dashboard refactoring and redesign in accordance with the provided layouts of the interface. Responsive styles for a better view on tablets and mobile devices;
- Implementation of account export/import interface using files on the user side;
- Site styles and DEX styles were refactored and unified to have a consistent look and feel of the whole application. Every page style and every modal dialog was refactored;
- The main menu of the application was redesigned and reimplemented, all related bugs fixed;
- Cordova build scripts of UI were rewritten to have common builds for IOS and Android mobile platforms. Necessary plugins were added to fix problems with different versions of supported mobile platforms;
- Client-side information storing developed using the Redux component. This reduced excess requests to the backend and made the UI faster and more responsive.

Conclusion

Refactoring is an important process towards creating a unique product with proprietary technological “know-hows” and functional stability. This document discussed the work that has been done to finish the refactoring process from the reasons why the project needed to change the code to the results of the mentioned process. This stage will grant us to rationally use the resources and stably develop the project further.

CHAPTER 1

Terminology

Steel is an iterative symmetric block encryption algorithm for 128-bit data blocks with a 256-bit key. The number of rounds is 14. The symmetric algorithm has three modes: main mode (OP), mode 1, mode 2.

Basic mode. Encryption procedure. The sequence of execution of procedures from 1 to 13 rounds is the following: SubBytes ➔ PerBits ➔ ShiftBytes ➔ PerBytes. The sequence of execution of procedures in the 14th round is the following: SubBytes ➔ PerBits ➔ ShiftBytes. Decryption procedure. The sequence of performing procedures in round 1 is the following: InvShiftBytes ➔ InvPerBits ➔ InvSubBytes. The sequence of performing procedures from 2 to 14th rounds is the following: InvPerBytes ➔ InvShiftBytes ➔ InvPerBits ➔ InvSubBytes.

Mode 1. The specified mode preserves the architecture of the main mode, but differs from it in operations PerBitsM and InvPerBitsM only (which are used instead of operations PerBits and InvPerBits, respectively).

Encryption procedure (see Appendix A). The sequence of execution of procedures from 1 to 13 rounds is the following: SubBytes ➔ PerBitsM ➔ ShiftBytes ➔ PerBytes. The sequence of performing procedures in round 14 is the following: SubBytes ➔ PerBitsM ➔ ShiftBytes.

Decryption procedure (see Appendix B). The sequence of performing the procedures in the 1st round: InvShiftBytes ➔ InvPerBitsM ➔ InvSubBytes. The sequence of execution of procedures from 2 to 14th rounds: InvPerBytes ➔ InvShiftBytes ➔ InvPerBitsM ➔ InvSubBytes.

Mode 2. The specified mode is the main mode without the PerBits and InvPerBits operations.

Encryption procedure (see Appendix C). The sequence of performing procedures from 1 - 13 rounds is the following: SubBytes ➔ ShiftBytes ➔ PerBytes. The sequence of the procedures in the 14th round is the following: SubBytes ➔ ShiftBytes. Decryption procedure (see Appendix D). The sequence of performing the procedures in the 1st round is the following: InvShiftBytes ➔ InvSubBytes. The sequence of performing procedures from 2 to 14th rounds is the following: InvPerBytes ➔ InvShiftBytes ➔ InvSubBytes

A **byte** is a sequence of 8 bits, a byte is considered as the plafal - PF in the context of algorithm herewith.
A **block** is a sequence of 16 bytes that the algorithm operates on.
The block is considered as (plafal docking procedure) - PF $\frac{doc}{S^{16}}$ in the context of algorithm herewith.
The **state matrix** is a complex of 16 bytes that displays the state of the block and the form before, while and upon the execution of all-round procedures.
Form is a sequence of 16 bytes, which is considered as plafal - PF $\frac{uniq}{ad}$
Key is a sequence of 32 bytes used as the encryption key.
Word is a sequence of 4 bytes. Nb - the number of words in the block (Nb = 4).
Nk - the number of words in the key (Nk = 8).
Round is an iteration of the transformation cycle over the state matrix. Number of rounds - Nr = 14.
Round key is a key used in the round. It is calculated for each round.
SubBytes is substitution of bytes in the state matrix using a substitution table.
PerBits is permutation of bits in a byte, it is considered as a counterclockwise rotation absolutely dynamic plafal - PF $\frac{uniq}{ad}$round the centre of symmetry at an angle $\varphi = \frac{360° \cdot n}{8} = 45° \cdot n$, where n is the number of turns in the context of algorithm herewith.
PerBitsM is a permutation of bits in a byte. In the context of this algorithm, it is considered as a composition of two transformations, namely: 1. Operation PerBits; 2. changing the place of bits.
ShiftBytes is bytes cyclic shift in state matrix for various qua $\frac{doc}{S^{16}}$ties. It is considered as a parallel transfer of PFi on PFj; i ≠ j of the PF complex in the context of algorithm herewith.
PerBytes is a permutation of bytes in the state matrix. It is considered as counterclockwise rotation of the absolutely dynamic plafal - PFuniqad around the centre of the symmetry at an angle $\varphi = \frac{360° \cdot n}{16} = 22.5° \cdot n$, where n is the number of turns, in the context of algorithm herewith.
InvSubBytes is a byte substitution in the state matrix using a reverse substitution table.
InvPerBits is a permutation of bits in a byte. It is considered as a clockwise rotation of absolutely dynamic plafal PF $\frac{uniq}{ad}$round the centre of symmetry at an angle of $\varphi = \frac{360° \cdot n}{8} = 45° \cdot n$, in the context of algorithm herewith, where n is the number of turns.

InvPerBitsM is a permutation of bits in a byte. In the context of this algorithm, it is considered as a composition of two transformations, namely: 1. operation InvPerBits; 2. changing the place of bits.

InvShiftBytes is a cyclic shift of bytes in the state matrix by different values. In the context of this algorithm, it is considered as a parallel transfer of PFj to PFi j ≠ i of the PF $\frac{doc}{S^{16}}$ omplex.

InvPerBytes is a permutation of bytes in the state matrix. In the context of this algorithm, it is considered as a clockwise rotation of absolutely dynamic plafal - PF $\frac{uniq}{ad}$ round the centre of the symmetry at an angle of $\varphi = \frac{360° \cdot n}{16} = 22.5° \cdot n$, where n is the number of turns.

Key Schedule

ExpandKey is a calculation of round keys for all rounds.

AddRoundKey is addition of the round key with the state matrix.

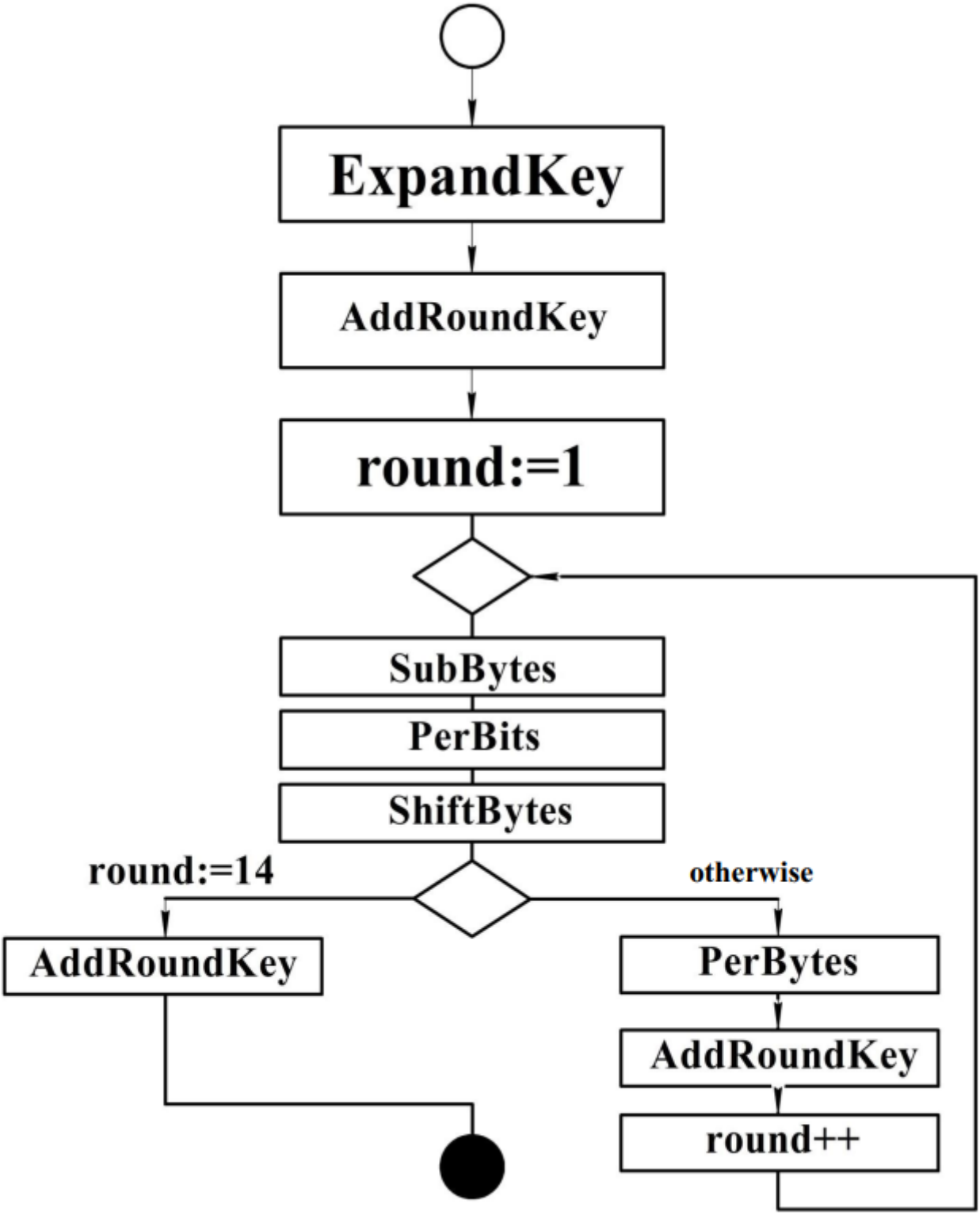


Figure 1.1 - Basic mode. Encryption

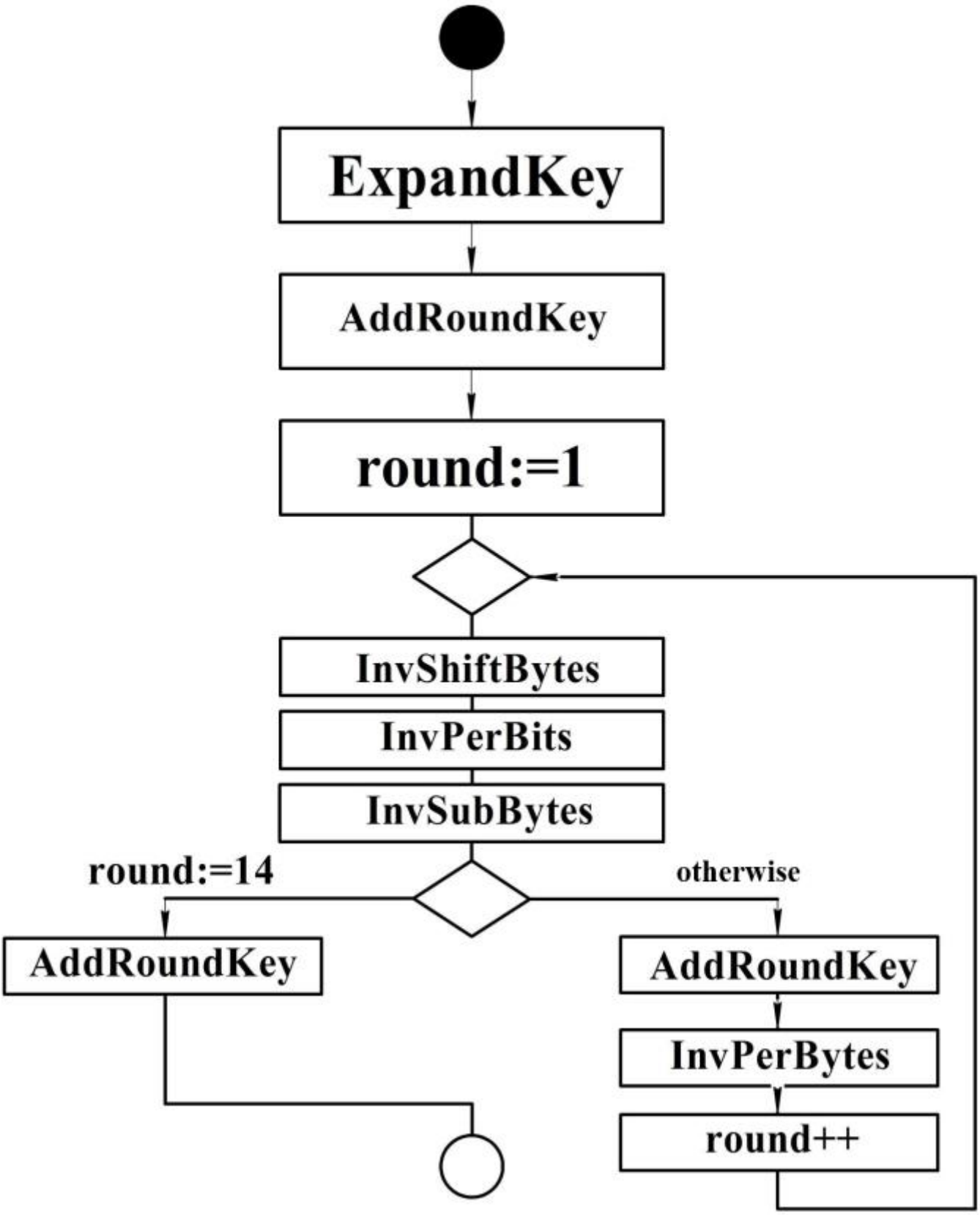


Figure 1.2 - Basic mode. Decryption

Main Mode

Byte

Operations are performed on bytes in a cryptoalgorithm. The number of plafales is equal to the number of states, that is, 28 = 256. According to [1, p. 16], there is a one-toone correspondence (bijection) for the byte {b7b6b5b4b3b2b1b0}, which forms a plafal: $b_{8-i} \leftrightarrow i, i = \overline{1;8}$; where i is the side of a regular octagon. For example, for byte 00010011 the corresponding plafal is shown on the Fig. 1.3.

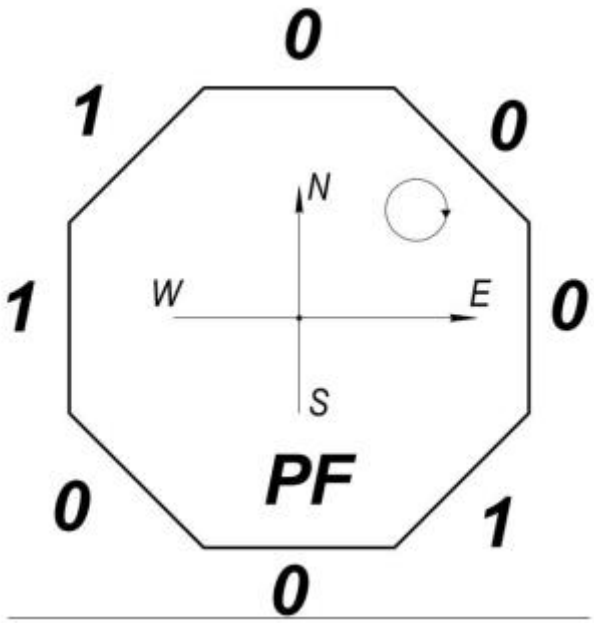


Figure 1.3 - Representation of byte 00010011

The orientation of the sides of the plafal is from north to northwest, that is: 1st side is in the north, 2nd side is in the northeast, and so on. Accordingly, the bypass rule is around the clockwise. The orientation of the sides does not change while turning around the center of symmetry (operations PerBits and InvPerBits) plafal.

Block

The block is a sequence of 16 plafales, virtually creating a cellular (honeycomb) structure - docking "of plafales" [1, p. 605] of sixteen plafales PF $\frac{uniq}{ad}$ p. 589] - fig. 1.4. Each of the plafales has 2, 3 or 4 common sides with each other. Fig. 1.5 shows the cellular structure of the positions of the docking complex of plafales. That is, $\mathrm{PF}_k, k = \overline{1;16}$ means that one of the sixteen plafales occupies the k-th position in the honeycomb structure; at the same time, PFk is not the k-th plafal (the algorithm does not have the concept of the k-th plafal). Proceeding from plafales docking definition procedure, we have the following: plafal, which has 2, 3 or 4 common sides with other plafales, certainly retains its byte structure (in Fig. 1.4: the 3rd side of PF1 (which takes the 1st position) corresponds to the set {0} and the 7th side of PF5 (which takes the 5th position) to the set {1 }).

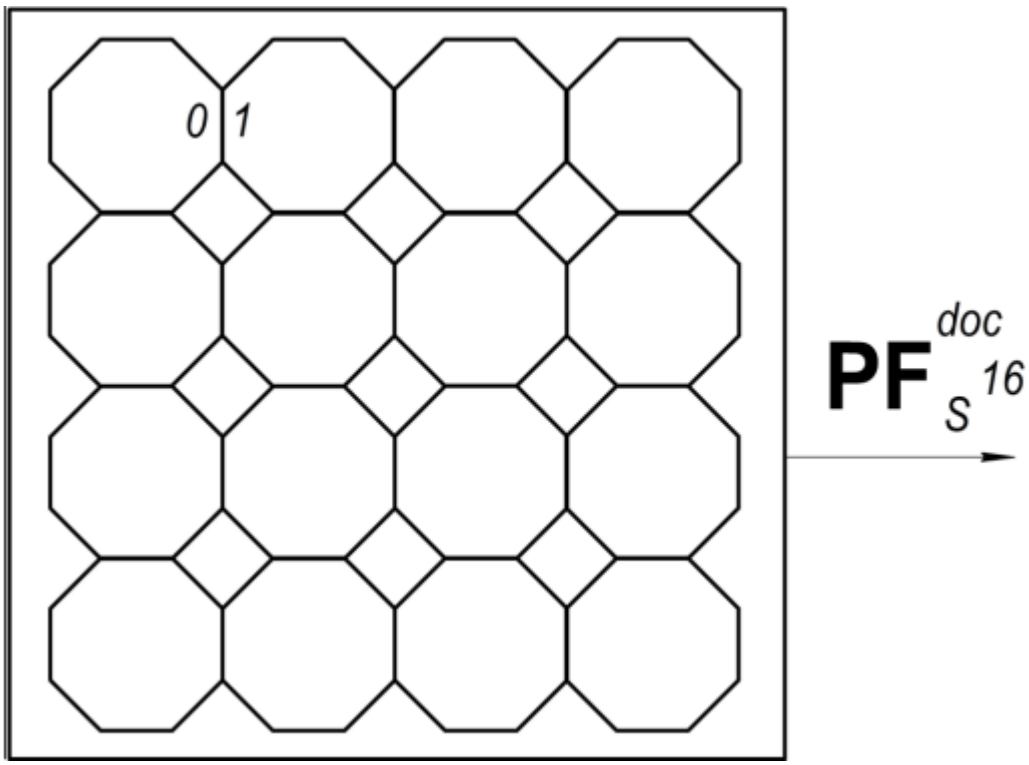


Figure 1.4 - Cellular structure of the block (“docking" of plafales)

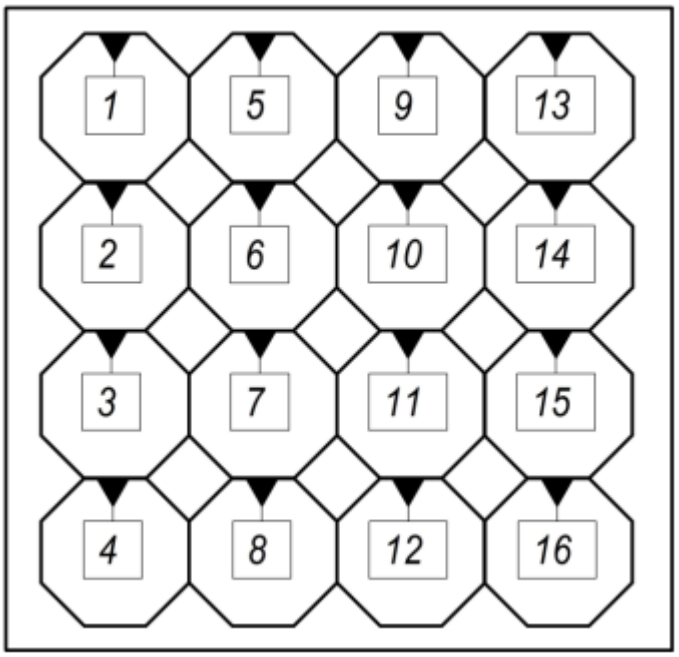


Figure 1.5 - Cellular structure of block positions

The static canvas of plafal

The above honeycomb block structure is situated on "the static canvas of plafal" (static plafal carpet) PF $\frac{U^{sp}}{16}$I, p. 16], which represents a plane in geometry - a twodimensional space of zero curvature (that is, on E2). In the context of the algorithm, the honeycomb structure of the block and the affine transformations performed on it (round operations PerBits, InvPerBits, ShiftBytes, InvShiftBytes) will be performed on regular octagons that form each of the plafales of the PF $\frac{doc}{S^{16}}$ complex. We will perform round operations PerBytes and InvPerBytes on a regular hexagon. All sixteen regular octagons are congruent with each other. Characteristics of a regular octagon: inscribed circle radius r=1, side length is $a = \frac{2}{1+\sqrt{2}}$, the radius of the circumscribed circle $R = \sqrt[4]{\frac{8}{3+2\sqrt{2}}}$ ... Accordingly, for the PF $\frac{doc}{S^{16}}$ complex: the origin of the regular rectangular coordinate system is a point O1 (0; 0) which is placed in the center of symmetry PF1, the corresponding benchmark is $R_1\ (O_1, \vec{e_1}, \vec{e_2})$; vectors ➔ $\vec{e_1}$ and ➔ $\vec{e_2}$ are orthonormal (with unit lengths). Coordinates of the centers of symmetries PFk, k = $\overline{1;16}$ and corresponding benchmarks are presented in the Table 1.1.

Table 1.1

PF_k	Symmetry centers coordinates	Benchmark
PF_1	p. $O_1(0;0)$	$R_1(O_1, \vec{e_1}, \vec{e_2})$
PF_2	p. $O_2(0;-2)$	$R_2(O_2, \vec{e_1}, \vec{e_2})$
PF_3	p. $O_3(0;-4)$	$R_3(O_3, \vec{e_1}, \vec{e_2})$
PF_4	p. $O_4(0;-6)$	$R_4(O_4, \vec{e_1}, \vec{e_2})$
PF_5	p. $O_5(2;0)$	$R_5(O_5, \vec{e_1}, \vec{e_2})$
PF_6	p. $O_6(2;-2)$	$R_6(O_6, \vec{e_1}, \vec{e_2})$
PF_7	p. $O_7(2;-4)$	$R_7(O_7, \vec{e_1}, \vec{e_2})$
PF_8	p. $O_8(2;-6)$	$R_8(O_8, \vec{e_1}, \vec{e_2})$
PF_9	p. $O_9(4;0)$	$R_9(O_9, \vec{e_1}, \vec{e_2})$
PF_{10}	p. $O_{10}(4;-2)$	$R_{10}(O_{10}, \vec{e_1}, \vec{e_2})$
PF_{11}	p. $O_{11}(4;-4)$	$R_{11}(O_{11}, \vec{e_1}, \vec{e_2})$
PF_{12}	p. $O_{12}(4;-6)$	$R_{12}(O_{12}, \vec{e_1}, \vec{e_2})$
PF_{13}	p. $O_{13}(6;0)$	$R_{13}(O_{13}, \vec{e_1}, \vec{e_2})$
PF_{14}	p. $O_{14}(6;-2)$	$R_{14}(O_{14}, \vec{e_1}, \vec{e_2})$
PF_{15}	p. $O_{15}(6;-4)$	$R_{15}(O_{15}, \vec{e_1}, \vec{e_2})$
PF_{16}	p. $O_{16}(6;-6)$	$R_{16}(O_{16}, \vec{e_1}, \vec{e_2})$

The above configuration is shown in Fig. 1.6.

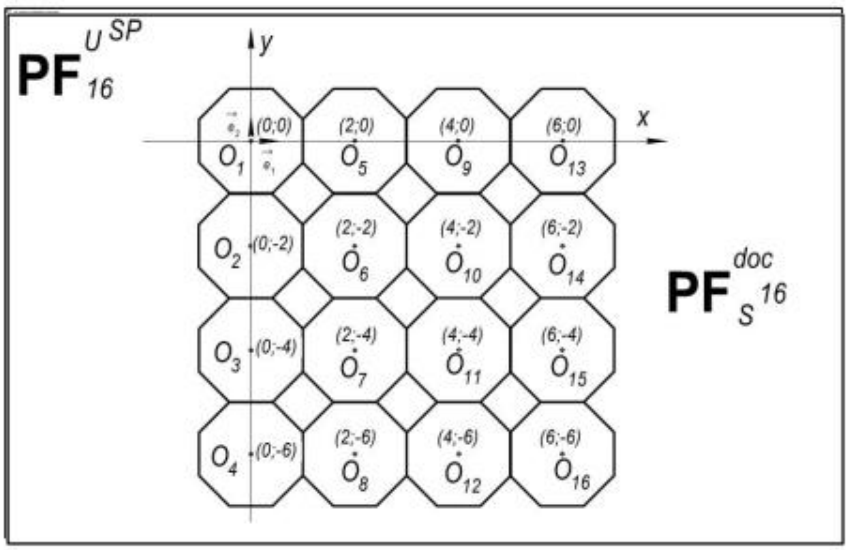


Figure 1.6 - Configuration on E_2

Each of the sixteen regular octagons is uniquely determined by the coordinates of its vertices. It is enough to consider the position of PF1 - fig. 1.7; the coordinates of

the vertices of the remaining regular octagons are determined by the parallel translation vector $\overrightarrow{O_1O_j}, j = \overline{2;16}$.
The coordinates of the vertices of a regular octagons and a regular hexagons are determined by the formulas:

$$x_i = O_x + R \cdot \cos(\phi_0 + \frac{2\pi i}{8}), i = \overline{0; n-1},$$

$$y_i = O_y + R \cdot \sin(\phi_0 + \frac{2\pi i}{8}), i = \overline{0; n-1}.$$

(Ox; Oy) are the coordinates of the center of symmetry; φ0 is the angular coordinate of the first vertex.
PF1 vertices coordinates: (≈0.415;1), (1;≈0.415), (1;≈-0.415), (≈0.415;-1), (≈-0.415;-1), (-1;≈-0.415), (-1;≈0.415), (≈-0.415;1).

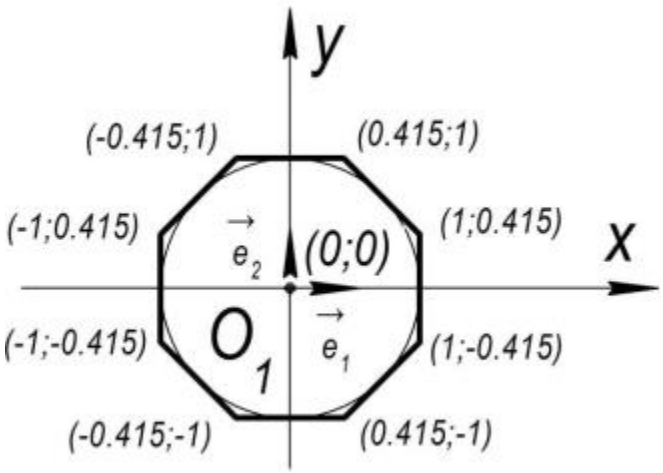


Figure 1.7 - PF1 vertex coordinates

The northern side of PF_1, is determined in the R_1 benchmark by the coordinates of two vertices before and after the execution of the round procedures: (≈0.415; 1), (≈-0.415; 1). The orientation of the other sides is determined in accordance with clause 1.2.1.

State Matrix

The state matrix (Fig. 1.8, Fig. 1.9) displays the state of the block and the form during the execution of all round procedures. That is, a 4 × 4 matrix: $A = [a_{ij}]_{i=\overline{1;4}}^{j=\overline{1;4}}$ _

which reflects the unchanged order of bytes in a block according to the following principle:

$$PF_k \leftrightarrow a_{ij}, k = \overline{1;16}. \tag{1.1}$$

Table 1.2

$PF_1 \leftrightarrow a_{11}$	$PF_2 \leftrightarrow a_{21}$	$PF_3 \leftrightarrow a_{31}$	$PF_4 \leftrightarrow a_{41}$
$PF_5 \leftrightarrow a_{12}$	$PF_6 \leftrightarrow a_{22}$	$PF_7 \leftrightarrow a_{32}$	$PF_8 \leftrightarrow a_{42}$
$PF_9 \leftrightarrow a_{13}$	$PF_{10} \leftrightarrow a_{23}$	$PF_{11} \leftrightarrow a_{33}$	$PF_{12} \leftrightarrow a_{43}$
$PF_{13} \leftrightarrow a_{14}$	$PF_{14} \leftrightarrow a_{24}$	$PF_{15} \leftrightarrow a_{34}$	$PF_{16} \leftrightarrow a_{44}$

$$\Lambda_{16} \leftrightarrow a_{ij}. \tag{1.2}$$

PFk is a cellular structure of complex positions i.e. “docking“ of plafales (fig. 1.5). More detailed: a_{11}– 1-st byte, a_{21}– 2-nd byte, etc. It may be PF, etc, during the permutations of the plafales (ShiftBytes, InvShiftBytes, PerBytes, InvPerBytes) at position a_{11} (1st byte) Since each byte $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$ written in the binary number system, corresponds to a record in the hexadecimal system - Λ_{16}, then according to thecorrespondence (1.2), in the matrix of states the first byte a11 will correspond to the byterecord in Λ_{16}, which is removed from PF1, according to the orientation of the sides of theregular octagon (Sec. 1.2.1, Sec. 1.2. 3). Of course, PF_k is uniquely recovered from Λ_{16}.
For example, for byte 00010011 (p. 1.2.1): {00010011} ↔ 13_{16}. Accordingly, 13_{16} ↔ a_{11} (in the case of PF_1).

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

Figure 1.8 - Matrix of states (general view)

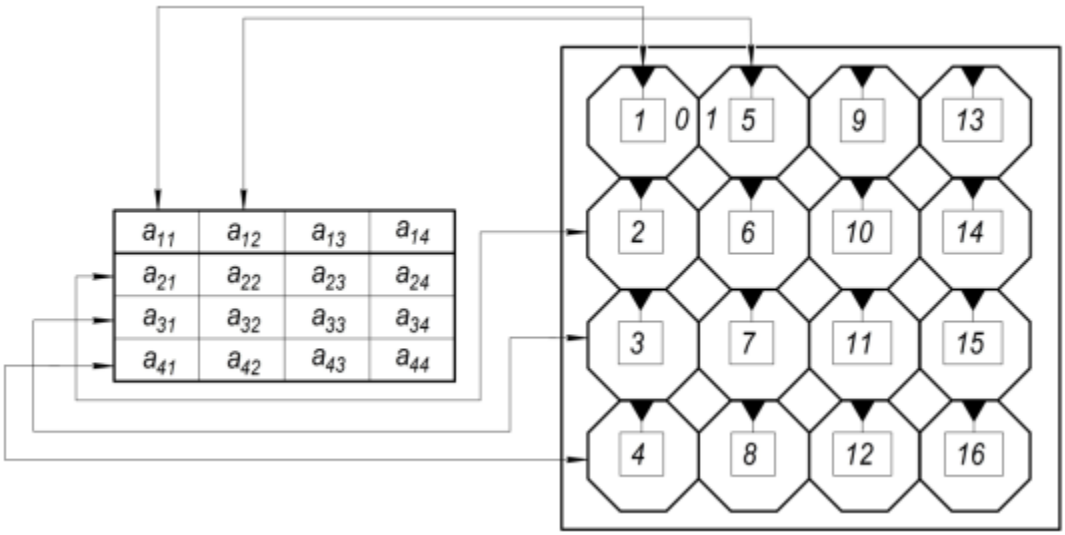


Figure 1.9 - Dependence of the state matrix and the structure of block positions

Form

The form is a plafal - PF_{ad}^{uniq} [1, p. 16, p. 589], formed as follows: there is a one-to-one (*mutually unambiguous*) correspondence that forms a plafal for a regular hexagon : $a_{ij} \leftrightarrow l \Leftrightarrow \Lambda_{16} \leftrightarrow l, l = \overline{1;16}$ where l is the side of a regular hexagon. For example, for the matrix of states (Fig. 1.10): the corresponding plafal is presented on the
Fig. 1.11.

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

↔

13	17	AB	64
57	FD	E3	CA
83	B7	EA	48
D4	C3	BF	A1

Figure 1.10 - State Matrix (as an example)

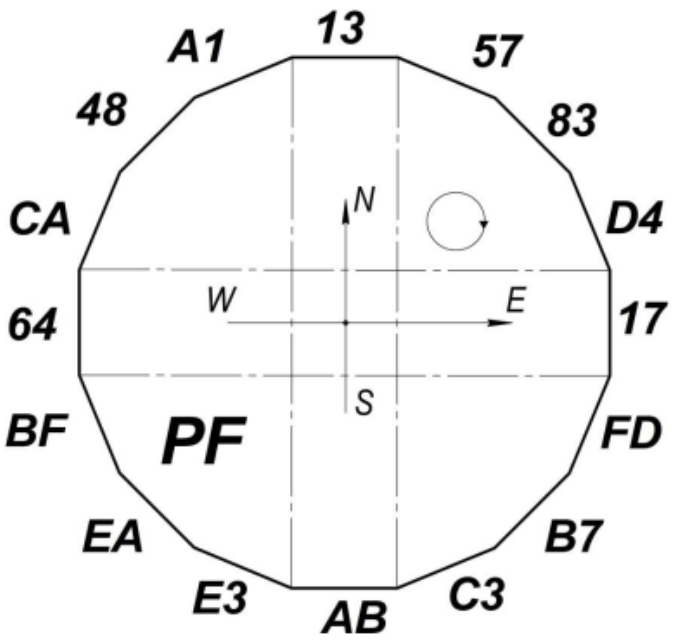


Figure 1.11 - Form

The orientation of the sides of the plafal is from north to north northwest, that is: 1 side is directed northward, 2 side is directed north to the north-eastward, etc. Accordingly, the bypass rule is the clockwise one. When turning around the plafal's center of symmetry (operations PerBytes and InvPerBytes), the orientation of the sides does not change. We will perform round operations PerBytes and InvPerBytes on a regular hexagon. The regular hexagon is placed on a static plafal canvas different from the one on which the $\mathrm{PF}_{S^{16}}^{doc}$ complex is located. Regular hexagon characteristics: the incircle's radius r = 1, side length $a = 2r\tan\frac{\pi}{16} \approx 0.3976$, the circumcircle's radius $R = \frac{r}{\cos\frac{\pi}{16}} \approx 1.0195$. Accordingly, for PFForm: the beginning of the regular rectangular coordinate system i.e. the point. O1 (0; 0) is in the center of symmetry of a regular hexagon, the corresponding benchmark is $\mathrm{R}_1\ (\mathrm{O}_1, \overrightarrow{e_1}, \overrightarrow{e_2})$; the vectors and are orthonormal (with unit lengths). Vertices' PFForm coordinates are the following: (≈0.2; 1), (≈0.56; ≈0.84), (≈0.84; ≈0.56), (1; ≈0.2), (1; ≈-0.2), (≈0.84; ≈-0.56), (≈0.56; ≈-0.84), (≈0.2; -1), (≈-0.2; -1), (≈-0.56; ≈-0.84), (≈-0.84; ≈ -0.56), (-1; ≈-0.2), (-1; ≈0.2), (≈-0.84; ≈0.56), (≈-0.56; ≈0.84), (≈-0.2; 1). The PFForm north side is defined prior to and upon the execution of the round procedures in the benchmark R1 by the coordinates of two vertices: (≈0.2; 1), (≈-0.2; 1).

The orientation of other sides should be defined via the clockwise round.

The above configuration is depicted on the Fig. 1.12

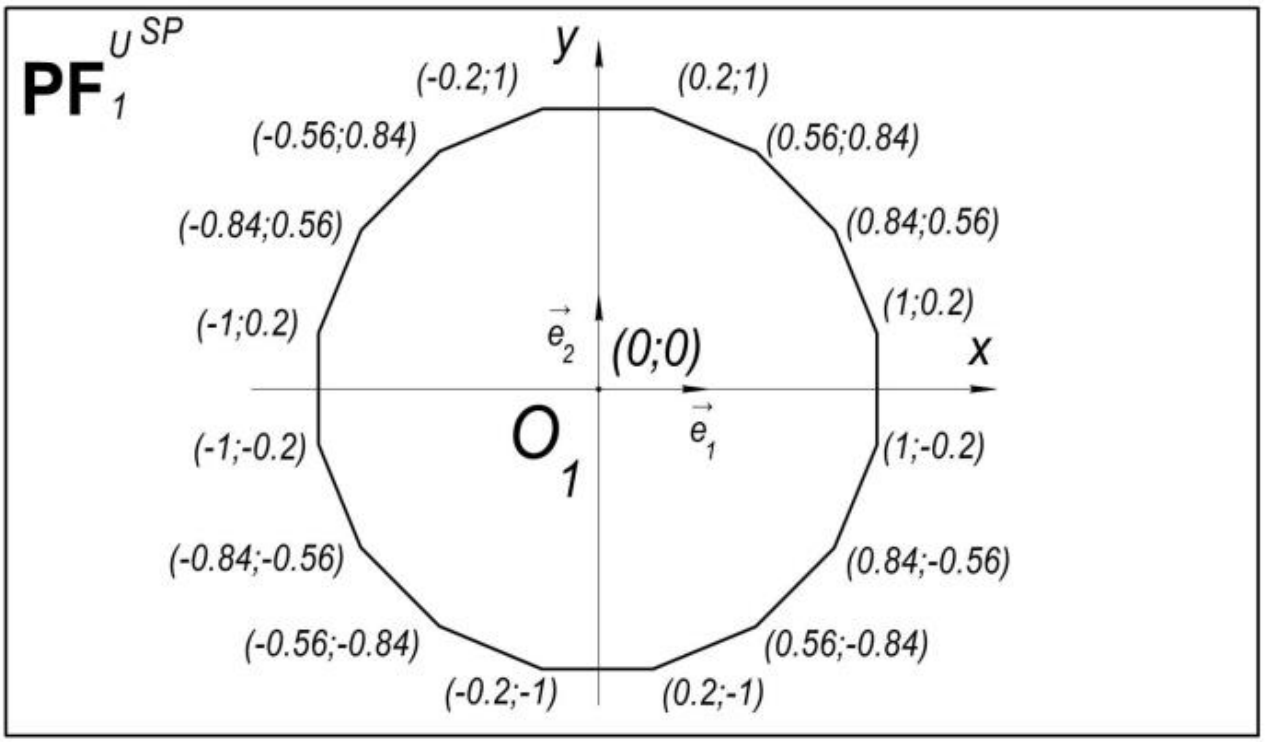


Figure 1.11 - Form

Relationship between form, state matrix and block

While summarizing results of 1.2.1 – 1.2.5, it becomes possible to gain the comprehensive and logical relationship between a form, state matrix and block – see Fig. 1.13. The relationship is expressed via the mutually-unambiguous correspondence:

$$l \leftrightarrow \Lambda_{16} \leftrightarrow a_{ij} \leftrightarrow \{b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0\} \leftrightarrow PF_k. \qquad (1.3)$$

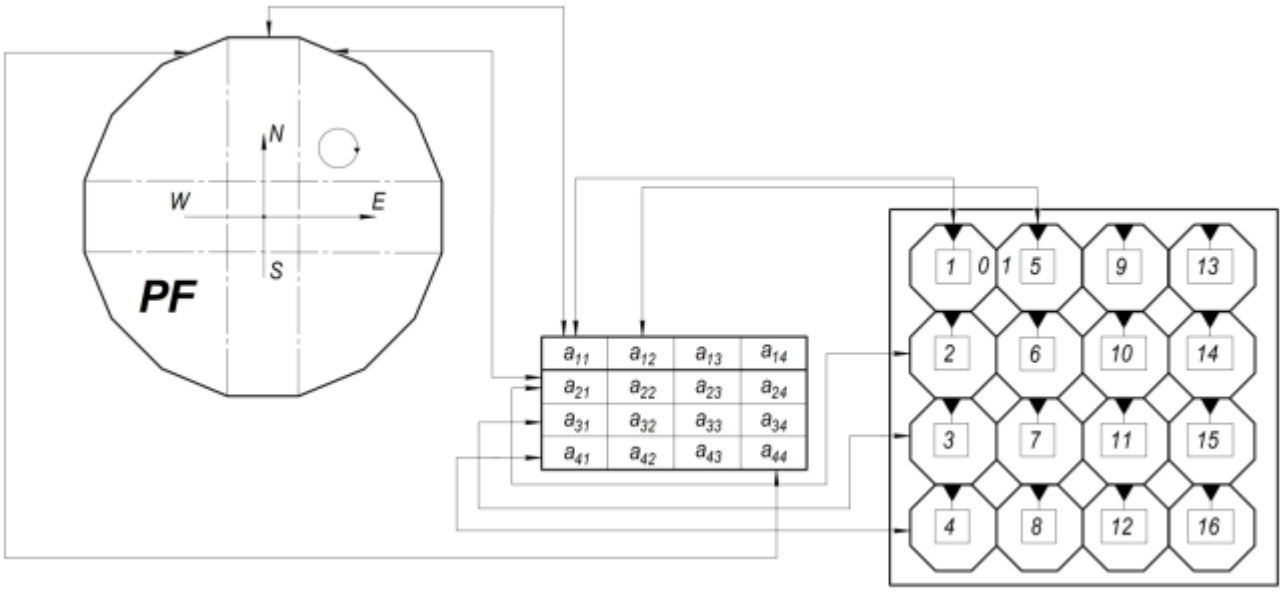


Figure 1.13 — the logical relationship between a form, state matrix and block

The state matrix reflects the block and form state prior to, in the course of and upon the performing of all round procedures.
Let's show it on example a11. The first byte a11 corresponds to the record in the hexadecimal number system –Λ16 and in the binary number system – {b7b6b5b4b3b2b1b0}.C{b7b6b5b4b3b2b1b0} creating the PF_1 (point 1.2.1). Λ16 corresponds to the northern side of the PF_{Form} (point 1.2.5). Upon of round operations completion on the block $PF_{S^{16}}^{doc}$ (PerBits, InvPerBits, ShiftBytes, InvShiftBytes) with PF1, according to the sides orientation (point 1.2.1 – 1.2.3), the record {b7b6b5b4b3b2b1b0} is read and converted into the Λ16, etc. Just by the same way in reverse order. Undoubtfully, encrypted (decrypted) block is gained from the State Matrix.

Operation SubBytes

Round Operation SubBytes performs byte permutation in a state matrix by means of permutation table – see Fig. 1.14. For example {41} will be replaced by {86}.

x/y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	5a	73	dd	af	d9	67	5f	99	42	01	b5	65	f7	bb	a7	d3
1	d4	84	c9	f5	ed	56	39	e4	f1	b4	e0	fc	93	b0	71	a0
2	7b	fe	a5	2a	47	cf	df	78	43	a9	ea	f2	8b	aa	49	26
3	08	cb	23	d1	0a	4b	12	2d	64	22	40	e1	9f	17	6a	37
4	03	86	13	1a	27	4f	33	60	91	5b	9b	ad	32	79	dc	48
5	5c	b1	00	f9	10	eb	c3	6d	66	bc	b7	4e	51	54	70	7d
6	c2	fb	74	bf	07	1d	3a	61	45	be	04	ef	30	0f	77	98
7	34	95	02	1f	a2	3d	15	f3	e5	b6	da	09	20	ff	ee	d8
8	97	0c	a1	ba	3f	57	14	36	c1	76	bd	ab	0d	ce	a8	75
9	21	06	7c	6e	11	58	41	18	46	de	b8	05	7e	2f	0b	e7
a	31	2c	69	81	52	24	82	1b	0e	d6	53	62	29	96	a3	e6
b	fa	38	9d	f4	8d	7a	2e	63	6c	35	3e	8f	f0	e9	d5	80
c	3b	87	25	8e	1c	a6	b2	3c	59	db	55	5e	72	6f	9a	89
d	16	6b	b3	cc	50	44	f6	4a	c4	1e	5d	b9	83	68	4d	c7
e	e2	f8	a4	90	2b	9e	9c	94	ae	ca	e8	ec	cd	c5	28	fd
f	8c	d0	8a	4c	7f	e3	c0	92	88	ac	c6	d2	85	c8	d7	19

Figure 1.14 — Permutation table of operation SubBytes

Operation PerBits

Round Operation PerBits performs permutation of bits in a byte. In a context of the algorithm herewith, the operation is considered as a turn of one separate absolutely dynamic plafal – PF_{ad}^{uniq} [1,c589] of the PF_{S16}^{doc}, complex around the center of symmetryon the angle of $\varphi = \frac{360^\circ \cdot n}{8} = 45^\circ \cdot n, n \in N$, n – is the number of rounds. According to the p. 1.2.3, round operations PerBits will be performed on regular octagons, creatingeach of plafales of the PF_{S16}^{doc} complex. The turn on a such angle transfers the regular octagon into itself. The set of turn angles is: M={45◦;90◦;135◦;180◦;225◦;270◦;315◦;360◦}.It is obviously that $M \cong Z_8$, Z_8 – the module residue ring 8. The separately taken regular octagon – PFk has its own number of turns which is defined by a formula:

$$n = \begin{cases} n \equiv f(k) \pmod 8,\ 8 \nmid f(k), \\ n \equiv k \pmod 8,\ 8 \mid f(k), \end{cases}$$

where f (*k*) is a function of a turns number it has a unique kind for each round; k– is a position of PFk in the cellular block structure (p. 1.2.2).
The function of work of a turn $\omega(t)_{PF_d^{uniq}}$ [1, c 590] (the change of regular octagon vertices coordinates in the benchmark in a context of algorithm herewith: $\mathrm{R}_k, k = \overline{1;16}$):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}. \tag{1.4}$$

$\begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$ - the matrix of counterclockwise turn; (x'; y') – the point coordinates sinϕ cosϕ gained via the point (x; y).

The tables will be performed for all rounds (1 – 14) depicting the above functional features.

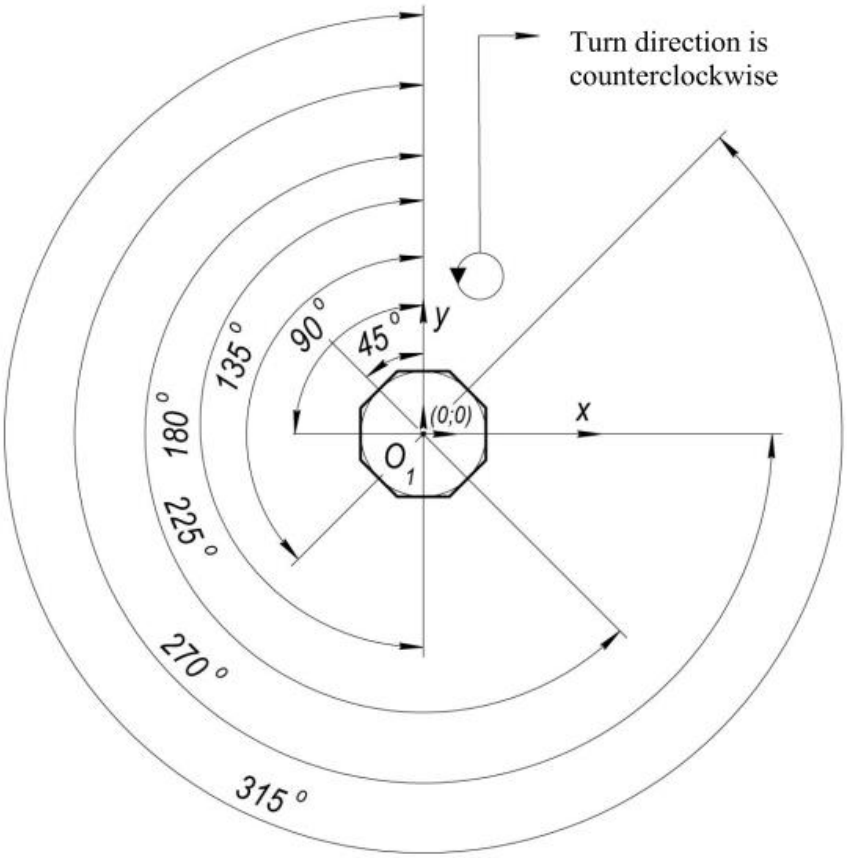


Figure 1.15 — the set of turn angles for the PF1 of the complex PF^{doc}_{S16}

Table 1.3 — the turn matrix depending on a turn number n

n	φ	$\begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$
1	45°	$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$
2	90°	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
3	135°	$\begin{bmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$
4	180°	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
5	225°	$\begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$
6	270°	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
7	315°	$\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$

Round 1

$f(k)=k$:

Table 1.4

PF_k	$f(k)$	n
PF_1	1	1
PF_2	2	2
PF_3	3	3
PF_4	4	4
PF_5	5	5
PF_6	6	6
PF_7	7	7

PF_8	8	0
PF_9	9	1
PF_{10}	10	2
PF_{11}	11	3
PF_{12}	12	4
PF_{13}	13	5
PF_{14}	14	6
PF_{15}	15	7
PF_{16}	16	0

Round 2

$f(k)=k^2$:

Table 1.5

PF_k	$f(k)$	n
PF_1	1	1
PF_2	4	4
PF_3	9	1
PF_4	16	4
PF_5	25	1
PF_6	36	4
PF_7	49	1
PF_8	64	0
PF_9	81	1
PF_{10}	100	4
PF_{11}	121	1
PF_{12}	144	4
PF_{13}	169	1
PF_{14}	196	4
PF_{15}	225	1
PF_{16}	256	0

Round 3

Round 4

$$f(k)=k+k^2+k^3:$$

Table 1.6

PF_k	$f(k)$	n
PF_1	3	3
PF_2	14	6
PF_3	39	7
PF_4	84	4
PF_5	155	3
PF_6	258	2
PF_7	399	7
PF_8	584	0
PF_9	819	3
PF_{10}	1110	6
PF_{11}	1463	7
PF_{12}	1884	4
PF_{13}	2379	3
PF_{14}	2954	2
PF_{15}	3615	7
PF_{16}	4368	0

$$f(k)=k^2+k:$$

Table 1.7

PF_k	$f(k)$	n

PF_1	2	2
PF_2	6	6
PF_3	12	4
PF_4	20	4
PF_5	30	6
PF_6	42	2
PF_7	56	7
PF_8	72	0
PF_9	90	2
PF_{10}	110	6
PF_{11}	132	4
PF_{12}	156	4
PF_{13}	182	6
PF_{14}	210	2
PF_{15}	240	7
PF_{16}	272	0

Round 5

$$f(k)=k^3+k:$$

Table 1.8

PF_k	$f(k)$	n
PF_1	2	2
PF_2	10	2
PF_3	30	6
PF_4	68	4
PF_5	130	2
PF_6	222	6
PF_7	350	6
PF_8	520	0

PF_9	738	2
PF_{10}	1010	2
PF_{11}	1342	6
PF_{12}	1740	4
PF_{13}	2210	2
PF_{14}	2758	6
PF_{15}	3390	6
PF_{16}	4112	0

Round 6

$f(k)=[\pi^k]$:

Table 1 . 9

PF_k	$f(k)$	n
PF_1	3	3
PF_2	9	1
PF_3	31	7
PF_4	97	1
PF_5	306	2
PF_6	961	1
PF_7	3020	4
PF_8	9488	0
PF_9	29809	1
PF_{10}	93648	2
PF_{11}	294204	4
PF_{12}	924269	5
PF_{13}	2903677	5
PF_{14}	9122171	3
PF_{15}	28658145	1
PF_{16}	90032220	4

Round 7

Round 8

$$f(k)=2^k+1 :$$

Table 1.10

PF_k	$f(k)$	n
PF_1	3	3
PF_2	5	5
PF_3	9	1
PF_4	17	1
PF_5	33	1
PF_6	65	1
PF_7	129	1
PF_8	257	1
PF_9	513	1
PF_{10}	1025	1
PF_{11}	2049	1
PF_{12}	4097	1
PF_{13}	8193	1
PF_{14}	16385	1
PF_{15}	32769	1
PF_{16}	65537	1

$f(k)= \sinh(k) :$

Table 1.11

PF_k	$f(k)$	n
PF_1	1	1
PF_2	3	3

PF_3	10	2
PF_4	27	3
PF_5	74	2
PF_6	201	1
PF_7	548	4
PF_8	1490	2
PF_9	4051	3
PF_{10}	11013	5
PF_{11}	29937	1
PF_{12}	81377	1
PF_{13}	221206	6
PF_{14}	601302	6
PF_{15}	1634508	4
PF_{16}	4443055	7

Round 9

$f(k)=0.5\cdot(5k^3-3k)$:

Table 1.12

PF_k	$f(k)$	n
PF_1	1	1
PF_2	17	1
PF_3	63	7
PF_4	154	2
PF_5	305	1
PF_6	531	3
PF_7	847	7
PF_8	1268	4
PF_9	1809	1
PF_{10}	2485	5

PF_{11}	3311	7
PF_{12}	4302	6
PF_{13}	5473	1
PF_{14}	6839	7
PF_{15}	8415	7
PF_{16}	10216	0

Round 10

$$f(k) = |k^4 - 6k^2 + 3| :$$

Table 1.13

PF_k	$f(k)$	n
PF_1	2	2
PF_2	5	5
PF_3	30	6
PF_4	163	3
PF_5	478	6
PF_6	1083	3
PF_7	2110	6
PF_8	3715	3
PF_9	6078	6
PF_{10}	9403	3
PF_{11}	13918	6
PF_{12}	19875	3
PF_{13}	27550	6
PF_{14}	37243	3
PF_{15}	49278	6
PF_{16}	64003	3

Round 11

$$f(k)=2k^2-1$$

Table 1.14

Round 12

PF_k	$f(k)$	n
PF_1	1	1
PF_2	7	7
PF_3	17	1
PF_4	31	7
PF_5	49	1
PF_6	71	7
PF_7	97	1
PF_8	127	7
PF_9	161	1
PF_{10}	199	7
PF_{11}	241	1
PF_{12}	287	7
PF_{13}	337	1
PF_{14}	391	7
PF_{15}	449	1
PF_{16}	511	7

$f(k)= [\exp(k)]$:

Table 1.15

PF_k	$f(k)$	n
PF_1	2	2

PF_2	7	7

PF_3	20	4
PF_4	54	6
PF_5	148	4
PF_6	402	2
PF_7	1093	5
PF_8	2971	3
PF_9	8074	2
PF_{10}	21938	2
PF_{11}	59612	4
PF_{12}	161979	3
PF_{13}	440129	1
PF_{14}	1195920	6
PF_{15}	3249556	4
PF_{16}	8829694	6

Round 13

$f(k)=k!!$:

Table 1.16

PF_k	$f(k)$	n
PF_1	1	1
PF_2	2	2
PF_3	3	3
PF_4	8	4
PF_5	15	7
PF_6	48	6
PF_7	105	1
PF_8	384	0
PF_9	945	1
PF_{10}	3840	2

PF_{11}	10395	3
PF_{12}	46080	4
PF_{13}	135135	7
PF_{14}	645120	6
PF_{15}	2027025	1
PF_{16}	10321920	0

Round 14

$f(k)=k$:

Table 1.17

PF_k	$f(k)$	n
PF_1	1	1
PF_2	2	2
PF_3	3	3
PF_4	4	4
PF_5	5	5
PF_6	6	6
PF_7	7	7
PF_8	8	0
PF_9	9	1
PF_{10}	10	2
PF_{11}	11	3
PF_{12}	12	4
PF_{13}	13	5
PF_{14}	14	6
PF_{15}	15	7
PF_{16}	16	0

OperationShiftBytes

The round operation ShiftBytes cyclically shifts the bytes in the state matrix by different values. In the context of this algorithm, it is considered as a parallel transfer of PFi (Section 1.2.2) to PFj (Section 1.2.2); i ≠ j of the complex $\mathbf{PF}_{S^{16}}^{doc}$ The parallel transfer means the “lucidification“ of the system of plafales [1, c 592] (clarification of the honeycomb structure of the block) - $\mathrm{LUC}_{PF_{S^{16}}}$

$$LUC_{PF_{S^{16}}} = \begin{cases} PF_2 \to PF_{14}, PF_{14} \to PF_{10}, PF_{10} \to PF_6, PF_6 \to PF_2, \\ PF_3 \to PF_{11}, PF_{15} \to PF_7, PF_{11} \to PF_3, PF_7 \to PF_{15}, \\ PF_4 \to PF_8, PF_{16} \to PF_4, PF_{12} \to PF_{16}, PF_8 \to PF_{12}. \end{cases}$$

According to section 1.2.3, we will perform the round ShiftBytes operations on regular octagons that form each of the plafales of the $\mathbf{PF}_{S^{16}}^{doc}$ complex. That is, a regular octagon that occupies the 2nd position, by a parallel transfer goes to the 14th position, etc. In fact, a regular octagon from the frame Ri goes to the frame Rj. The vector of the indicated transition is $\overrightarrow{O_iO_j}$. To summarize the above:

$$\begin{cases} R_2 \to R_{14}, \overrightarrow{O_2O_{14}} = (6,0), \\ R_{14} \to R_{10}, \overrightarrow{O_{14}O_{10}} = (-2,0), \\ R_{10} \to R_6, \overrightarrow{O_{10}O_6} = (-2,0), \\ R_6 \to R_2, \overrightarrow{O_6O_2} = (-2,0), \\ R_3 \to R_{11}, \overrightarrow{O_3O_{11}} = (4,0), \\ R_{15} \to R_7, \overrightarrow{O_{15}O_7} = (-4,0), \\ R_{11} \to R_3, \overrightarrow{O_{11}O_3} = (-4,0), \\ R_7 \to R_{15}, \overrightarrow{O_7O_{15}} = (4,0), \\ R_4 \to R_8, \overrightarrow{O_4O_8} = (2,0), \\ R_{16} \to R_4, \overrightarrow{O_{16}O_4} = (-6,0), \\ R_{12} \to R_{16}, \overrightarrow{O_{12}O_{16}} = (2,0), \\ R_8 \to R_{12}, \overrightarrow{O_8O_{12}} = (2,0). \end{cases}$$

This configuration is performed in all rounds 1 - 14

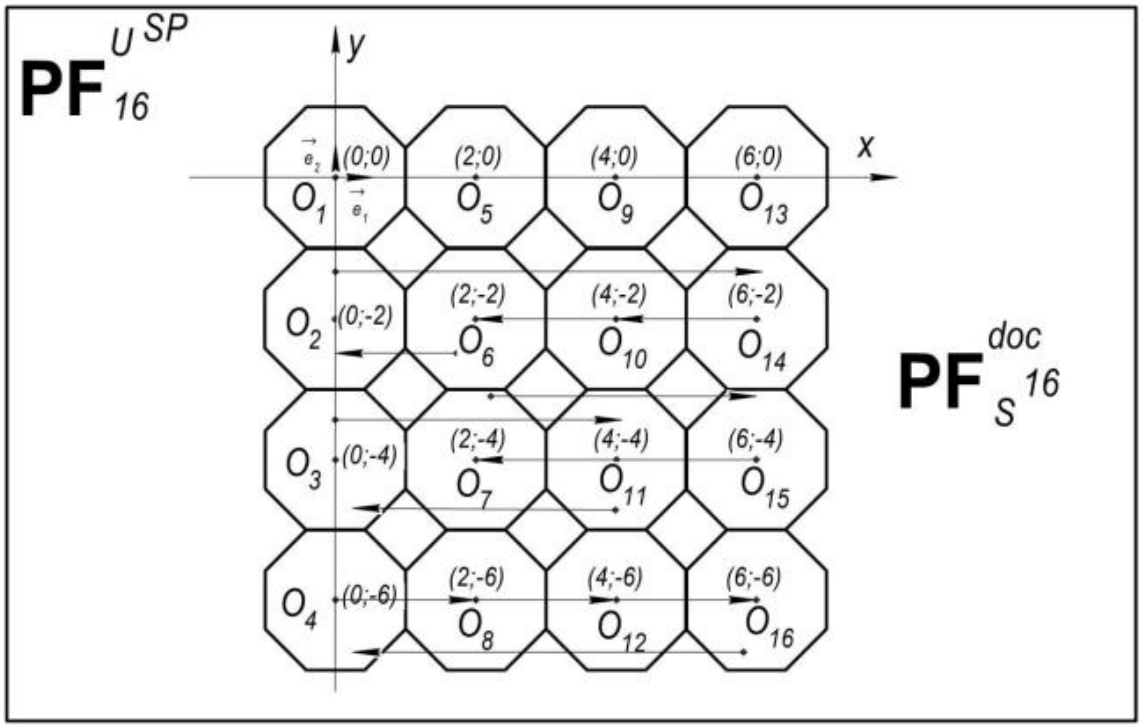


Figure 1.16 — Round Operation ShiftBytes

Operation PerBytes

The PerBytes operation permutes bytes in the state matrix. In the context of this algorithm, it is considered as a counterclockwise rotation of absolutely dynamic plafal – PF_{ad}^{uniq} around the centre of symmetry by an angle $\varphi = \frac{360^\circ \cdot n}{16} = 22.5^\circ \cdot n$ where n is the number of turns.

According to p. 1.2.3 and p. 1.2.5, we will perform round PerBytes operations on a regular hexagon that forms a shape. A rotation, by a given angle, transforms a regular hexagon into itself. The set of round angles:

$$M = \{22.5^\circ; 45^\circ; 67.5^\circ; 90^\circ; 112.5^\circ; 135^\circ; 157.5^\circ; 180^\circ; 202.5^\circ; 225^\circ; 247.5^\circ; 270^\circ;$$

$$292.5^\circ; 315^\circ; 337.5^\circ; 360^\circ\}.$$

Where $M \cong Z_{16}$, Z_{16} is a residue ring modulo 16. The regular hexagon is a PFForm, has the unique number of turns exists in each round, determined by a formula:

$$n = \sum_{i-0}^{12} \alpha_i k^i,$$

$$\alpha_0 = -32868,\ \alpha_1 = \frac{303385829}{3080},\ \alpha_2 = -\frac{5092571437}{41580},\ \alpha_3 = \frac{7358386177}{86400},\ \alpha_4 = -\frac{406877250083}{10886400},$$

$$\alpha_5 = \frac{3196480771}{290304},\ \alpha_6 = -\frac{19592011253}{8709120},\ \alpha_7 = \frac{781515743}{2419200},\ \alpha_8 = -\frac{471962587}{14515200},\ \alpha_9 = \frac{51743}{23040},$$

$$\alpha_{10} = -\frac{882839}{8709120},\ \alpha_{11} = \frac{214933}{79833600},\ \alpha_{12} = -\frac{15277}{479001600}.$$

Work function of a turn $\omega(l)_{PF_d^{uniq}}$ [1, c 590] (change of vortices coordinates of the regular hexagon in the benchmark R1 in a context of algorithm herewith):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{1.5}$$

Table 1.18 — Kind of turn matrix depending on turns number n

n	φ	$\begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}$
1	22.5°	$\begin{bmatrix} 0.92 & -0.38 \\ 0.38 & 0.92 \end{bmatrix}$
2	45°	$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$
3	67.5°	$\begin{bmatrix} 0.38 & -0.92 \\ 0.92 & 0.38 \end{bmatrix}$
4	90°	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
5	112.5°	$\begin{bmatrix} -0.38 & -0.92 \\ 0.92 & -0.38 \end{bmatrix}$
6	135°	$\begin{bmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$
7	157.5°	$\begin{bmatrix} -0.92 & -0.38 \\ 0.38 & -0.92 \end{bmatrix}$

8	180°	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
9	202.5°	$\begin{bmatrix} -0.92 & 0.38 \\ -0.38 & -0.92 \end{bmatrix}$
10	225°	$\begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$
11	247.5°	$\begin{bmatrix} -0.38 & 0.92 \\ -0.92 & -0.38 \end{bmatrix}$
12	270°	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
13	292.5°	$\begin{bmatrix} 0.38 & 0.92 \\ -0.92 & 0.38 \end{bmatrix}$
14	315°	$\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$
15	337.5°	$\begin{bmatrix} 0.92 & 0.38 \\ -0.38 & 0.92 \end{bmatrix}$

Round 1

Table 1.19

PF_{Form}	k	n
PF_{Form}	1	3

Round 2

Table 1.20

PF_{Form}	k	n
PF_{Form}	2	14

Round 3

Table 1.21

PF_{Form}	k	n
PF_{Form}	3	5

Round 4

Table 1.22

PF_{Form}	k	n
PF_{Form}	4	15

Round 5

Table 1.23

PF_{Form}	k	n
PF_{Form}	5	1

Round 6

Table 1.24

PF_{Form}	k	n
PF_{Form}	6	11

Round 7

Table 1.25

PF_{Form}	k	n
PF_{Form}	7	6

Round 8

Table 1.26

PF_{Form}	k	n

PF_{Form}	8	13

Round 9

Table 1.27

PF_{Form}	k	n
PF_{Form}	9	6

Round 10

Table 1.28

PF_{Form}	k	n
PF_{Form}	10	10

Round 11

Table 1.29

PF_{Form}	k	n
PF_{Form}	11	2

Round 12

Table 1.30

PF_{Form}	k	n
PF_{Form}	12	7

Round 13

Table 1.31

PF_{Form}	k	n
PF_{Form}	13	9

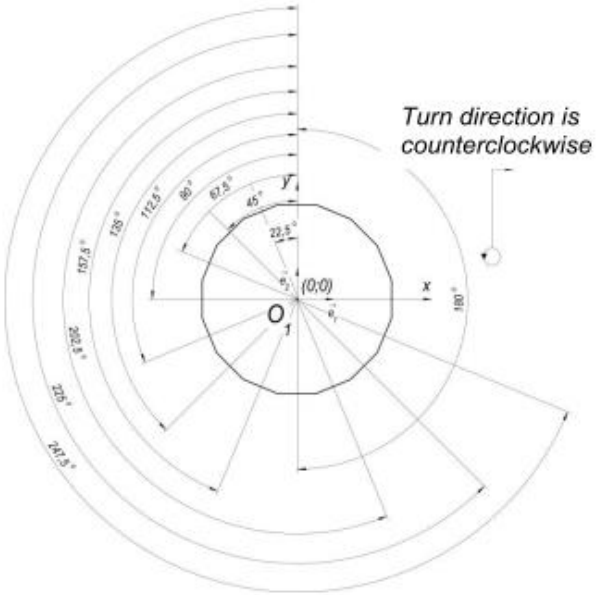


Figure 1.17 — Set of turn angles for PF_{Form}

Operation InvSubBytes

This transformation is inverted to the operation SubBytes. Round Operation InvSubBytes performs the bytes substitution in the state matrix by means of the reversed substitution table – Fig. 1.18

x/y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	72	40	6a	9b	91	64	30	7b	34	9e	81	8c	a8	6d
1	54	94	36	42	86	76	d0	3d	97	ff	43	a7	c4	65	d9	73
2	7c	90	39	32	a5	c2	2f	44	ee	ac	23	e4	a1	37	b6	9d
3	6c	a0	4c	46	70	b9	87	3f	b1	16	66	c0	c7	75	ba	84
4	3a	96	08	28	d5	68	98	24	4f	2e	d7	35	f3	de	5b	45
5	d4	5c	a4	aa	5d	ca	15	85	95	c8	00	49	50	da	cb	06
6	47	67	ab	b7	38	0b	58	05	dd	a2	3e	d1	b8	57	93	cd
7	5e	1e	cc	01	62	8f	89	6e	27	4d	b5	20	92	5f	9c	f4
8	bf	a3	a6	dc	11	fc	41	c1	f8	cf	f2	2c	f0	b4	c3	bb
9	e3	48	f7	1c	e7	71	ad	80	6f	07	ce	4a	e6	b2	e5	3c
a	1f	82	74	ae	e2	22	c5	0e	8e	29	2d	8b	f9	4b	e8	03
b	1d	51	c6	d2	19	0a	79	5a	9a	db	83	0d	59	8a	69	63
c	f6	88	60	56	d8	ed	fa	df	fd	12	e9	31	d3	ec	8d	25
d	f1	33	fb	0f	10	be	a9	fe	7f	04	7a	c9	4e	02	99	26
e	1a	3b	e0	f5	17	78	af	9f	ea	bd	2a	55	eb	14	7e	6b
f	bc	18	2b	77	b3	13	d6	0c	e1	53	b0	61	1b	ef	21	7d

Figure 1.18 —Table of substitutions of operation InvSubBytes

Operation InvPerBits

This conversion is the inverse of the PerBits conversion. The round operation InvPerBits swaps the bits in a byte. In the context of this algorithm, the operation is considered as a rotation around the clockwise hand of a single absolutely dynamic plafal. – PF_{ad}^{uniq}

Rotation work function $\omega(l)_{PF_{d}^{uniq}}$ [1, c 590] (it is the change of the coordinates of the vertices of a regular octagon in the frame in the context of the algorithm herewith):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{1.6}$$

$\begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}$ is the clockwise rotation matrix point coordinates gained by a rotation of point (x;y).

Table 1.32 — View of the rotation matrix from the number of turns

n	φ	$\begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}$
1	45°	$\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$
2	90°	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
3	135°	$\begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$
4	180°	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
5	225°	$\begin{bmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$
6	270°	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
7	315°	$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$

Rounds 1 – 14 are performed in a schedule 14 – 1 of Round operation PerBits.

Namely, the 1st Round of operation InvPerBits to 14th Round of operation PerBits, etc.

Operation InvShiftBytes

This transformation is the inverse of the ShiftBytes transformation. The round operation InvShiftBytes performs a cyclic shift of bytes in the state matrix by different values. In the context of this algorithm, it is considered as parallel transfer of PFj (p. 1.2.2) to PFi (p. 1.2.2); j ≠ i of $\mathrm{PF}\frac{doc}{S^{16}}$ complex. Parallel transfer means the “lucidification” of the system of plafales [1, c 592] $\mathrm{LUC}_{PF_{S^{16}}}$.

$$LUC_{PF_{S^{16}}} = \begin{cases} PF_{14} \to PF_2, PF_{10} \to PF_{14}, PF_6 \to PF_{10}, PF_2 \to PF_6, \\ PF_{11} \to PF_3, PF_7 \to PF_{15}, PF_3 \to PF_{11}, PF_{15} \to PF_7, \\ PF_8 \to PF_4, PF_4 \to PF_{16}, PF_{16} \to PF_{12}, PF_{12} \to PF_8. \end{cases}$$

The $\overrightarrow{O_jO_i} = -\overrightarrow{O_iO_j}$. serves as vector of mentioned transfer. Summarizing the above:

$$\begin{cases} R_{14} \to R_2, \overrightarrow{O_2O_{14}} = (-6, 0), \\ R_{10} \to R_{14}, \overrightarrow{O_{14}O_{10}} = (2, 0), \\ R_6 \to R_{10}, \overrightarrow{O_{10}O_6} = (2, 0), \\ R_2 \to R_6, \overrightarrow{O_6O_2} = (2, 0), \\ R_{11} \to R_3, \overrightarrow{O_3O_{11}} = (-4, 0), \\ R_7 \to R_{15}, \overrightarrow{O_{15}O_7} = (4, 0), \\ R_3 \to R_{11}, \overrightarrow{O_{11}O_3} = (4, 0), \\ R_{15} \to R_7, \overrightarrow{O_7O_{15}} = (-4, 0), \\ R_8 \to R_4, \overrightarrow{O_4O_8} = (-2, 0), \\ R_4 \to R_{16}, \overrightarrow{O_{16}O_4} = (6, 0), \\ R_{16} \to R_{12}, \overrightarrow{O_{12}O_{16}} = (-2, 0), \\ R_{12} \to R_8, \overrightarrow{O_8O_{12}} = (-2, 0). \end{cases}$$

Such configuration is executed in all rounds 1 – 14.

Operation InvPerBytes

This conversion is the inverse of the PerBytes conversion. The InvPerBytes operation performs permutation of bytes in the state matrix. In the context of this algorithm, it is considered as a clockwise rotation of absolutely dynamic plafal $\_PF_{ad}^{uniq}$.

Rotation work function: [1, p. 590] (there is a change of regular hexagon's vertices coordinates in the benchmark R1):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{1.7}$$

$\begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}$ is a matrix of clockwise rotation; are coordinates of point taken by a rotation of point (*x:y*).

Table 1.33 — View of turn matrix (depending) on turns number n

n	φ	$\begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix}$
1	22.5°	$\begin{bmatrix} 0.92 & 0.38 \\ -0.38 & 0.92 \end{bmatrix}$
2	45°	$\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$
3	67.5°	$\begin{bmatrix} 0.38 & 0.92 \\ -0.92 & 0.38 \end{bmatrix}$
4	90°	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
5	112.5°	$\begin{bmatrix} -0.38 & 0.92 \\ -0.92 & -0.38 \end{bmatrix}$
6	135°	$\begin{bmatrix} -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$

7	157.5°	$\begin{bmatrix} -0.92 & 0.38 \\ -0.38 & -0.92 \end{bmatrix}$
8	180°	$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$
9	202.5°	$\begin{bmatrix} -0.92 & -0.38 \\ 0.38 & -0.92 \end{bmatrix}$
10	225°	$\begin{bmatrix} -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}$
11	247.5°	$\begin{bmatrix} -0.38 & -0.92 \\ 0.92 & -0.38 \end{bmatrix}$
12	270°	$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$
13	292.5°	$\begin{bmatrix} 0.38 & -0.92 \\ 0.92 & 0.38 \end{bmatrix}$
14	315°	$\begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$
15	337.5°	$\begin{bmatrix} 0.92 & -0.38 \\ 0.38 & 0.92 \end{bmatrix}$

Rounds 2 – 14 are performed in order 13 – 1 of Round's operation PerBytes. Namely of 2nd Round's decryption procedure for InvPerBytes –13th Round's operation PerBytes, etc

Mode 1

Mode 1 completely preserves the architecture of the main mode, only differs from itin the operation PerBitsM. The specified operation is an extension of the PerBits operation PerBits.

.

Operation PerBitsM

The round operation PerBitsM permutes the bits in a byte. In the context of this algorithm, the operation is considered as a composition of two transformations, namely: 1.Operation PerBits; 2. Change of bit places (for example, for byte 10010010: bit {0} of the northeast side changed places with bit {1} of the southeast

side (p. 1.2.1). The total byte is 11000010. It should be noted that the received byte 11000010 cannot be obtained thanks to the PerBits operation.
The corresponding formula for the PerBitsM operation is as follows:

$$G(n,i,i',j,k)=\begin{cases} n\equiv f(k)\ \ (mod\,8),\ 8\nmid f(k),\\ F(i,i',j,k),\ 8\mid f(k), \end{cases}$$

$$F(i,i',j,k)=\begin{cases} i\leftrightarrow i',i\neq i',\\ i\equiv j\ \ (mod\,9),\\ i'\equiv k\ \ (mod\,9),\\ j=9\Rightarrow i=1,\\ k=\{9\}\Rightarrow i'\equiv j^2+1\ \ (mod\,9),\\ i=i'\Rightarrow\{i+1=i',\ i=\overline{1,7};\ i=8\Rightarrow i'=1\}, \end{cases}$$

f (k) - function of the number of turns, it has an individual form for each round; k is the position of PF_k in the cellular block structure (p. 1.2.2); j - round number, {i, i1} - sides of the plafale; $i \leftrightarrow i^1$ means that the bit of the i-th side is permuted with the bit of the i^1-th side. Let us give numerical examples for F (i, i^1, j, k). Let be {*j=5, k=6*} , Then

$$F(i,i',5,6)=\begin{cases} 5\leftrightarrow 6,\\ 5\equiv 5\ \ (mod\,9),\\ 6\equiv 6\ \ (mod\,9), \end{cases}$$

the bit on the 5th side is permuted with the bit on the 6th side.

Let {j= 5, k= 9}, then

$$F(i,i',5,9)=\begin{cases}5\leftrightarrow 8,\\ 5\equiv 5\pmod 9,\\ k=\{9\}\Rightarrow 8\equiv 5^2+1\pmod 9,\end{cases}$$

the bit on the 5th side is permuted with the bit on the 8th side

Let {j = 5, k = 5}, then.

$$F(i,i',5,5)=\begin{cases}5\leftrightarrow 6,\\ 5\equiv 5\pmod 9,\\ 5\equiv 5\pmod 9,\\ 5=5\Rightarrow\{i'=6\},\end{cases}$$

the bit on the 5th side is permuted with the bit on the 6th side.

Tables will be built for all Rounds (1 – 14) reflecting the functional performances Above.

Round 1

f(k)=k:

Table 1.34

PF_k	$f(k)$	n
PF_1	1	1
PF_2	2	2
PF_3	3	3
PF_4	4	4
PF_5	5	5
PF_6	6	6
PF_7	7	7
PF_8	8	$1\leftrightarrow 8$
PF_9	9	1

PF_{10}	10	2
PF_{11}	11	3
PF_{12}	12	4
PF_{13}	13	5
PF_{14}	14	6
PF_{15}	15	7
PF_{16}	16	$1 \leftrightarrow 7$

Round 2

$f(k)=k^2$:

Table 1.35

PF_k	$f(k)$	n
PF_1	1	1
PF_2	4	4
PF_3	9	1
PF_4	16	4
PF_5	25	1
PF_6	36	4
PF_7	49	1
PF_8	64	$2 \leftrightarrow 8$
PF_9	81	1
PF_{10}	100	4
PF_{11}	121	1
PF_{12}	144	4
PF_{13}	169	1
PF_{14}	196	4
PF_{15}	225	1
PF_{16}	256	$2 \leftrightarrow 7$

Round 3

$$f(k)=k+k^2+k^3:$$

Table 1.36

Round 4

PF_k	$f(k)$	n
PF_1	3	3
PF_2	14	6
PF_3	39	7
PF_4	84	4
PF_5	155	3
PF_6	258	2
PF_7	399	7
PF_8	584	$3 \leftrightarrow 8$
PF_9	819	3
PF_{10}	1110	6
PF_{11}	1463	7
PF_{12}	1884	4
PF_{13}	2379	3
PF_{14}	2954	2
PF_{15}	3615	7
PF_{16}	4368	$3 \leftrightarrow 7$

$$f(k)=k^2+k:$$

Table 1.37

PF_k	$f(k)$	n
PF_1	2	2
PF_2	6	6

PF_3	12	4
PF_4	20	4
PF_5	30	6
PF_6	42	2
PF_7	56	7
PF_8	72	$4 \leftrightarrow 8$
PF_9	90	2
PF_{10}	110	6
PF_{11}	132	4
PF_{12}	156	4
PF_{13}	182	6
PF_{14}	210	2
PF_{15}	240	7
PF_{16}	272	$4 \leftrightarrow 7$

Round 5

$$f(k)=k^3+k:$$

Table 1.38

PF_k	$f(k)$	n
PF_1	2	2
PF_2	10	2
PF_3	30	6
PF_4	68	4
PF_5	130	2
PF_6	222	6
PF_7	350	6

PF_8	520	5↔8
PF_9	738	2
PF_{10}	1010	2

PF_{11}	1342	6
PF_{12}	1740	4
PF_{13}	2210	2
PF_{14}	2758	6
PF_{15}	3390	6
PF_{16}	4112	5↔7

Round 6

$f(k)=[\pi^k]$:

Table 1.39

PF_k	$f(k)$	n
PF_1	3	3
PF_2	9	1
PF_3	31	7
PF_4	97	1
PF_5	306	2
PF_6	961	1
PF_7	3020	4
PF_8	9488	6↔8
PF_9	29809	1
PF_{10}	93648	2
PF_{11}	294204	4
PF_{12}	924269	5
PF_{13}	2903677	5
PF_{14}	9122171	3
PF_{15}	28658145	1

PF_{16}	90032220	4

Round 7

$$f(k)\equiv 8$$

Table 1.40

Round 8

PF_k	$f(k)$	n
PF_1	8	$7\leftrightarrow 1$
PF_2	8	$7\leftrightarrow 2$
PF_3	8	$7\leftrightarrow 3$
PF_4	8	$7\leftrightarrow 4$
PF_5	8	$7\leftrightarrow 5$
PF_6	8	$7\leftrightarrow 6$
PF_7	8	$7\leftrightarrow 8$
PF_8	8	$7\leftrightarrow 8$
PF_9	8	$7\leftrightarrow 5$
PF_{10}	8	$7\leftrightarrow 1$
PF_{11}	8	$7\leftrightarrow 2$
PF_{12}	8	$7\leftrightarrow 3$
PF_{13}	8	$7\leftrightarrow 4$
PF_{14}	8	$7\leftrightarrow 5$
PF_{15}	8	$7\leftrightarrow 6$
PF_{16}	8	$7\leftrightarrow 8$

$$f(k)= \sinh(k) :$$

Table 1.41

PF_k	$f(k)$	n
PF_1	1	1
PF_2	3	3

PF_3	10	2
PF_4	27	3
PF_5	74	2
PF_6	201	1
PF_7	548	4
PF_8	1490	2
PF_9	4051	3
PF_{10}	11013	5
PF_{11}	29937	1
PF_{12}	81377	1
PF_{13}	221206	6
PF_{14}	601302	6
PF_{15}	1634508	4
PF_{16}	4443055	7

Round 9

$f(k)=0.5\cdot(5k^3-3k)$:

Table 1.42

PF_k	$f(k)$	n
PF_1	1	1
PF_2	17	1
PF_3	63	7
PF_4	154	2
PF_5	305	1
PF_6	531	3
PF_7	847	7
PF_8	1268	4

PF_9	1809	1
PF_{10}	2485	5

PF_{11}	3311	7
PF_{12}	4302	6
PF_{13}	5473	1
PF_{14}	6839	7
PF_{15}	8415	7
PF_{16}	10216	$1 \leftrightarrow 7$

Round 10

$$f(k) = |k^4 - 6k^2 + 3| :$$

Table 1.43

PF_k	$f(k)$	n
PF_1	2	2
PF_2	5	5
PF_3	30	6
PF_4	163	3
PF_5	478	6
PF_6	1083	3
PF_7	2110	6
PF_8	3715	3
PF_9	6078	6
PF_{10}	9403	3
PF_{11}	13918	6
PF_{12}	19875	3
PF_{13}	27550	6
PF_{14}	37243	3
PF_{15}	49278	6
PF_{16}	64003	3

Round 11

$$f(k)=2k^2-1$$

Table 1.44

Round 12

PF_k	$f(k)$	n
PF_1	1	1
PF_2	7	7
PF_3	17	1
PF_4	31	7
PF_5	49	1
PF_6	71	7
PF_7	97	1
PF_8	127	7
PF_9	161	1
PF_{10}	199	7
PF_{11}	241	1
PF_{12}	287	7
PF_{13}	337	1
PF_{14}	391	7
PF_{15}	449	1
PF_{16}	511	7

$f(k)$= [exp(k)] :

Table 1.45

PF_k	$f(k)$	n
PF_1	2	2
PF_2	7	7
PF_3	20	4
PF_4	54	6
PF_5	148	4
PF_6	402	2
PF_7	1093	5
PF_8	2971	3
PF_9	8074	2
PF_{10}	21938	2
PF_{11}	59612	4
PF_{12}	161979	3
PF_{13}	440129	1
PF_{14}	1195920	6
PF_{15}	3249556	4
PF_{16}	8829694	6

Round 13

$f(k)$=k!!:

Table 1.46

PF_k	$f(k)$	n
PF_1	1	1
PF_2	2	2

PF_3	3	3
PF_4	8	4
PF_5	15	7
PF_6	48	6
PF_7	105	1
PF_8	384	$4 \leftrightarrow 8$
PF_9	945	1
PF_{10}	3840	2

PF_{11}	10395	3
PF_{12}	46080	4
PF_{13}	135135	7
PF_{14}	645120	6
PF_{15}	2027025	1
PF_{16}	10321920	$4 \leftrightarrow 7$

Round 14

$f(k)=k$:

Table 1.47

PF_k	$f(k)$	n
PF_1	1	1
PF_2	2	2
PF_3	3	3
PF_4	4	4
PF_5	5	5
PF_6	6	6
PF_7	7	7
PF_8	8	$5 \leftrightarrow 8$
PF_9	9	1
PF_{10}	10	2

PF_{11}	11	3
PF_{12}	12	4
PF_{13}	13	5
PF_{14}	14	6
PF_{15}	15	7
PF_{16}	16	$5 \leftrightarrow 7$

Key Schedule

Round keys are obtained from the encryption key through a key generation algorithm. It contains two components: key expansion and round key selection. The underlying principles of the algorithm are as follows:

1. The total number of bits of the round keys is equal to the block length multiplied by the number of rounds, plus 1. That is, 128 • (14 + 1) = 1920 bits = 240 bytes. The size of each of the fifteen round keys is 16 bytes.
2. The encryption key is expanded into an extended key.
3. The round keys are taken from the extended key as follows: the first round key contains the first 16 bytes, the second - the next 16 bytes, and so on

ExpandKey Procedure

The extended key is a linear array of 4-byte words W [Nb (Nr + 1)]. The first eight words contain the encryption key. All other words are determined recursively from words with lower indices. The pseudocode is shown below.

```
KeyExpansion(CipherKey,W)
 {
     for (i=0; i<Nk; i++) W[i]=CipherKey[i];
     for (j=Nk; j<Nb*(Nr+1); j+=Nk)
     {
             W[j]=W[j-Nk] xor SubByte(PerBits(W[j-1])) xor Rcon[j/Nk];
             for (i=1; i<4; i++) W[i+j]=W[i+j-Nk] xor W[i+j-1];
             W[j+4]=W[j+4-Nk] xor SubByte(W[j+3]);
             for (i=5; i<Nk; i++) W[i+j]=W[i+j-Nk] xor W[i+j-1];
     }
 }
```

Round functions PerBits are executed over words in the PerBits $(W[j-1])$ operation.

$$N = \begin{cases} N(W[j/Nk]) \equiv j/Nk \pmod{14},\ 14 \nmid j/Nk, \\ 1,\ 14 \mid j/Nk, \end{cases}$$

$N(W[j/Nk])$ Rounds operation PerBits for a word $W[j/Nk]$. For example, for $W[16] : N(W[j/Nk]) = 2$ Wherein, the correspondence takes place for word's bytes relatively to a rounds number:

$a_{33+32i} \leftrightarrow \mathrm{PF}_1,\ a_{34+32i} \leftrightarrow \mathrm{PF}_2, a_{35 \mid 32i} \leftrightarrow \mathrm{PF}_3,\ a_{36 \mid 32i} \leftrightarrow \mathrm{PF}_4,\ i = \overline{1,6}$

Round's constant is determined by a following way:

$$Rcon[i] = (RC[i], 00, 00, 00),$$

Where $RC[i] \in GF(2^8)$, wherein $RC[0] = 1,\ RC[i] = xtime(Rcon[i-1]) = x^i.$

AddRoundKey Procedure

The round key state matrix is added to the block state matrix in this operation by a simple bitwise XOR (using the square matrix addition rules). For the encryption procedure, the first sixteen bytes of the encryption key are added to the plaintext matrix, and so on. For the decryption procedure, the round keys are used in reverse order.

Notes

We take n = 0 for an angle of 360◦ in the PerBits operation, without changing the generality.

According to the execution logic, the InvPerBitsM operation is similar to the InvPerBits operation.

Cryptic Resistance

Symmetric cryptographic algorithm “STEEL” is a modification of the symmetric algorithm “ECLECTIC-DT-1” [2]. Accurate substantiation of estimates of the practical stability of the STEEL algorithm relative to the methods of differential and linear cryptanalysis requires the use of a specific mathematical apparatus [2-13]. Analytical upper bounds are obtained for the parameters characterizing the practical stability of the algorithm with respect to the methods of differential and linear cryptanalysis. With regard to “STEEL”, the values of the upper estimates of the average probabilities of the differential and linear characteristics are 2^{-595} and 2^{-552}, respectively, which indicates the practical resistance of this algorithm to classical differential and linear attacks [14, 15].
It is also shown that the group of substitutions generated by round transformations of the “STEEL” algorithm is primitive. This excludes the possibility of carrying out a number of algebraic attacks on the cipher based on homomorphism [16 - 18].

Model setting

For ∀ m ∈ N, we define Vm, the set of binary vectors of length m. t, q, c ≥ 2, r ≥ 4, 4 • q ≤ 2t − 1 + 1, and put p = c • q, n = p • t.

$$\{n = 128,\ r = 14,\ p = 16,\ t = 8,\ c = 4,\ q = 4,\ 16 \le 2^{8-1} + 1 = 2^7 + 1\}.$$

We define on Vt = V8 a field structure of order 2t = 28 consistent with the coordinatewise Boolean addition of binary vectors. This operation will be denoted by the symbol ⊕ (regardless of the lengths of the vectors x, y ∈ Vm, which are summed). In what follows, the set Vt = V8 will be associated with the field GF (2t) = GF (28), and the set Vmt - with an m-dimensional vector space over this field. ∀ x ∈ Vn = GF (2t) p will be written in the form:

$$x=(x_1,\dots,x_c),\ \text{where}\ x_j = (x_{1,j},\dots,x_{q,j}),\ x_{i,j} \in V_t = GF(2^t),\ i \in \overline{1,q},\ j \in \overline{1,c}$$

Vectors $x_1,\dots,x_c$ will be mentioned as Components hereafter, and elemants $x_{1,1},\dots,x_{q,1},\dots,x_{1,c},\dots,x_{q,c}$ as vector coordinates of x.

Regarding to the algorithm herewith we have:

$V_n = V_{128} = V_{168}$, :he 16-dimensional vector space;
$\forall\ x \in V_{128} = GF(2^8)^{16}$: $x = (x_1, x_2, x_3, x_4)$ are the components:
$x_1 = (x_{1,1}, \ldots, x_{4,1})$; $x_2 = (x_{1,2}, \ldots, x_{4,2})$; $x_3 = (x_{1,3}, \ldots, x_{4,3})$;
$x_4 = (x_{1,4}, \ldots, x_{4,4})$ are coordinates of vector x, $i \in \overline{1,4}$; $j \in \overline{1,4}$
Lets fix the family of substitutions $s_{i,j} : V_8 \to V_8$, $i \in \overline{1,4}$; $j \in \overline{1,4}$.
Lets set the substitutions sj : $V_{8\text{-}4} \to V_{8\text{-}4}$ and $s : V_{128} \to V_{128}$, considering:

$$s_j(x_{1,j}, \ldots, x_{4,j}) = (s_{1,j}(x_{1,j}), \ldots, s_{4,j}(x_{4,j})), x_{i,j} \in V_8,\ i \in \overline{1,4},\ j \in \overline{1,4}, \quad (2.1)$$

$$s(x) = (s_1(x_1), s_2(x_2), s_3(x_3), s_4(x_4)), x \in V_{128}. \quad (2.2)$$

Lets fix the permutation on the set $\overline{1,4} \times \overline{1,4}$
And put for $\forall\ x \in V_{128} = = GF(2_8)^{16}$:

$$\widehat{g}(x) = (x_{g(1,1)}, \ldots, x_{g(4,1)}, \ldots, x_{g(1,4)}, \ldots, x_{g(4,4)}) \quad (2.3)$$

$\widehat{g}$ is a linear transformation of vector space $GF(2^8)^{16}$, which carries out permutation of coordinates $x_{i,j}$ of an arbitrary vector x in accordance with permutation g and be represented as follows:

$$\widehat{g}(x) = xG_{16},\ x \in GF(2^8)^{16} \quad ,2.4$$

Where G_{16} is a determined substitution matrix 16×16. The G_{16} matrix corresponds with the permutation g.
Let's define:

$$M_{16} = G_{16} D, \quad (2.5)$$

$$\varphi(x) = s(x)M_{16},\ x \in GF(2^8)^{16}, \quad (2.6)$$

is the MDR-matrix of the 16th order over the field $GF(2^8)$ [10,19]; Consider a 14-round block cipher $\Im$ with a set of open (encrypted) messages V_{128}, a set of round keys $K = V_{128}$, and a family of encryption transformations:

$$F_k = f_{14,k_{14}} \circ \ldots \circ f_{1,k_1}, k = (k_1, \ldots, k_{14}) \in K^{14}, \quad (2.7)$$

where $\forall\ x \in V_{128},\ k \in K,\ i \in \overline{1,14}$

$$f_{i,k}(x) = \varphi(x \oplus k). \quad (2.8)$$

Let us call the mapping ϕ of the form (2.6) the round function of the cipher $\Im$, and the substitutions $s_{i,j}$ $(i \in \overline{1,4},\ j \in \overline{1,4})$ replacement nodes (in the context of the

algorithm, the composition of the operations SubBytes and PerBits (M)). The incoming (outgoing) message x ∈ V128 is identified with the state matrix (Section 1.2.4), which consists of the coordinates xi, j of the vector x. The permutation g is determined by the formula

$$g(i,j) = \begin{cases} (i,j), i = 1, \\ (i, -\frac{2}{3} \cdot j^3 + 6 \cdot j^2 - \frac{49}{3} \cdot j + 15), i = 2, \\ (i, \frac{4}{3} \cdot j^3 - 10 \cdot j^2 + \frac{65}{3} \cdot j - 10), i = 3, \\ (i, -\frac{2}{3} \cdot j^3 + 4 \cdot j^2 - \frac{19}{3} \cdot j + 5), i = 4. \end{cases}$$

Using formulas (2.3), (2.4) and the ShiftBytes operation (section 1.2.9), we define G_{16}.

$$G_{16} = \begin{pmatrix} 1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0 \end{pmatrix}.$$

$\Omega = (\omega_0, \omega_1, \ldots, \omega_r) \in (V_n \backslash \{0\})^{r+1}$ is the probability of the differential characteristic [7] of the block cipher $\Im$ with the encryption key (k_1,..., k_r) which is determined by formula:

$$DP^{(k_1,\ldots,k_r)}(\Omega) = P(\bigcap_{i=1}^{\cdot} \{X_i \boxplus X_i' = \omega_i\} | X \boxplus X' = \omega_0), \tag{2.9}$$

where X, X' are independent random equiprobable binary vectors of length n,

$$X_i = (f_{i,k_i} \circ \ldots \circ f_{1,k_1})(X),$$
$$X'_i = (f_{i,k_i} \circ \ldots \circ f_{1,k_1})(X'),\ i \in \overline{1,r}.$$

The average value (2.9) over all (k_1,..., k_r) ∈ $\mathbf{K}^r$ is called the average probability of the differential characteristic Ω and is denoted by EDP (Ω). Thus

$$EDP(\Omega) = |K|^{-r} \sum_{(k_1,\ldots,k_r)\in K^r} DP^{(k_1,\ldots,k_r)}(\Omega \qquad (2.10)$$

Average probability of the linear characteristic Ω = (ω0, ω1,..., ω r) binary cypher is determined by the formula [7]::

$$ELP(\Omega) = \prod_{i=1}^{r} l^{(i)}(\omega_{i-1}, \omega_i) \qquad (2.11)$$

Where for any $\alpha, \beta \in V_n,\ i \in \overline{1,r}$

$$l^{(i)}(\alpha,\beta) = 2^{-n} \sum_{k\in V_n} (2^{-n} \sum_{x\in V_n} (-1)^{\alpha x \oplus \beta f_{i,k}(x)})^2$$

Parameters (2.10), (2.11) are standard indicators of the practical strength of block ciphers with respect to the methods of difference and linear cryptanalysis, respectively [7, 14, 15]. The stability of the Binary Cypher $\Im$ with respect to the method of homomorphisms is determined by the properties of the permutation group $G(\Im)$ generated by its round encryption transformations (2.8). In particular, the primitiveness of the group $\Im$ is a sufficient condition for the strength of the cipher $G(\Im)$ against the attacks described in [16 - 18]. Thus, to estimate the practical security of the considered block cipher with respect to the indicated cryptanalysis methods, it is required to obtain analytical upper bounds for parameters (2.10), (2.11) and check whether substitutions of the form (2.8) generate a primitive group

Characteristics of difference and correlation properties

$$d_{\oplus}^{s_{i,j}} = \max\{d_{\oplus}^{s_{i,j}}(\alpha,\beta) : \alpha,\beta \in V_8 \backslash \{0\}\}, \qquad (2.12)$$

$$l_{\oplus}^{s_{i,j}} = \max\{l_{\oplus}^{s_{i,j}}(\alpha,\beta) : \alpha,\beta \in V_8\backslash\{0\}\}, \quad (2.13)$$

$$\Lambda^{s_{i,j}} = \max\{\Lambda^{s_{i,j}}(\alpha,\beta) : \alpha,\beta \in V_8\backslash\{0\}\}, \quad (2.14)$$

where

$$d_{\oplus}^{s_{i,j}}(\alpha,\beta) = 2^{-8}\sum_{k\in V_8}\delta(s_{i,j}(k\ominus\alpha)\ominus s_{i,j}(k),\beta), \quad (2.15)$$

$$l_{\odot}^{s_{i,j}}(\alpha,\beta) = 2^{-8}\sum_{k\in V_8}(2^{-8}\sum_{x\in V_8}(-1)^{\alpha x\oplus\beta s_{i,j}(x\oplus k)})^2, \quad (2.16)$$

$$\Lambda^{s_{i,j}}(\alpha,\beta) = 2^{-8}\sum_{k\in V_8}(2^{-8}\sum_{a\in\{0,1\}}\left|\sum_{x\in V_8:\nu(x,k)=a}(-1)^{\alpha x\oplus\beta s_{i,j}(x\ominus k)}\right|)^2. \quad (2.17)$$

Parameter (2.12) characterizes the difference properties of the substitution si, j with respect to the operation ⊕, and parameters (2.13) and (2.14) characterize its correlation (linear) properties with respect to this operation Matrices of dimension (2^8 - 1) × (2^8 - 1), which consist of elements (2.15) and (2.16), where α and β take all nonzero values from the set V_8, are called the table of differences and the table of linear approximations of the substitution $s_{i,j}$ respectively. A vector α ∈ V_8 is called a linear translator of a function f: V_8 → {0, 1} if the equality f (x ⊕ α) = f (x) holds for an arbitrary x ∈ V_8. A substitution has a trivial linear structure [20] if for every nonzero linear combination of its coordinate functions there are no nonzero linear translators. It is known [21] that for the triviality of the linear structure of the substitution $s_{i,j}$: V_8 → V_8, it suffices to satisfy the condition NW ($s_{i,j}$) <2, where NW (si, j) is the maximum number of zero elements in the columns of the table of linear approximations of this substitutions.

The parameters values (2.12) – (2.14) and value of $NW^{(si,j)}$ are presented in the table 2.1

Table 2.1

$d_{\ominus}^{s_{i,j}}$	$l_{\oplus}^{s_{i,j}}$	$\Lambda^{s_{i,j}}$	$NW^{(s_{i,j})}$
2^{-5}	2^{-5}	0.04	37

According to [22] a nonnegative matrix P of order n is called indecomposable, if such substitution matrix does not exist, which:

$$RPR^{-1} = \begin{pmatrix} P_1 & 0 \\ P_2 & P_3 \end{pmatrix}$$

where P_1 is a square matrix of order less than n. A matrix P is called completely indecomposable if there are no permutation matrices R_1 and R_2 such that

$$R_1 P R_2 = \begin{pmatrix} P_1 & 0 \\ P_2 & P_{3.} \end{pmatrix}$$

It is known that a doubly stochastic matrix P is completely indecomposable if and only if the matrix P • PT is indecomposable.

The matrix $D^{(s_{i,j})} = D^{(s_{i,j})} \cdot (D^{(s_{i,j})})^1$ is the positive one [23],

Where $D^{(s_{i,j})} =: (d_{\odot}^{s_{i,j}}(\alpha, \beta))_{\alpha,\beta \in V_8 \setminus \{0\}}$

The positively defined matrix $\tilde{D}^{(s_{i,j})}$ is equipotential to existence of such $\forall\, u_1, u_2, \nu_1, \nu_2 \in V_{8.}$, where $u_1 \neq u_2, \nu_1 \neq \nu_{2.}$, that $k, k', k'' \in V_8$

$s_{i,j}(u_1 \odot k) \odot k' = = s_{i,j}(\nu_1 \ominus k'')$, $s_{i,j}(u_2 \oplus k) \oplus k' = s_{i,j}(\nu_2 \oplus k'')$

The noted fact testifies to the acceptable mixing properties of the replacement nodes [24]. For an arbitrary natural m and an arbitrary vector $z = (z^{(1)}, \ldots, z^{(m)}) \in GF(2^t)^m$ we denote by wt (z) the weight of the vector z:

$$wt(z) = \#\{i \in \overline{1,m} : z^{(i)} \neq 0\}$$

The branch number of the m × m matrix M over the field GF (2^t) is by definition a number [10] (internal branching index):

$$B_M = \min\{wt(z) + wt(zM) : z \in GF(2^t)^m \setminus \{0\}\}.$$

While D is the MDR matrix (damage-resistant matrix) of the 16th order over the field GF(2^8), we have [2, 3, 10,23]:

$$B_D = B_{D^T} = 16 + 1 = 17.$$

We find the maximum external ramification index B'_{L16} using results of [23, 25 – 27]:

$$B'_{L_{16}} = \mu(B_M) + 1 = 1 + 1 = 2.$$

Where μ(BM)≡1 is the number of MDR matrices used in cipher subblocks.

Let us find upper bounds for the parameters (2.10) and (2.11) characterizing the practical security of the cipher under consideration with respect to the methods of difference and, accordingly, linear cryptanalysis, using the results [3, 13, 23, 25]:

$$EDP(\Omega) \le (d_{\oplus}^{s_{i,j}})^{\lfloor \frac{r}{2} \rfloor \cdot B_D} = 2^{-5 \cdot (\lfloor \frac{14}{2} \rfloor \cdot 17)} = 2^{-595},$$

$$ELP(\Omega) \le (\max\{\Lambda^{s_{i,j}}, l_{\oplus}^{s_{i,j}}\})^{\lfloor \frac{r}{2} \rfloor \cdot B_D} = 2^{-4.644 \cdot (\lfloor \frac{14}{2} \rfloor \cdot 17)} = 2^{-552}.$$

A nonnegative square matrix P is called primitive if there is a natural number l such that $P^l > 0$. The smallest natural number l with the indicated property is called the exponent of the matrix P [28].
We denote by B = (b_{ij}) 16 × 16 the support of the matrix M_{16}, that is, the real (0, 1) - matrix with elements: $b_{ij} = 1$, if $m_{ij} \neq 0$; $b_{ij} = 0$ - otherwise.
The following statement is verified directly. When p= 16 the support of M_{16} is a primitive matrix.

Group generated by round transformations

Let us define the $G_{\Im} = \langle f_{i,k} : i \in \overline{1,14}, k \in V_{128} \rangle$, which is the permutation group generated by round transformations.
In [24], sufficient conditions were obtained under which $G_{\Im}$ is an alternating permutation group on the set V_{128}:
The support of the matrix M_{16} is a primitive matrix (the condition was fulfilled above).
•• for $\forall\, i \in \overline{1,14}, j \in \overline{1,14}$ the permutation group

$$G(s_{i,j}) = \langle s_{i,j}^{\alpha,\beta} : \alpha, \beta \in V_8 \rangle$$

Where , is 2× transitive and there is a permutation s i, j among elements of these groups, such that

••• the inequality 2pt <(2t - 1) p − 1 (2t + 2t − 1 - 2) is satisfied. As applied to the algorithm, we
have: $2^{128} = 3.4 \cdot 10^{38} < (2^8 - 1)^{16-1} (2^8 + 2^{8-1} - 2) = 4.7875 \cdot 10^{38}$. Then the group $G_{\Im}$ consists of all even permutations on the set V_{128}.

According to [24], the 2-transitivity of the group G ($s_{i,j}$) is equivalent to the fact that the matrix $\tilde{D}^{(s_{i,j})} = D^{(s_{i,j})} \cdot (D^{(s_{i,j})})^T$ is indecomposable (condition was performed above).
The difference table contains elements $d_{\ominus}^{s_{i,j}}(\alpha, \beta)$, which are equal to 2−8·6.
Accordingly, the ratio is fulfilled:

$$6 = 2^8 d_{\oplus}^{s_{i,j}}(\alpha, \beta) = \sum_{k \in V_8} \delta(s_{i,j}(k \boxplus \alpha) \boxplus s_{i,j}(k), \beta) =$$

$$= \#\{k \in V_8 : s_{i,j}(k) = k\} \notin \{0, 2^0, 2^1, \ldots, 2^8\}.$$

and •• is fully implemented.
Thus, the substitution group generated by round transformations with a block length of 128 bits is sign-alternating on the set V_{128}.

Theoretical security characteristics (provable security)

We obtain from formula (2.7) that $\Im$ is a product of ciphers $\Im_i$ and on the basis of formula (2.8) $\Im_{[1,14]}$ is a Markov one (with respect to operation $\oplus$) SP-cipher.
For 4 rounds, let us display the maximum indicators (estimates) of the theoretical security (provable security) [2, 23, 25 - 27]:

$$d_{\oplus,\oplus}^{\Im_{[1,14}}(\alpha, \beta) \leq (d_{\oplus}^{s_{i,j}})^{(B_D - 1)\cdot(B'_{L_{16}} - 1)} = (2^{-5})^{16 \cdot 1} = 2^{-80},$$

$$l^{\Im}(\alpha, \beta) \leq (l_{\oplus}^{s_{i,j}})^{(B_D - 1)\cdot(B'_{L_{16}} - 1)} = (2^{-5})^{16 \cdot 1} = 2^{-80}.$$

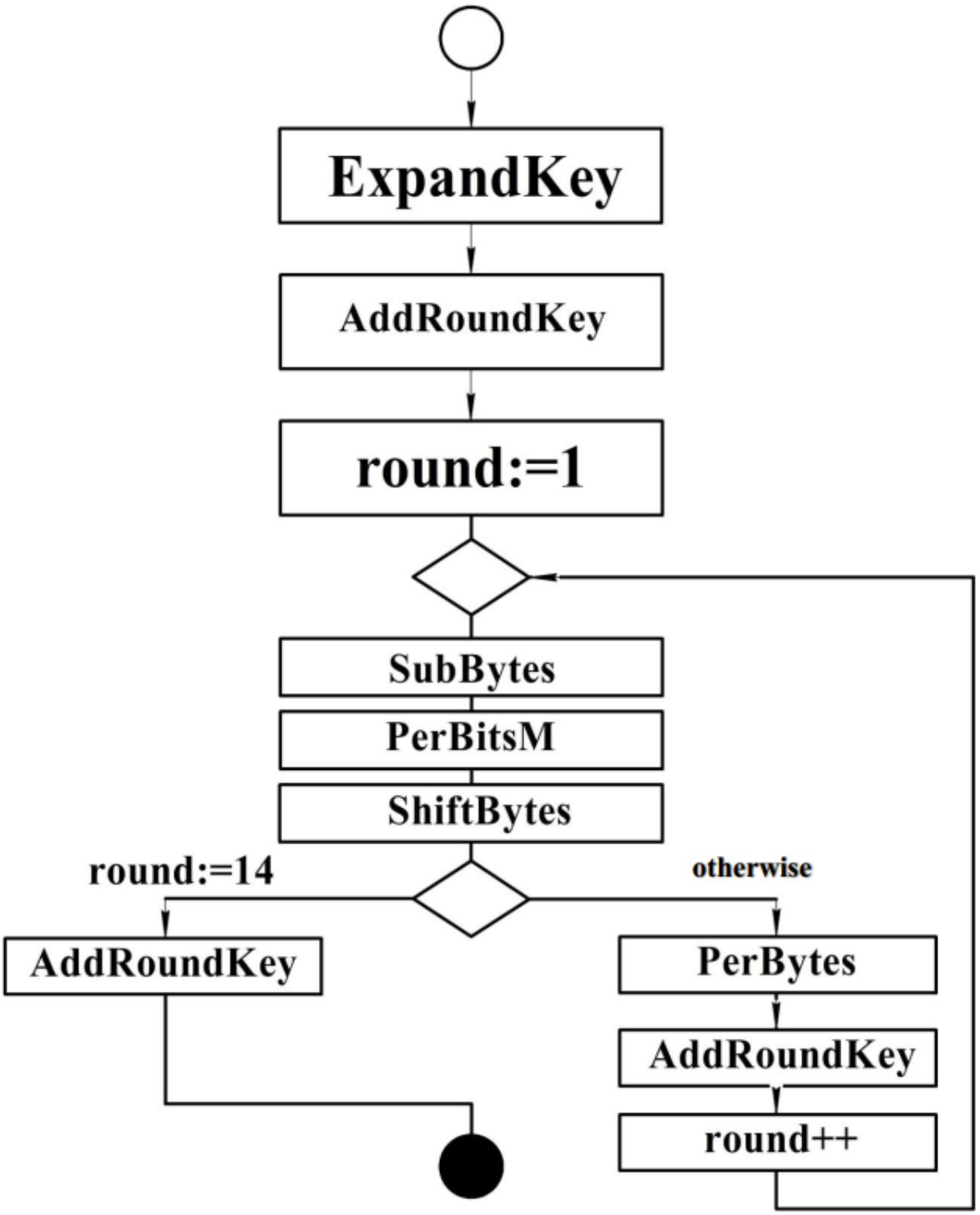
References

1. Topchyi D. The theory of plafales: the proof algorithms for millennium problems. – Best Global Publishing, 2013. – 695p.
2. Topchyi D. The theory of plafales: cryptographic complex «ECLECTIC-DT-1».– Chipmunkapublishing, 2015. – 65p.
3. A.N. Alekseychuk, L.V. Kovalchuk, E.V. Skrynnik, A.S. Shevtsov Estimates of the practical security of the block cipher "KALINA" relative to the methods of difference, linear cryptanalysis and algebraic attacks based on homomorphism // Applied Radioelectronics ... - 2008. - Volume 7 - Issue. 3. - P. 203 - 209.
4. A.N. Alekseychuk, L.V. Kovalchuk Upper bounds of the maximum values of the probabilities of differential and linear characteristics of the Feistel cipher containing an adder modulo 2^m // Applied Radioelectronics. - 2006. - Volume 5. - No. 1. - P. 74 - 82.
5. A.N. Alekseychuk, L.V. Kovalchuk, S.V. Palchenko Cryptographic parameters of replacement nodes that characterize the strength of GOST-like block ciphers with respect to methods of linear and differential cryptanalysis // Information Security. - 2007. - No. 2. - P. 12 - 23.
6. L.V. Kovalchuk Generalized Markov's ciphers: constructing an assessment of practical security with respect to differential cryptanalysis // Mathematics and security of information technologies. Proceedings of the conference at Moscow State University October 25 - 27, 2006 - M .: MCNMO, 2007. - P. 595 - 599.
7. Vaudenay S. On the security of CS-cipher // Fast Software Encryption. - FSE'99, Proceedings. - Springer Verlag, 1999. - P. 260-274.
8. Biryukov A. Block ciphers and stream ciphers: the state of the art 2004/094.
9. Vaudenay S. Decorrelation: a theory for block cipher security // J. of Cryptology. - 2003. - V. 16. - No. 4. - P. 249 - 286.
10. Daemen J. Cipher and hash function design strategies based on linear and differential cryptanalysis. - Doctoral Dissertation, 1995.
11. Daemen J., Rijmen V. Statistics of correlation and differentials in block ciphers // http://eprint.iacr.org/ 2005/212.
12. Kanda M., Takashima Y., Matsumoto T., Aoki K., Otha K. A strategy for constructing fast round functions with practical security against differential and linear cryptanalysis // Selected Areas in Cryptography. - SAC 1998, Proceedings. – Springer Verlag, 1999. - P. 264 - 279.
13. Kanda M. Practical security evaluation against differential and linear cryptanalyses for Feistel ciphers with SPN round function // Selected Areas in Cryptography. - SAC

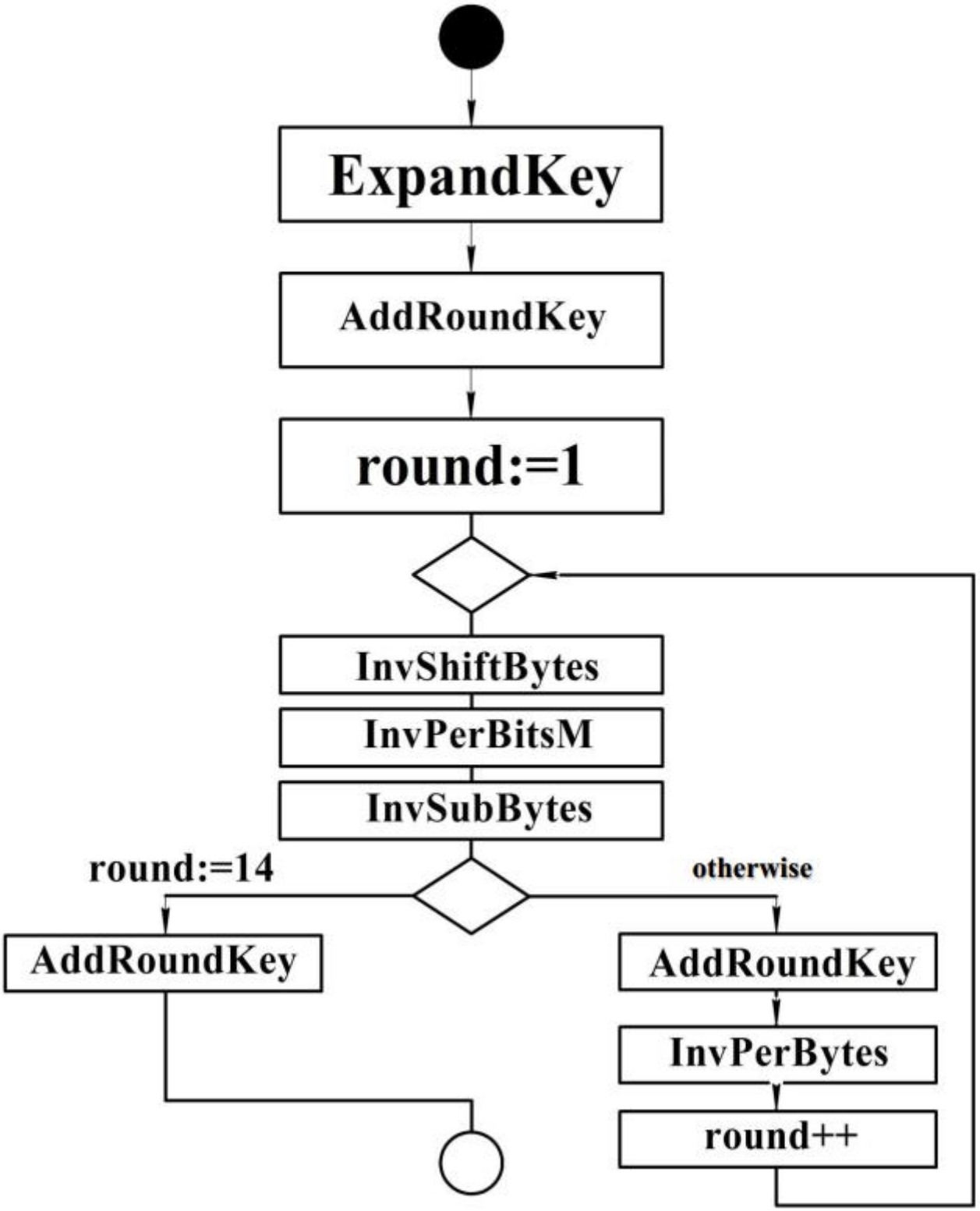
2000, Proceedings. - Springer Verlag, 2001. - P. 324 - 338.
14. 14. Lai X., Massey J. L., Murphy S. Markov's ciphers and differential cryptanalysis // Advances in Cryptology - EUROCRYPT'91, Proceedings. - Springer Verlag, 1991. - P. 17 - 38.
15. 15. Matsui M. Linear cryptanalysis methods for DES cipher // Advances in Cryptology - EUROCRYPT'93, Proceedings. - Springer Verlag, 1994. - P. 386 - 397.
16. 16. Campbell K. W., Wiener M. DES is not a group // Advances in Cryptology - CRYPTO'92, Proceedings. - Springer Verlag, 1993. - P. 512-520.
17. 17. Kaliski B. S., Rivest R. L., Sherman A. T. Is the Data Encryption Standard a group? (Results of cycling experiments on DES) // Journal of Cryptology. - 1988. - No. 1. - P. 3 - 36.
18. 18. Paterson K. G. Imprimitive permutation groups and trapdoors in iterated block ciphers // Fast Software Encryption. - FSE'99, Proceedings. - Springer Verlag, 1999. - P. 201-214.
19. 19. McWilliams F. J. Theory of error-correcting codes. - M .: Communication, 1979 .-- p. 743
20. 20. A.N. Alekseychuk A criterion for primitiveness of a permutation group generated by round transformations of a Rijndael-like block cipher // Registration, storage and processing of data. - 2004. - Volume 6. - No. 2. - p. 11 - 18.
21. 21. A.N. Alekseychuk Classes of mappings with a trivial linear structure over a finite field // Registration, storage and processing of data. - 2008. - Volume 10. - No. 3. - p. 80 - 88.
22. 22. V.N. Sachkov Introduction to combinatorial methods of discrete mathematics. - M .: MNTSNMO, 2004 .-- p. 424
23. 23. A.N. Alekseychuk, L.V. Kovalchuk et al. Research results of the cryptographic properties of the encryption algorithm "KALINA" // Collection of scientific works "Special telecommunication systems and information security". - Issue 1 (25). – 2014 .-- p. 5 - 23.
24. 24. A.S. Maslov, "On conditions for generating an alternating group by SAsubstitutions," Tr. Institute of Mathematics. - 2007. - Volume 15. - No. 2. - p. 58 - 68.
25. 25. 25. Ju-Sung Kang et al. Practical and Provable Security against Differential and Linear Cryptanalysis for Substitution-Permutation Networks // ETRI Journal. - Volume 23. - Number 4. - December 2001. - P. 158 - 167.
26. 26. Sangwoo Park et al. On the Security of Rijndael-like Structures against Differential and Linear Cryptanalysis // Advances in Cryptology - ASIACRYPT 2002. – Lecture Notes in Computer Science. - Volume 2501. - 2002. - P. 176 - 191.

27. Sangwoo Park et al. Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES // Fast Software Encryption. - Lecture Notes in Computer Science. - Volume 2887 .-- 2003 .-- P. 247 - 260.
28. V. N. Sachkov Combinatorics of non-negative matrices. - M .: TVP, 2000 .-- p. 447

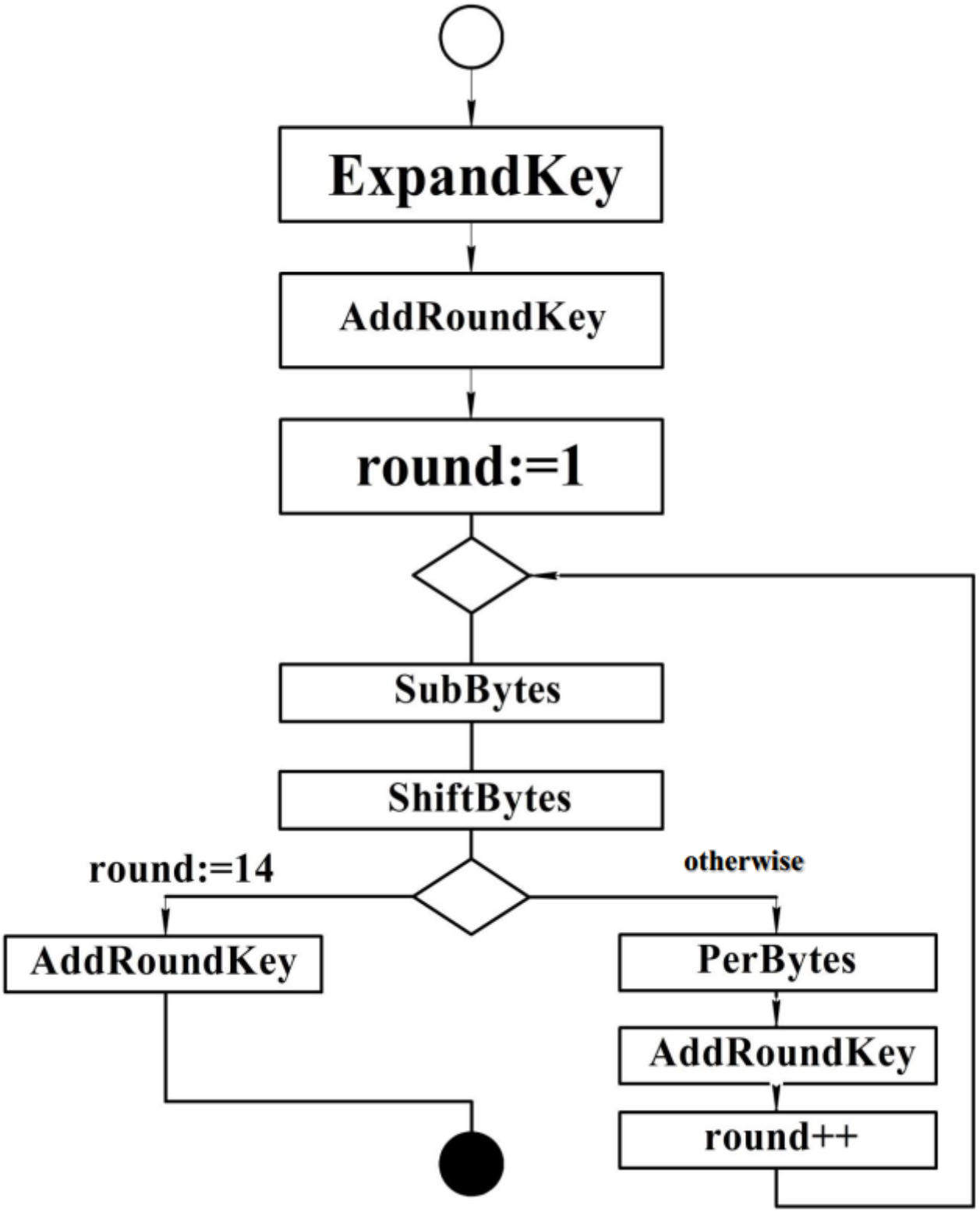
APPENDIX A MODE 1. ENCRYPTION



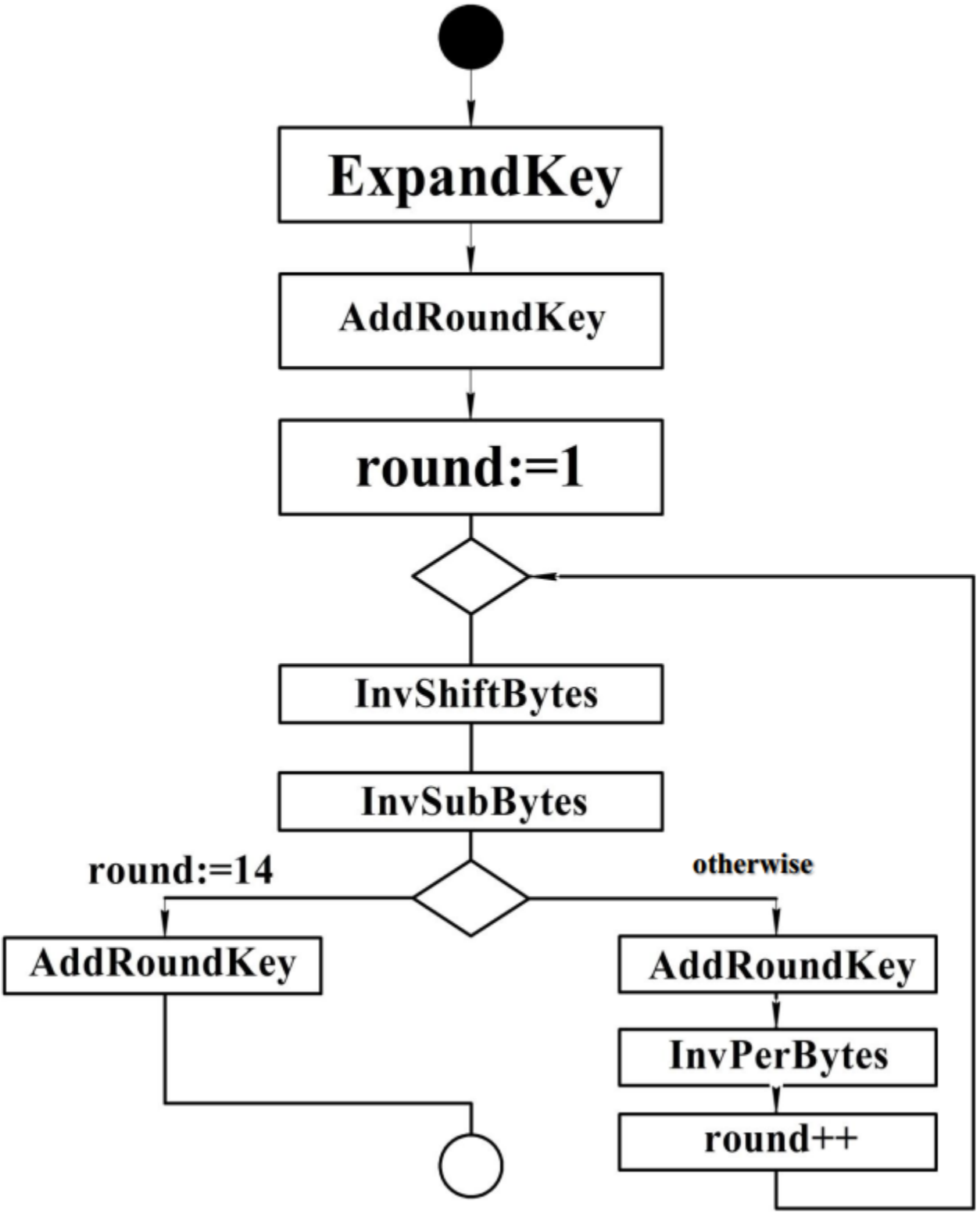
APPENDIX B MODE 1. DECRYPTION



APPENDIX C MODE 2. ENCRYPTION



APPENDIX D MODE 2. ENCRYPTION



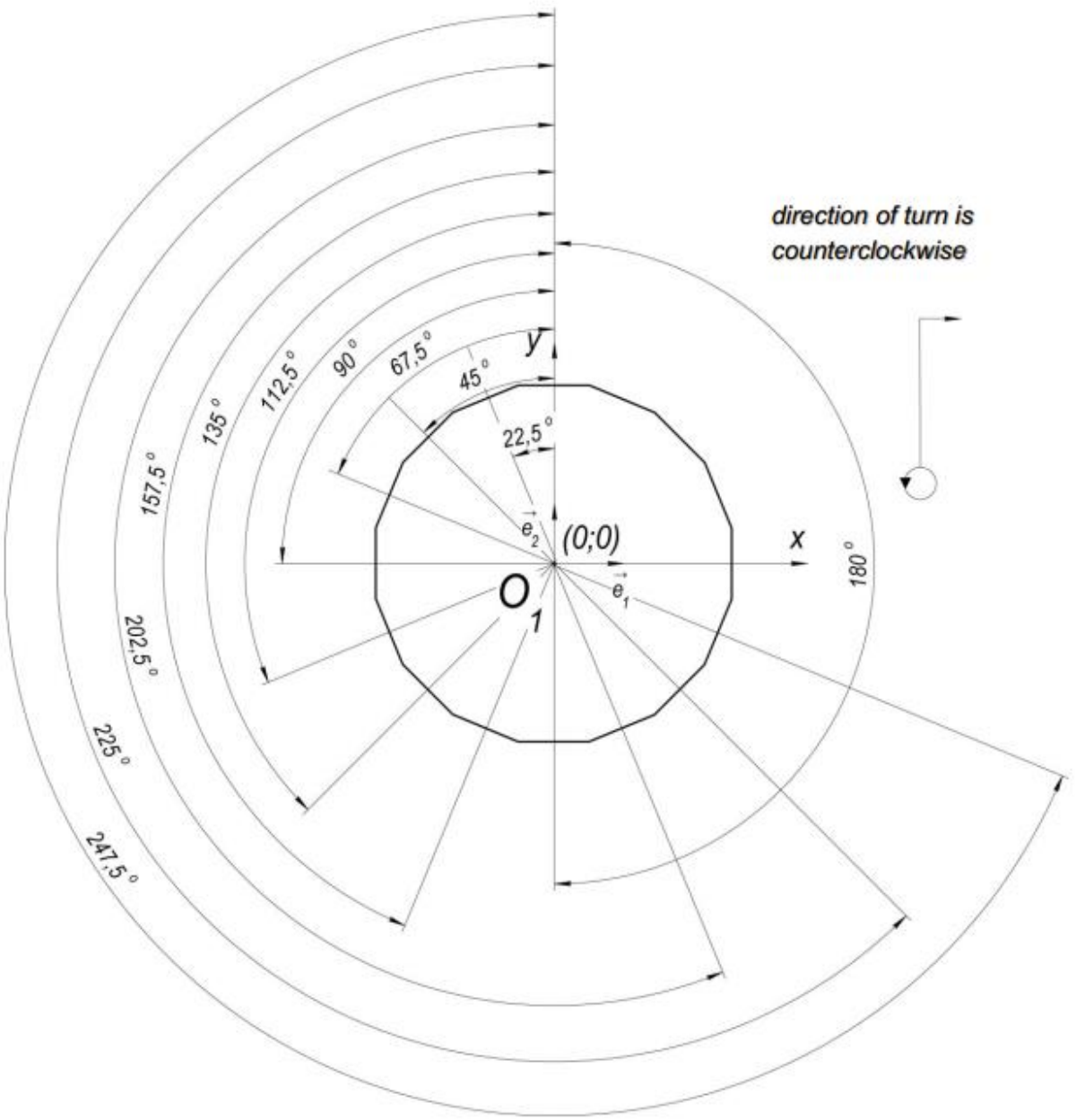
APPENDIX E SUBBYTES OPERATIONS TABLE

x/y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	5a	73	dd	af	d9	67	5f	99	42	01	b5	65	f7	bb	a7	d3
1	d4	84	c9	f5	ed	56	39	e4	f1	b4	e0	fc	93	b0	71	a0
2	7b	fe	a5	2a	47	cf	df	78	43	a9	ea	f2	8b	aa	49	26
3	08	cb	23	d1	0a	4b	12	2d	64	22	40	e1	9f	17	6a	37
4	03	86	13	1a	27	4f	33	60	91	5b	9b	ad	32	79	dc	48
5	5c	b1	00	f9	10	eb	c3	6d	66	bc	b7	4e	51	54	70	7d
6	c2	fb	74	bf	07	1d	3a	61	45	be	04	ef	30	0f	77	98
7	34	95	02	1f	a2	3d	15	f3	e5	b6	da	09	20	ff	ee	d8
8	97	0c	a1	ba	3f	57	14	36	c1	76	bd	ab	0d	ce	a8	75
9	21	06	7c	6e	11	58	41	18	46	de	b8	05	7e	2f	0b	e7
a	31	2c	69	81	52	24	82	1b	0e	d6	53	62	29	96	a3	e6
b	fa	38	9d	f4	8d	7a	2e	63	6c	35	3e	8f	f0	e9	d5	80
c	3b	87	25	8e	1c	a6	b2	3c	59	db	55	5e	72	6f	9a	89
d	16	6b	b3	cc	50	44	f6	4a	c4	1e	5d	b9	83	68	4d	c7
e	e2	f8	a4	90	2b	9e	9c	94	ae	ca	e8	ec	cd	c5	28	fd
f	8c	d0	8a	4c	7f	e3	c0	92	88	ac	c6	d2	85	c8	d7	19

APPENDIX F TABLE OF OPERATIONS INVSUBBYTES

x/y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	72	40	6a	9b	91	64	30	7b	34	9e	81	8c	a8	6d
1	54	94	36	42	86	76	d0	3d	97	ff	43	a7	c4	65	d9	73
2	7c	90	39	32	a5	c2	2f	44	ee	ac	23	e4	a1	37	b6	9d
3	6c	a0	4c	46	70	b9	87	3f	b1	16	66	c0	c7	75	ba	84
4	3a	96	08	28	d5	68	98	24	4f	2e	d7	35	f3	de	5b	45
5	d4	5c	a4	aa	5d	ca	15	85	95	c8	00	49	50	da	cb	06
6	47	67	ab	b7	38	0b	58	05	dd	a2	3e	d1	b8	57	93	cd
7	5e	1e	cc	01	62	8f	89	6e	27	4d	b5	20	92	5f	9c	f4
8	bf	a3	a6	dc	11	fc	41	c1	f8	cf	f2	2c	f0	b4	c3	bb
9	e3	48	f7	1c	e7	71	ad	80	6f	07	ce	4a	e6	b2	e5	3c
a	1f	82	74	ae	e2	22	c5	0e	8e	29	2d	8b	f9	4b	e8	03
b	1d	51	c6	d2	19	0a	79	5a	9a	db	83	0d	59	8a	69	63
c	f6	88	60	56	d8	ed	fa	df	fd	12	e9	31	d3	ec	8d	25
d	f1	33	fb	0f	10	be	a9	fe	7f	04	7a	c9	4e	02	99	26
e	1a	3b	e0	f5	17	78	af	9f	ea	bd	2a	55	eb	14	7e	6b
f	bc	18	2b	77	b3	13	d6	0c	e1	53	b0	61	1b	ef	21	7d

APPENDIX G ROTATION ANGLES FOR PF_{FORM}



DECENTRALIZED APPLICATION INFRASTRUCTURE

What is a blockchain platform and how it can provide a functionality to launch a project on it

At the heart of any blockchain platform is DLT (Distributed Ledger Technology). Blockchain is nothing more than transactions secured and executed by a scripting language using cryptographic methods. This means that blockchain is a platform with a scripting language that can solve many use cases other than just cryptocurrencies.

Blockchain is a promising technology for the future. However, in its initial form, it is inconvenient to use, and as a result, it is difficult for a variety of practical applications.

The evolution of blockchain technology (under the pressure of flexibility and applicability needs) led to smart contract's appearance. As a result, developers can create private cryptocurrencies and contract-based applications using a programming language, which allows businesses to use this language to set their own rules and policies in such applications. Thus, the blockchain platforms appeared, combining the blockchain proper and flexible tools, such as smart contracts, for implementing various projects based on it.

Blockchain technology (or DLT) that is used in blockchain platforms offer multiple benefits to businesses that make a difference when implementing a solution that requires a high degree of confidence for business transactions (see fig. 1). Using the blockchain platforms offers the possibility to reduce costs and offers the opportunity for businesses to build and maintain an infrastructure that delivers capabilities at lower expenses than traditional centralized models.

Due to the distributed architecture and the use of cryptography, blockchain technology provides a high level of data protection. Furthermore, where its a public ledger or a private blockchain it provides full transparency to its users.

The blockchain platform, whether blockchain or other DLT, should provide an opportunity to launch various projects.

Some blockchain platforms aims to satisfy specific industries such as finance or supply chain.

Another blockchain platforms, on the other hand, are general-purpose platforms, although its design enables a greater utility to applications that make use of thin clients, like the IoT industry.

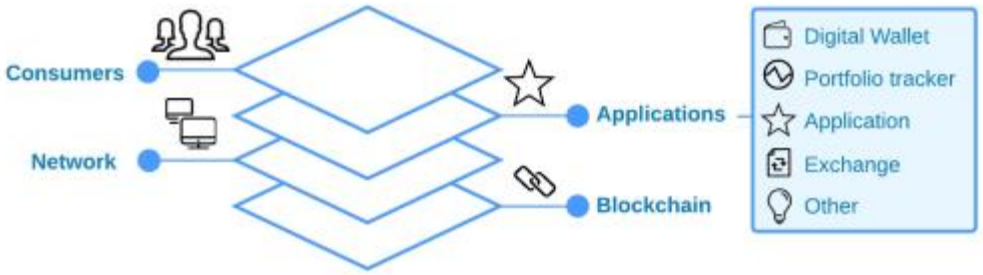


Figure 1. Generalized scheme of blockchain platform

Trying to expand the ability of the blockchain platform, some additional technologies can be included in it. For example, the applied virtual machine has far-reaching implications, since it will allow developers to program smart contracts in traditionally used programming languages. Many compatible programming languages also have support from tested compilers, enabling higher security which is needed for our industry. A combination of such technologies makes possible to create a multi-purpose blockchain platform.
Overall, blockchain technology can be successfully applied in all activities where various types of distributed data confirmation and their network storage are used. And blockchain platforms, in turn, allow you to implement such projects.
In this document, the Apollo blockchain platform will be considered as one of the platforms for launching projects with its distinctive features and advantages. It will be shown how its implemented features make it possible to effectively use the Apollo platform in various fields of application. The document will show that Apollo can support a lot of decentralized applications on a commercial scale, better than known competitors like Ethereum or EOS.
Unlike the competitors, the Apollo platform is oriented on the end-user and its convenience using a Apollo product. Performance increasing features (frameworks, virtual machines, side chains support, etc.) will be automatically implemented into the platform. User tools and interfaces also are implemented to the Apollo platform. Such solutions will allow the Apollo platform to be applied by users with minimal technical knowledge in the DLT field.

Examples of using the Apollo platform to build concrete specialized projects will be also given.

Economic expediency

Investigations of International Data Corporation (IDC) [1] shows that spending growth of buying servers and storage in 2019 is not increased compared to 2018 (see fig. 2). This fact indicates the growing popularity of network and cloud solutions. Of course, some part of these spendings will be on DLT including deploying projects on blockchain platforms.

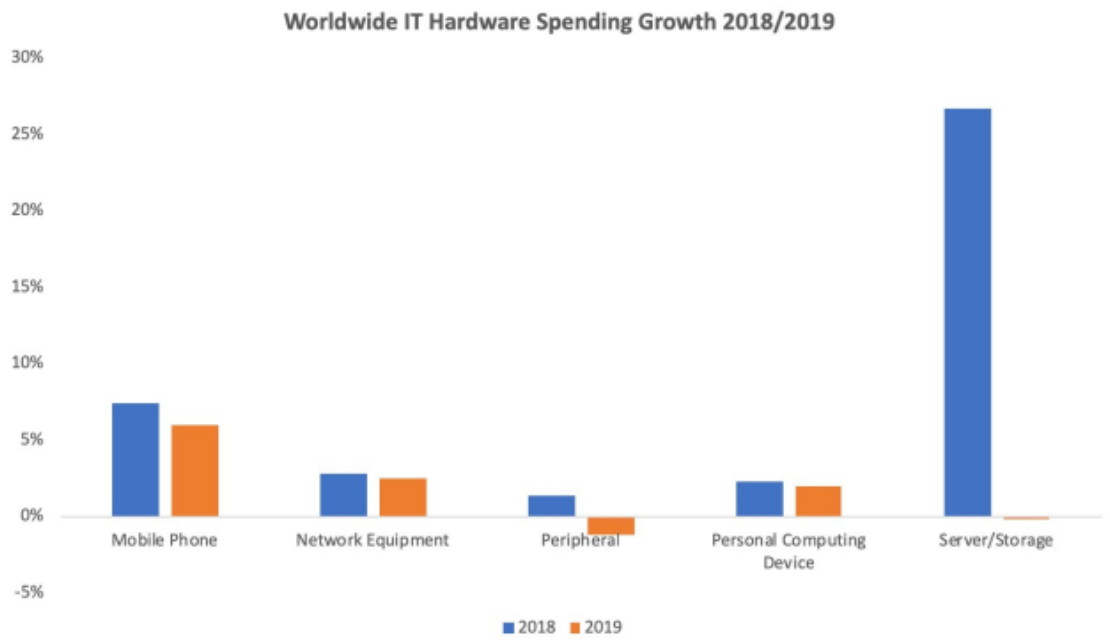


Figure 2. Worldwide IT Hardware Spending Growth 2018/2019 (acc. IDC data [1])

Though blockchain hasn't reached its full potential, savvy executives who took part in Deloitte's 2019 global blockchain survey [2], are confident about new and evolving use cases. They still considering the technology as a connecting platform that can enable many business processes.
Since the last Deloitte's survey,[2] respondents report that overall corporate blockchain investment is growing across most sectors as new, practical applications gain traction.

Following 2019 survey data [2], respondents noted that blockchain providing more diverse advantages than in 2018. Similarly, the increasing diversification of potential use cases for blockchain and the wider array and greater parity of identified barriers to blockchain adoption.

> According to International Data Corporation (IDC) [2], global spending on blockchain solutions reached $1.5 billion in 2018.
>
> According to experts [3], in 2019 the volume of the global blockchain market will grow by 88.7%, amounting to $2.9 billion, and in 2022 it will be measured $ 12.4 billion. The average annual growth rate of the market in question is expected to reach 76%.

This chapter describes the prospects for using blockchain technologies and how existing blockchain platforms help in the implementation of various projects. Since blockchain is a distributed database- any data can be stored on it. The most popular blockchain applications are:

- Financial services,
- Identity management,
- Registration and Verification,
- Contract Execution,
- Tracking supplies and purity of goods origin,
- Distributed data storage,
- Insurance industry,
- Notary sphere,
- Copyright,
- Internet of Things (IoT),
- Automated Security.

The list of possible applications is long and growing constantly. This is due to the following advantages of blockchain technology:

Decentralization. The blockchain does not have a single control center or storage location, and all network participants whose nodes are located around the world are directly involved in maintaining operability.

Data security. Multiple duplications of data among its participants guarantees the safety and immutability of the information entered into the blockchain.

Fast transactions without intermediaries. Blockchain allows payments without intermediaries. Transactions are confirmed by network participants, which means that there is no longer any need for the participation of banks and other payment systems. So, the transaction cost is also reduced.

Transaction accuracy. The accuracy of payments on the blockchain is guaranteed by smart contracts. They are an algorithm whose conditions cannot be violated. Unlike ordinary contracts, a smart contract will be executed only if the conditions are met.

Data security. Information that is stored on the blockchain is encrypted automatically. So, transactions become irreversible, and the information inside the blocks becomes inaccessible to outsiders.

Here is the top use case on 2019 blockchain applications market share (value (constant annual)) in accordance with IDC Spending Guide [3]:

15.9% - Cross-border payments;
10.0% - Trade finance and Post trade;
9.8% - Lot lineage/provenance;
8.6% - Assets/goods management;
7.3% - Regulatory compliance.

Note. In this document, the term "blockchain" summarizes the entire set of DLT. In particular, Directed Acyclic Graph (DAG) as an unofficial 3rd generation of blockchain will also be included in the term “blockchain”.

Apollo platform vs Competitors

Choosing a blockchain platform for project deployment

When creating a new blockchain project, the question inevitably arises:

- Use the existing blockchain platform with a ready-made infrastructure or
- Develop the platform by yourself from scratch.

Ultimately, it all depends on the requirements of the future project and the capabilities of the blockchain platform.

The creation of various cryptocurrency platforms has greatly simplified the creation of new blockchain projects and decentralized applications (DApps). Thus, it became possible to use the blockchain system as a financial instrument daily.

We will compare the existing multi-purpose blockchain platforms below in this document and show what advantages Apollo has.

During the search the best blockchain platform for your needs, you should focus on the following requirements:

1. Considered blockchain platform must correspond to tasks and technical requirements (functionality, possibilities, max TPS rate, etc.) of your project.
2. The chosen blockchain platform must use the newest, stable and most productive technologies.

3. Such a blockchain platform should be able to scale and flexible modernization for the implemented project growing. It should make possible a quick extension of the new project functionality.
4. 4. The presence of additional tools for project implementation, debugging and management at the chosen blockchain platform will be a plus.

We affirm the Apollo platform will correspond to the declared requirements. So, let's analyze the Apollo platform by these four positions with the focus on its features and compare them with known competitors.

A smart contract is the primary feature of the blockchain platform that defines its wide functionality and application area. A smart contract implies the possibility of executing arbitrary program code if the contract terms are met. In this case, the smart contract is recorded in the blockchain (or DLT), and its condition and implementation cannot be further changed without public publishing and auditing it's code (upgradeable smart contract).
Implementation of the smart contract makes possible to apply the Apollo platform for financial reasons like exchanges, financial intermediaries, etc. But smart contracts application is not restricted by the field of finance. It can be used in any field where automatic and rigorous fulfillment of conditions is necessary, for example, logistics and distribution of goods, manufacturing, management, etc.
Today, Ethereum is a main smart-contracts platform for creating new projects. The investigation [4] of Ethereum`s smart contracts findings indicate high levels of contract activity (largely independent of price), but low levels of contract diversity: most contracts are direct - or near - copies of other contracts. This fact represents a potential risk of buggy or a vulnerable code was copied.
Also, it was shown in a survey of attacks on Ethereum smart contracts [5], that about 30% of all smart contracts on the Ethereum network are involved in the execution of other smart contracts – users do not use them directly. This is a potential threat to network stability.
The Apollo smart contract is developing and implementing with an understanding of these risks and with built-in protection against such threats.
Another important obstacle to the widespread adoption of smart contracts is the technical side of creating smart contracts and the need to attract technical experts.

3. Such a blockchain platform should be able to scale and flexible modernization for the implemented project growing. It should make possible a quick extension of the new project functionality.
4. 4. The presence of additional tools for project implementation, debugging and management at the chosen blockchain platform will be a plus.

We affirm the Apollo platform will correspond to the declared requirements. So, let's analyze the Apollo platform by these four positions with the focus on its features and compare them with known competitors.

A smart contract is the primary feature of the blockchain platform that defines its wide functionality and application area. A smart contract implies the possibility of executing arbitrary program code if the contract terms are met. In this case, the smart contract is recorded in the blockchain (or DLT), and its condition and implementation cannot be further changed without public publishing and auditing it's code (upgradeable smart contract).
Implementation of the smart contract makes possible to apply the Apollo platform for financial reasons like exchanges, financial intermediaries, etc. But smart contracts application is not restricted by the field of finance. It can be used in any field where automatic and rigorous fulfillment of conditions is necessary, for example, logistics and distribution of goods, manufacturing, management, etc.
Today, Ethereum is a main smart-contracts platform for creating new projects. The investigation [4] of Ethereum`s smart contracts findings indicate high levels of contract activity (largely independent of price), but low levels of contract diversity: most contracts are direct - or near - copies of other contracts. This fact represents a potential risk of buggy or a vulnerable code was copied.
Also, it was shown in a survey of attacks on Ethereum smart contracts [5], that about 30% of all smart contracts on the Ethereum network are involved in the execution of other smart contracts – users do not use them directly. This is a potential threat to network stability.
The Apollo smart contract is developing and implementing with an understanding of these risks and with built-in protection against such threats.
Another important obstacle to the widespread adoption of smart contracts is the technical side of creating smart contracts and the need to attract technical experts.
The Apollo platform involves the usage of a wide range of popular JVM-based programming languages to create smart contracts. In the future, it is planned to create user`s tools to simplify the creation of new smart contracts without, or with less involvement of technical specialists.

It should be noted that smart contracts implementation helps the Apollo platform to be used for different applications. But the Apollo team does not stop on this achievement. Another step is a migration to the true modular architecture of the Apollo platform. In general, such architecture contributes to the quick and convenient configuration of the Apollo platform to the needs of the concrete project (that is clause 3 of our requirements list). This also greatly simplifies updating and expanding the platform (clause 2). Now let's focus on these two requirements in more detail.
The main benefit of Apollo`s true modular architecture is the possibility to create any desired software configuration of the platform using the flexible kit of interacted software blocks.
So, the software update inside the Apollo platform comes down to a simple exchange of any software block to the newest one. The same algorithm is applied to change some internal algorithm. For example, the applied Proof-of-Stake consensus algorithm can be changed to another by the exchange of corresponding software blocks. Such manipulation will not require the intervention of the software kernel of the Apollo platform.
On a large scale, all blockchain technology can be replaced with another DLT like a DAG. In this case, the number of transactions per second will be significantly increased. This will allow the Apollo platform to be used as a high-performance payment system or for IoT with a large number of micropayments.

Note, that **true modular architecture** is the unique feature of the Apollo platform. None of the existing competitors can boast such a flexible architecture. The applicability of this feature to the implementation of various projects on the Apollo platform is difficult to underestimate.

Now a few words about the scalability of the Apollo platform. Described modular architecture allows the growing productivity and functionality of the Apollo platform. That means that Apollo can support thousands of decentralized applications on a commercial scale, better than Ethereum or EOS. Parallel execution, asynchronous communication, and other functions also increase the scalability of the Apollo platform.

It should be marked here that Apollo (unlike well-known Hyperledger Fabric) has its cryptocurrency. That fact simplifies the usage of the Apollo platform for financial purposes (decentralized exchange, payment system, etc.). Apollo cryptocurrency can also be used for internal settlements in a non-financial project that is based on the Apollo platform.

Here are the questions related to additional tools (clause 4 of our requirements list).
The Apollo development team understands the importance of Apollo platform usability for end users. Therefore, a toolkit will be further developed. It will allow users to do the following:

1. 1Quickly and effectively deploy your projects based on the Apollo platform (improved analog of the HyperLedger Composer).
2. Monitor the project, manage the project Composer (improved analog of the PlayGround).
3. Quickly and easily create smart contracts (mentioned above).

Unlike the Hyper Ledger Fabric [6-7] the Apollo tools will be oriented to the end-user and its convenience using a software product. Performance increasing features (frameworks, virtual machines, side chains support, etc.) will be automatically implemented into the platform.
Such solutions will allow the Apollo platform to be applied by users with minimal technical knowledge in the DLT field.
The described set of user tools will be the final stage of the Apollo platform adaptation for the deployment of various commercial and non-commercial projects on its basis.

Table 1. Comparison the Apollo with similar multi-purposes blockchain platforms

	Apollo	Ethereum	EOS	Tron	Hyperledger Fabric
Purpose	**Universal blockchain platform**	Universal blockchain platform	Platform to suit a wide range of business needs	Decentralized blockchain platform for different content type	Corporate applications
***Market capitalization**	**$27 M**	$24 293 M	$3 929 M	$1 464 M	N/A
***Market place**	**157**	2	8	12	N/A
Currency	**APL**	ETH	EOS	TRX	N/A
Access to network	**Open, public or private**	Open, public or private	Open, public or private	Public or private	Open, public or private
Consensus algorithm	**• Proof-of-Stake (DPoS) consensus Migration to Directed • Directed Acyclic Graph (DAG) technology with different types of consensus algorithms is announced**	• Proof-of-Work (PoW) • Registry level.	• Delegated Proof-of-Stake (DPoS) consensus algorithm • Asynchronous Byzantine fault tolerance (aBFT) • Proof of Completeness (PoC)	Delegated Proof-of-Stake (DPoS) consensus algorithm	• Several options to choose a client • Transactional Level

Smart contract (SC)	+ (announced)	+	+	+	+ Smart contracts as governing transactions and chaincode governs how smart contracts are packaged for deployment.
User interface for SC	**Any language that runs in JVM**	Solidity	C++ with conversion to web assembly	TRON Studio is an IDE for developing, deploying, and debugging smart contracts.	Golang, JavaScript, Java
Virtual Machine	+ (**Java Virtual Machine, JVM is announced**)	Ethereum Virtual Machine (EVM) C++, Python (announced)	- (announced)	TRON Virtual Machine (TVM)	-
* - acc. to CoinMarketCap (https://coinmarketcap.com/), August 2019					

The Apollo platform was compared with well-known blockchain platforms, such as Ethereum, EOS, Hyperledger Fabric and Tron in table 1 above. Each of the considered competitors has certain disadvantages. That fact does not allow them to be fully called the "universal platform for a wide range of applications".
The Apollo platform has incorporated the best solutions and technologies, which will allow it to be in demand in various fields of activity.

Apollo blockchain platform: possible applications

Finance

According to [3], cross border payments & settlements, and trade finance & post trade/transaction settlements are the two blockchain use cases that will receive the most investment ($453 million and $285 million, respectively) in 2019. The banking industry will be the largest investor in both use cases.
That means that the Apollo platform has a huge potential to be used as a cross-border payment system. This is also facilitated by the high technical performance of the platform. The Apollo team has achieved significant success, reducing the transaction time to two seconds in the network. Such a result was provided by blockchain optimization and new features implementation like adoptive forging.

It should be noted that the transition to new technologies (in particular, the introduction of DAG algorithm) will increase the transaction speed (transactions per second or TPS) of the system by at least an order of magnitude and decrease of transaction fee significantly.

Assets such as financial securities must be able to be dematerialized on a blockchain network so that all stakeholders of an asset type will have direct access to that asset, allowing them to initiate trades and acquire information on an asset without going through layers of intermediaries.
Trades should be settled in near real-time and all stakeholders must be able to access asset information in near real-time. A stakeholder should be able to add business rules on any given asset type, as one example of using automation logic to further reducing operating costs. So, the Apollo platform meets these requirements and can be used as a hi-tech asset depository.

Internet of Things (IoT)

According to Cisco White Paper [8], 50 billion devices are due to come online by 2020. With so many connected devices all sending, receiving and processing instructions to turn on, dial down and move up, the sheer amount of data due to come on-stream could come with unprecedented costs. Other issues including tracking and managing billions of connected devices, storing the metadata that these devices produce, and do it all reliably and securely.

Blockchain solution based on Apollo platform could perhaps be a good solution needed by the IoT industry. Blockchain technology can be used in tracking billions of connected devices, enable the processing of transactions and coordination between devices, allow for significant savings for IoT industry manufacturers. This decentralized approach would eliminate single points of failure, creating a more resilient ecosystem for devices to run on. The cryptographic algorithms used by blockchains would also help to make consumer data more private. The benefits of decentralizing IoT are numerous and notably superior to current centralized systems.
It should be noted that the expected migration of the Apollo platform to the DAG algorithm will allow making fast extremely small transactions (micropayments) that are necessary for IoT and similar applications

Logistics
Apollo has great prospects for usage in logistics and cross-border transportation. In the “traditional” logistic process, each of the parties involved has their own internal IT systems which function like a system of records. Lots of used logistic documents also tend to be manual, so it is not easy to track them online. Every party is waiting for documents to show up for them to take action. As documents move from one party to another it takes time (see fig. 3).

Using the Apollo platform, it is possible to revolutionize this whole process and provide visibility to all the parties involved. They all rely on a trusted distributed ledger which is immutable. All the documents, certificates, customs clearances, approvals needed, etc. are digitized and stored on the blockchain. Everyone has access to the same information within a few minutes of that document being approved and submitted to the ledger.

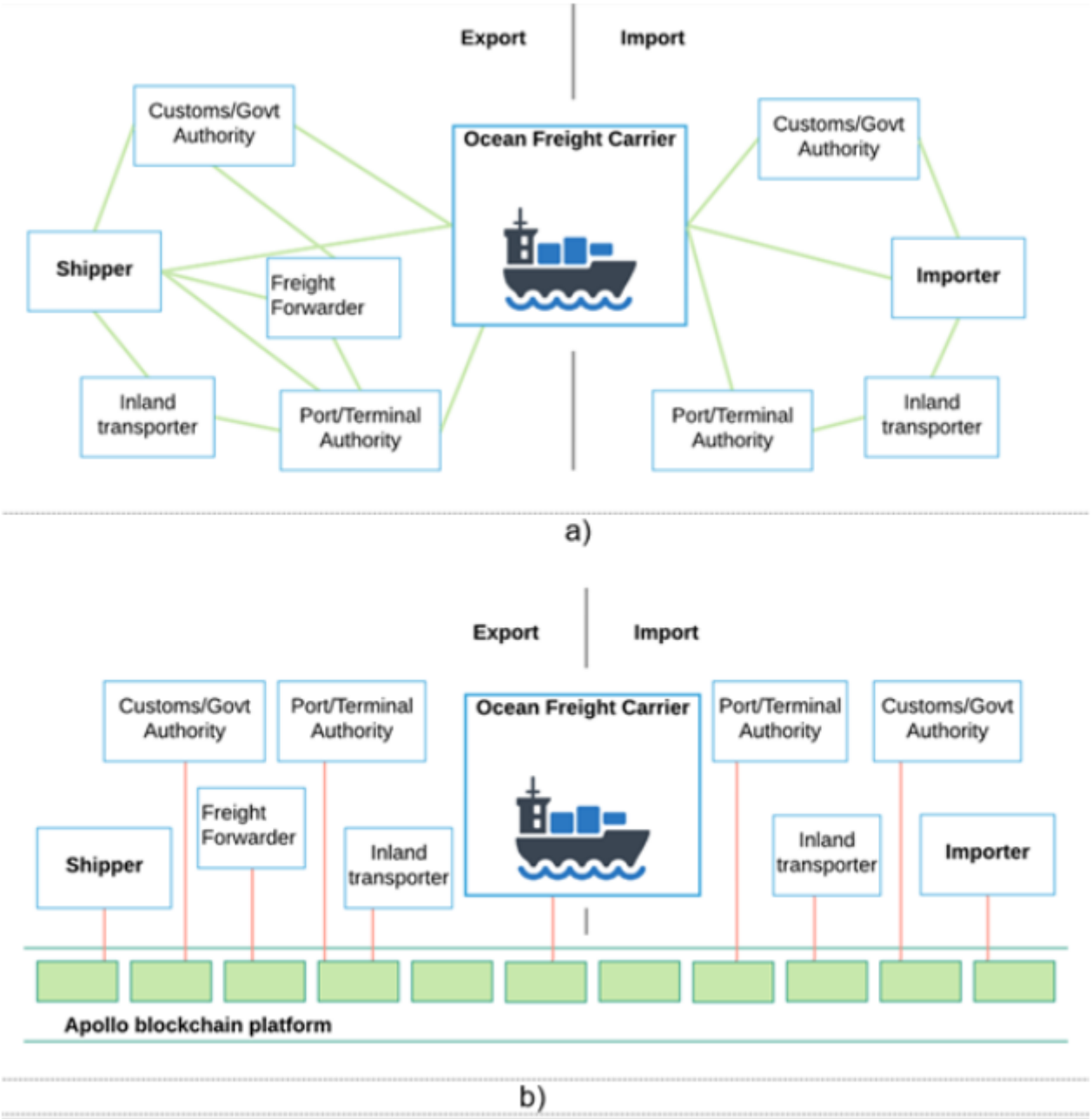


Figure 3. Scheme of "traditional" process (a) and a process on blockchain (b)

Permissioned ledgers of Apollo platform are faster than the open networks like Ethereum or Bitcoin as they do not rely on Proof-of-Work for consensus. As they work on another consensus mechanism like Proof-of-Stake (Zero-Knowledge Proof in the future) and techniques like sharding to improve the output, they could end up being potential platform options for such applications.

Other applications

Decentralized identity is how individuals control when, where and with whom they share their credentials. The Apollo platform can propose a decentralized approach to identity management that is enabled by blockchain.

The Apollo platform makes it possible to store the digital content and provide its sharing from the creators and consumers without a fee for the intermediary (for the Apollo distributed storage) like were declared in Tron White Paper [9].

Applied Apollo blockchain technology can cut out the middlemen for this process. All of the stored data will be cryptographically secure via the blockchain, and the creators retain all of their profits instead of other companies taking a cut. The modular architecture of the Apollo platform will allow it to be adapted for various types of stored content and conditions for its use.

Apollo Smart Contracts

The smart contract is a feature that makes it possible to use the blockchain platform for a lot of applications. For this reason, Apollo will include smart contracts in the near future.

Apollo Smart Contract is the implementation of the mechanism of smart contracts for the Apollo blockchain. It is the ability to create complex data processing scenarios, the possibility of conducting the Initial Exchange Offering (IEO). Smart contracts enable us to perform reliable and confidential transactions without the involvement of external intermediaries. Also, such transactions are traceable, transparent and irreversible. Smart contracts not only contain information about the obligations of the parties and sanctions for their violation, but they automatically ensure the fulfillment of all the terms of the contract.

Apollo's smart contract will be based on the best modern solution is a Java Virtual machine. It allows using different popular programming languages like Java, Kotlin, Scala, Javascript to create smart contracts.

As a model of smart contract implementation, the R3 Corda Master [10] is taken. It should be noted that the R3 Corda Master is verified by wide industrial acceptance. It is the most productive and the best prototype for blockchain solution on Java.

The smart contract code (see fig. 4) can be written in any JVM language, and has access to the full capabilities of the language, including the following rules:

The smart contract code (see fig. 4) can be written in any JVM language, and has access to the full capabilities of the language, including the following rules:

- ·Checking the number of inputs, outputs, commands, time-window, and/or attachments;
- Checking the contents of any of these components;
- Looping constructs, variable assignment, function calls, helper methods, etc.;
- Grouping similar states to validate them as a group (e.g. imposing a rule on the combined value of all the cash states)
- A transaction that is not contractually valid is not a valid proposal to update the ledger, and thus can never be committed to the ledger.

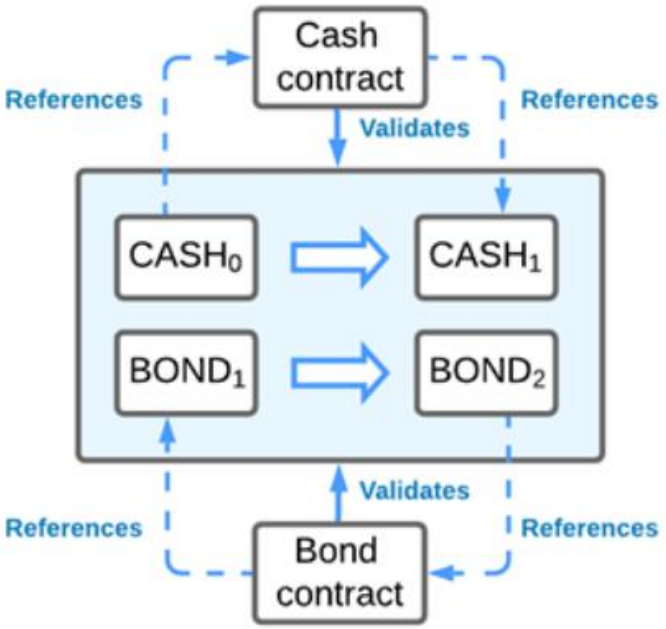


Figure 4. General scheme of smart contract validation

Java will be used for the first iteration of the Apollo smart contracts implementation. Other programming languages will be added at the next stages.
Smart contracts will execute inside the very restricted sandbox.
The gas conception will be not applied to Apollo's smart contracts.
Smart contracts will be a jar file with a special manifest and cryptographic signatures.

As a result, the best virtual machine - JVM and a wide selection of programming languages will be used to create smart contracts.

Steps to implement Smart Contracts in Apollo

Smart contracts will be implemented to the Apollo platform in accordance with the following steps:

1. Smart contract researching. Choosing the optimal technologies and solutions for its implementation.
2. Creating a secure test container (sandbox) for smart contract execution.
3. Separation of the smart contract API in a single package, integration with the sandbox.
4. Deployment of the test environment (sandbox + test Java Virtual Machine (JVM) + Java compiler).
5. Testing the smart contracts creation using the Java language in Test-net.
6. Testing related to smart contracts. Study of platform stability.
7. Formation of a smart contract modular solution in accordance with the true modular architecture of the Apollo platform.
8. Integration with the Apollo platform.
9. Front-end development.
10. JVM deployment and implementation of smart contracts to the Main-net.
11. Development of custom tools for working with smart contracts (interfaces for creating, managing, etc.).

Decentralized architecture scheme

As mentioned above, soon the Apollo platform will migrate to a truly modular architecture (see fig. 5). As a result, the Apollo software will become a flexible kit for quickly and conveniently build any desired software configuration.

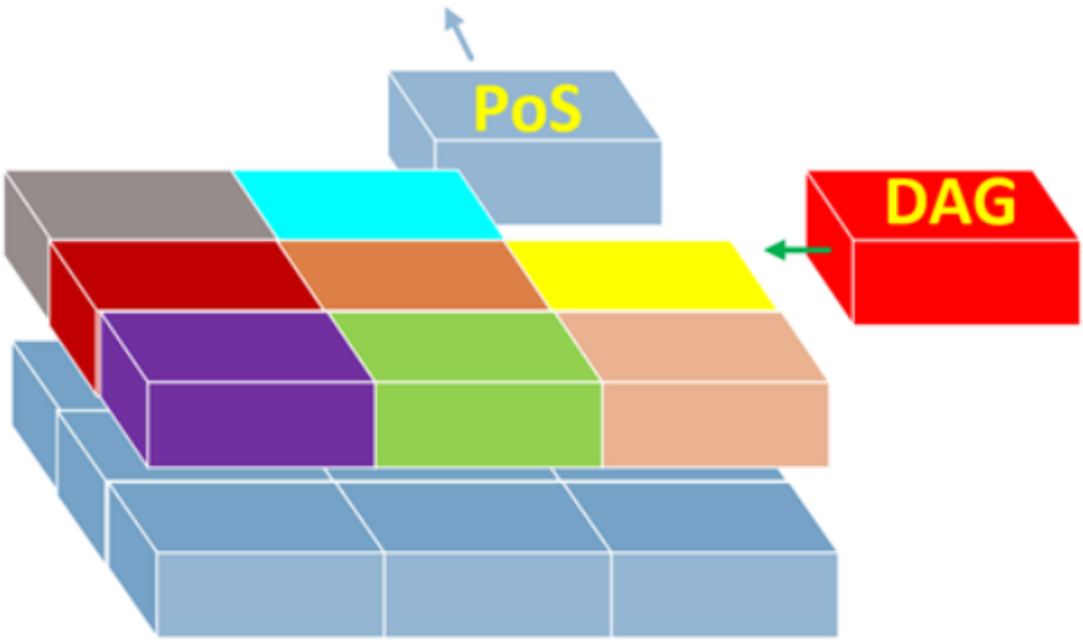


Figure 5. Concept of Apollo modular structure as a flexible blocks kit. DLT blocks exchanging is shown

Conclusion

This section describes the Apollo blockchain platform and compares it with competitors in the market for similar solutions.
It is shown that the announced Apollo platform features, such as smart contracts implementation and migration to a truly modular architecture, are designed to significantly expand the range of Apollo applications. A variety of projects can be deployed at the considered Apollo platform. This will allow the Apollo to be called a universal and multi-purpose blockchain platform.
The combination of unique features of the Apollo platform will make it the best among the competitors.

References

[1] 3 Things To Expect in the IT Market in 2019
https://blogs.idc.com/2019/01/24/3-things-to-expect-in-the-it-market-in-2019/

[2] Deloitte's 2019 Global Blockchain Survey
https://www2.deloitte.com/insights/us/en/topics/understanding-blockchain-potential/globalblockchain-survey.html

[3] Worldwide Blockchain Spending Forecast to Reach $2.9 Billion in 2019, According to New IDC Spending Guide
https://www.idc.com/getdoc.jsp?containerId=prUS44898819

[4] Analyzing Ethereum's Contract Topology. Lucianna Kiffer, Dave Levin, Alan Mislove https://mislove.org/publications/Ethereum-IMC.pdf

[5] A survey of attacks on Ethereum smart contracts. Nicola Atzei, Massimo Bartoletti, and Tiziana Cimoli https://eprint.iacr.org/2016/1007.pdf

[6] IBM Blockchain https://www.ibm.com/blockchain

[7] Hyperledger Fabric https://wiki.hyperledger.org/display/fabric

[8] Cisco White Paper. The Internet of Things How the Next Evolution of the Internet Is Changing Everything
https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[9] Tron. Whitepaper Version: 2.0. Advanced Decentralized Blockchain Platform
https://tron.network/static/doc/white_paper_v_2_0.pdf

[10] Corda Smart Contract https://docs.corda.net/key-concepts-contracts.html

FUTURE PROJECT DEVELOPMENT

Smart Contracts

The smart contract is a feature that makes it possible to use the blockchain platform for a lot of applications. For this reason, Apollo will include smart contracts in the near future.

Apollo Smart Contract is the implementation of the mechanism of smart contracts for the Apollo blockchain. It is the ability to create complex data processing scenarios, the possibility of conducting the Initial Exchange Offering (IEO). Smart contracts enable us to perform reliable and confidential transactions without the involvement of external intermediaries. Also, such transactions are traceable, transparent and irreversible. Smart contracts not only contain information about the obligations of the parties and sanctions for their violation, but they automatically ensure the fulfillment of all the terms of the contract.

Apollo's smart contract will be based on the best modern solution is a Java Virtual machine. It allows using different popular programming languages like Java, Kotlin, Scala, Javascript to create smart contracts.

As a model of smart contract implementation, the R3 Corda Master [10] is taken. It should be noted that the R3 Corda Master is verified by wide industrial acceptance. It is the most productive and the best prototype for blockchain solution on Java.

The smart contract code (see fig. 13) can be written in any JVM language, and has access to the full capabilities of the language, including the following rules:

- Checking the number of inputs, outputs, commands, time-window, and/or attachments;
- Checking the contents of any of these components;
- Looping constructs, variable assignment, function calls, helper methods, etc.;
- Grouping similar states to validate them as a group (e.g. imposing a rule on the combined value of all the cash states)
- A transaction that is not contractually valid is not a valid proposal to update the ledger, and thus can never be committed to the ledger.

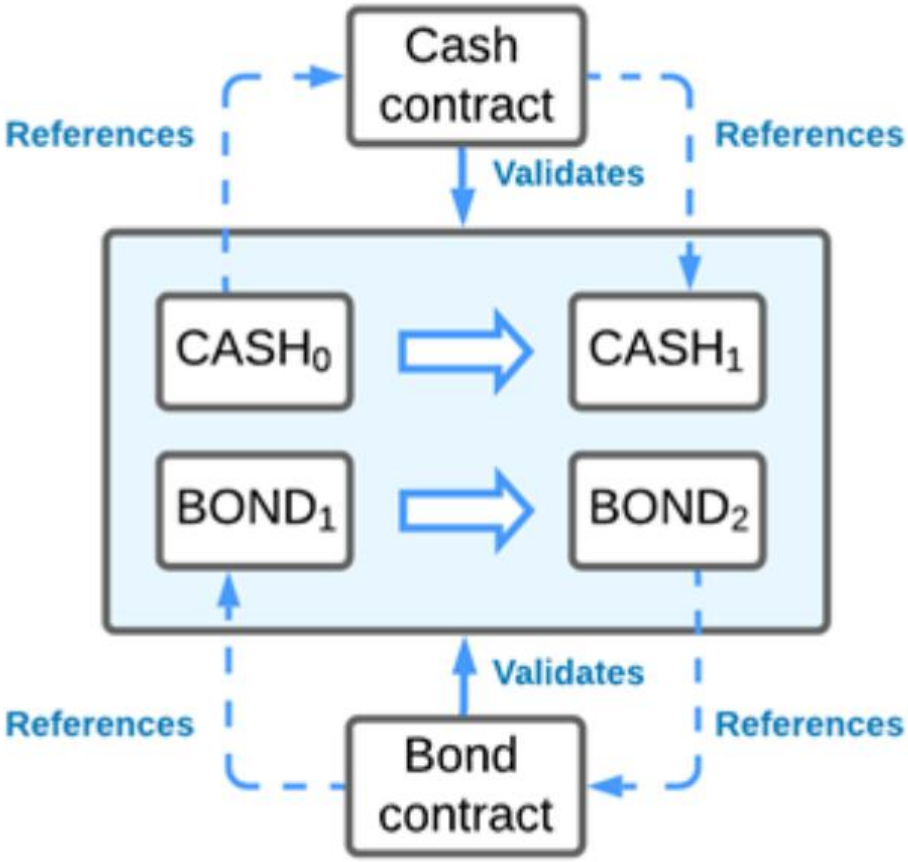


Figure 13. General scheme of smart contract validation

Java will be used for the first iteration of the Apollo smart contracts implementation. Other programming languages will be added at the next stages.
Smart contracts will execute inside the very restricted sandbox.
The gas conception will be not applied to Apollo's smart contracts.
Smart contracts will be a jar file with a special manifest and cryptographic signatures.
As a result, the best virtual machine - JVM and a wide selection of programming languages will be used to create smart contracts.

Smart contracts enable anyone to perform reliable and confidential transactions without the involvement of external intermediaries.
Such transactions are traceable, transparent, and irreversible. Smart contracts not only contain information about the obligations of the parties and sanctions for their violation but they also automatically ensure the fulfilment of all the terms of the contract.

Arknet

The next logical step that comes after smart contracts implementation is the launch of the Arknet, Apollo's DApp platform. DAPPs are a structural unit that provides various use cases to tailor the application to project needs. DApps introduce a new paradigm for applications that work in Fintech segment. The use of Dapps oers many advantages, such as security, transparency, and persistence in operational processes.

DApps allow decentralizing all data, which makes them immutable and protected from outside interference. Given the decentralized nature of these applications and the blockchain data protection mechanisms, DApps have an almost unlimited variety of applications. Both smart contracts and ARKnet will add drastic competitive advantages to Apollo and also expand the application range of both the project and the platform as a whole.

Apollo Decentralized Data Storage (ADDS)

ADDS is a decentralized alternative for cloud services. One of the disputed issues of modern techno-community is the storage of data. Users search for services that solve storage requests and also offer a highly secure environment.

The Apollo Foundation has analyzed the most popular solutions that are available on the market and came to the following conclusions:

- There are no fully decentralized data storage solutions on the market;
- The cost of data storage is an entry barrier for those who have a high volume of data;
- Security is the most critical factor for users when choosing a file storage service.

Based on these factors the Apollo Foundation formed a radically new concept that will not only allow to create a fully decentralized data storage but also make it secure and convenient in use due to the technologies on which the Apollo platform is built and continues to evolve.

The data storage scheme will be built on full decentralization - data is stored in the network, and users can store not only their information but also create the ability to store for others, thereby earning from this. The more opportunities you open for data storage, the higher your income from this can be.

DDS as a system will exclude the possibility to attack the network through spamming with junk files. File segmentation will occur in such a way to achieve maximum availability and data security. The result of the development of DDS as a structural element of the Apollo platform will be the emergence of service with a dynamic pricing system, and the possibility of monetization of user's hardware as a unit of the system.

Ark - reaching unlimited capabilities with DAG implementation

ARK aims to functionally transform the Apollo Platform as well as the cryptocurrency market and highlight unique competitive advantages.

Distributed Ledger Technology (DLT) is developing very fast. A new technology called Directed Acyclic Graph (DAG) has appeared as a front runner in the various advancements available. At the heart of this technology is the solution to the problem of scalability, high commissions, and high transaction confirmation time. Now the market is seeing a trend toward the best solutions and strongest platforms. Many now claim that DAG is a 4th generation of the blockchain and the future of cryptocurrency.

First of all, DAG is a type of distributed ledger technology that differs from blockchain in the structure of records and asynchrony. Many people mistakenly believe that DAG is a type of blockchain or some new consensus. Both blockchain and DAG are different solutions for distributed ledger technology. Blockchain and DAG can be called close relatives, but they solve their tasks in different ways.

DAG functions as a network of interconnected branches that grows outward in several directions. Transactions can be confirmed in order of magnitude faster while remaining decentralized since each node confirms only the previous one.

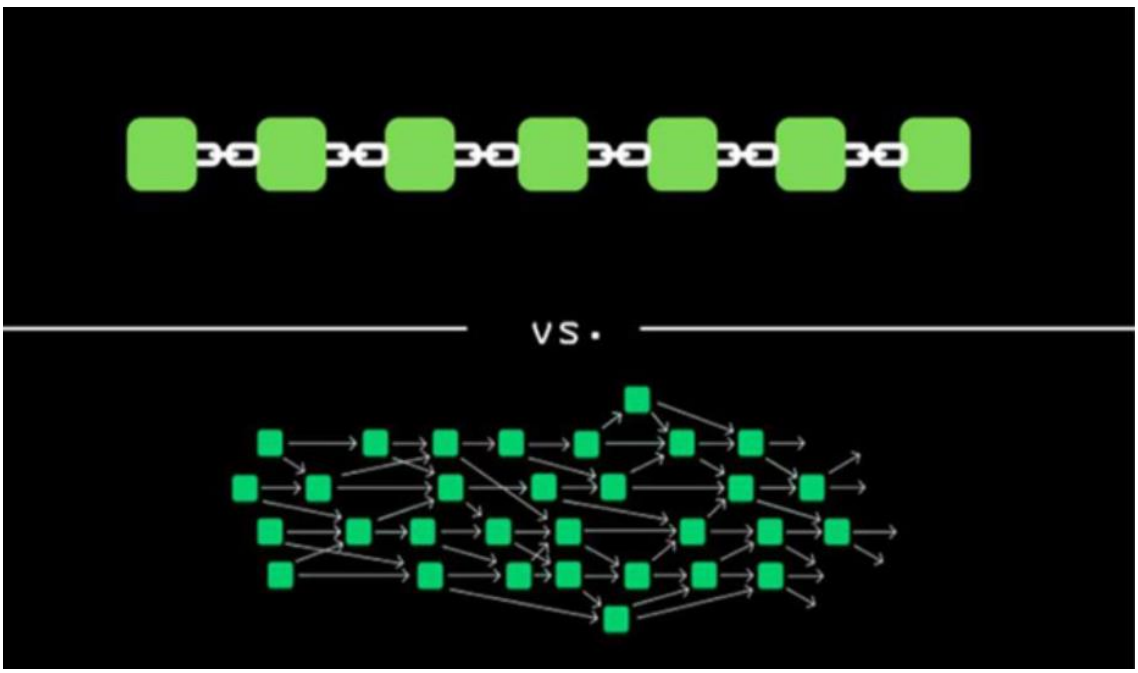


Figure 14. Blockchain vs DAG structures comparison

Each transaction in the DAG refers to the previous (parent) ones, signing their hashes and including them in its composition. Thus, a "tree" of transactions is formed, where each of them is confirmed and unchanged. A notable difference between ARK and traditional blockchain is that the more users that are on a traditional blockchain the slower than blockchain becomes. In contrast, the more users on ARK, the faster the platform will operate. There are no limits to the amount of users that ARK will be able to operate with, whether there are 10 users or 10 billion, ARK will perform better.

Pros of the ARK DAG technology:

- A high number of transactions per second (TPS);
- Low transaction fees;
- Nano transactions – micropayments for IoT and similar applications;
- Scalability. DAG technology appeared as a result of the inability of the blockchain to cope with heavy loads and network congestion. When using DAG, only nodes are controllers and validators of transactions, not blocks.

Apollo with implemented DAG intends to solve the following:

- To solve the scalability issues of existing public distributed ledger technologies;
- To distinguish the Apollo platform from the traditional block ledger-based storage infrastructure by employing an improved version of existing DAG-based protocols.
- To theoretically process up to 1 million transactions or more per second.

Conclusion

Currently, the Apollo platform is on the way to becoming a true modular project architecture. It will be a multi-functional block constructor for various applications. The Apollo Foundation has achieved significant success, reducing the transaction time to 2 seconds in the network. Thus, the actual limit of the used blockchain technology was reached.

However, the Apollo Foundation has extensive plans for further development and improvement of the platform. Apollo platform will implement ARK, a modern DLT based on DAG algorithms to eliminate these shortcomings and further develop the project to replace the existing blockchain. Such an implementation will give the Apollo platform many competitive advantages over nearly all cryptocurrency platforms on the market. In particular, multiple increases in the number of transactions, the total amount of users, and a significant reduction in its processing time are expected. The ability to use smart contracts, the launch of ARKnet DApp platform, the optional choice of consensus algorithm based on project's modularity, and launch of Apollo Cloud - truly decentralized data storage, opens up a virtually unlimited scope for Apollo.

ROADMAP

2017

✓

CREATION OF APOLLO

Legend

In Progress

Complete ✓

2018

1Q

- ✓ Fan Store Available
- ✓ Non Technical Whitepaper Available
- ✓ WEB WALLET RELEASE
- ✓ MONETARY & ASSET SYSTEM INTEGRATED

2Q

- ✓ OLYMPUS PROTOCOL 1.0 RELEASE
- ✓ IP Masking 1.0
- ✓ Private Transactions (API Level)
- ✓ WALLET 1.0 WINDOWS AND OSX
- ✓ Olympus Protocol Guide
- ✓ OLYMPUS PROTOCOL WALLETS RELEASED
- ✓ CONFIRMED ON MULTIPLE EXCHANGES
- ✓ Voting Feature Used for Coin Design
- ✓ Whitepaper Translated to Chinese

3Q

- ✓ OFFICIAL LAUNCH ON EXCHANGES
- ✓ R&D
- ✓ WEB WALLET 2.0 RELEASED
- ✓ Full Source Code Released with OP 1.0
- ✓ Full Technical Whitepaper Release

4Q

- ✓ HERMES 1.0
  - ✓ Chain ID
  - ✓ ApolloUpdater
  - ✓ Block2sec
- ✓ IOS & ANDROID WALLET COMPLETED
- ✓ New Website UI/UX
- ✓ OLYMPUS 2.0
  - ✓ IP Masking 2.0
  - ✓ Transport Live Version
  - ✓ ApolloMixer
  - ✓ Two Factor Authentication

2019

1Q

- ✓ HERMES 2.0
  - ✓ ApolloSharding Generation
- ✓ DECENTRALIZED EXCHANGE INTEGRATED INTO WALLET

2Q

- ✓ HERMES 2.0
  - ✓ Pre-Sharding Refactor
  - ✓ Stabilization
  - ✓ Testing
- ✓ APOLLO AFRICA
  - ✓ Apollo Africa Tour
  - ✓ Start of Zimbabwe Initiative
  - ✓ Start of Swaziland Initiative
  - ✓ ADF Partnership
  - ✓ Apollo Africa HQ
- ✓ WALLET 3.0

3Q

- ✓ FIRST SHARD EXECUTED
  - ✓ Block 2,250,000
- ✓ APOLLO VISION 2020
- ✓ HERMES 3.0
  - ✓ FastApollo
  - ✓ ApolloDEX
- ✓ APOLLO AFRICA
  - ✓ Start of Comoros Initiative
  - ✓ Start of Lesotho Initiative
  - ✓ Start of Botswana Initiative

4Q

- ✓ APOLLO AFRICA
  - ✓ Zimbabwe Sandbox 1.0
  - ✓ Swaziland Second Trip
  - ✓ Start of Senegal Initiative
  - ✓ Start of Malawi Initiative
  - ✓ Start of Benin Initiative
  - ✓ Start of Ivory Initiative
  - ✓ Start of Burkina Faso Initiative
  - ✓ Start of Togo Initiative
  - ✓ Start of Mali Initiative
  - ✓ Start of Niger Initiative
  - ✓ Start of Egypt Faso Initiative

2020

1Q

- ✓ CORE REFACTORING
  - ✓ Blockchain Transport
  - ✓ Modularization
  - ✓ Stabilization
  - ✓ Testing
- ✓ APOLLO REBRANDING
  - ✓ New Website UX/UI
- ✓ APOLLO DEX
  - ✓ ETH smart contract update
  - ✓ Improvements in processes
  - ✓ Tradingview Chart
  - ✓ Code Optimization

2Q

- ✓ KNOXVIP LAUNCH
- ✓ GSX CDE START
- ✓ MAJOR DB UPDATE
- REST API IMPLEMENTATION
- INSTALLERS AND SCRIPT UPDATES

3Q

- EXPLAINER VIDEO 2.0
- ✓ WHITEPAPER 2.0
- ✓ TECHPAPER 2.0
- ✓ ROADMAP 3.0
- KNOX BANK
- STRATUS PHASE 1
  - Local Hub (Facebook)
  - Global Hub (Twitter)
  - Blog Hub (Medium)
  - Pay Hub (Paypal)
- NATIONAL PAYMENT PLATFORM - BETA
- APOLLO CASH - BETA
- BACK END OPTIMIZATION AND IMPROVEMENTS

4Q

- ACADEMIC VALIDATION
- E-GOV PLATFORM
- STRATUS PHASE 2
  - Photo Hub (Instagram)
  - Video Hub (Youtube)
  - Coin Hub (Coinbase)
  - Trade Hub (Robinhood)
- UPDATER 2.0
- NATIONAL PAYMENT PLATFORM - FULL
- ALIAS REFACTORING
  - ✓ Alias Search
  - ✓ Alias Purchase from User
  - Alias Send
- COINMIXING REFACTOR
- MIGRATION TO MARIADB+ROCKS DB
- MULTISIGNATURE ACCOUNTS

2021

1Q

- LIGHT WALLET
- APOLLO ID FEATURE
- APOLLO CASH
- NEW P2P
- STRATUS PHASE 3
  - Clip Hub (Tik Tok)
  - Link Hub (Linked in)
  - Chat Hub (WhatsApp)
  - Social Hub (Telegram)
- NEW FEE STRUCTURE
- KEY MANAGEMENT SYSTEM 1.0
- APOLLO C.A

2Q

- STRATUS PHASE 4
  - Travel Hub (Travelocity)
  - Domain Hub (GoDaddy)
  - ... (Upwork)
  - ... (Envato)
- API VERSION 2.0
- DESKTOP UPDATE

3Q

- SMART CONTRACTS
- APOLLO DECENTRALIZED CLOUD

4Q

- DFS PAYMENT MODULE
- ARK 1.0
  - R&D
  - New Consensus Protocol
  - Algorithm Integration
  - Testing
  - 1 Million TPS Goal

APLFintech.com