remme

Distributed Public Key Infrastructure (PKI) protocol and Access Management DApps

Whitepaper

v.0.3
January, 2018

remme.io

Introduction

Nowadays, the main method of accessing various local and network resources is a password, that has proven itself as a way of identifying and securing users and resources. However, it has one serious drawback: in case of stealing a password, an attacker gets access to all the data of the user who owned this password. In addition, users usually have the same password for multiple services, the situation gets even worse because these passwords can be weak or even be a subject for vocabulary attack. Password databases from a variety of resources periodically fall into open access. In general, even if the resource provides the ability to change the password, this method is somewhat vulnerable, since the user's mailboxes are usually protected by a password, often the same as that used on other resources. Thus, the password can not serve as a sufficient instrument for protecting user data and guarantee the security of the user session.

An approach to this problem is password managers. This software, which provides a secure storage for passwords, and, in the case of integration as a browser extension, is able to withstand numerous ways of stealing passwords. Also, password managers are often able to generate secure passwords that are unique to each resource, which takes security to a new level. The obvious drawback of such protection is that the storage is protected with a master password (Lastpass, 1Password), and in case of theft or brute force the master password, it all boils down to the previous thesis.

Two-factor authentication provides a partial solution of password problems. For two-factor authentication, in addition to the password, it is required to provide the resource with some more data that should be available only to a particular user. Examples of the second factor:

1) One-time passwords that are generated every n seconds are the most common option. Usually implemented with TOTP protocol.
2) One-time passwords sent in the message. Usually, SMS or instant messengers is used.
3) Hardware tokens.

Another way of authorizing a user into the system is SSL/TLS certificates, which are widely adopted in enterprise solutions, such as banking, tax services, etc.
A system of Public Key Infrastructure supports the creation, distribution and identification of public encryption keys, enabling users and systems to both

securely exchange data over untrusted environment such as the Internet as well as verify the identity of the other party of conversation. PKI provides possibilities for digital signature (confirmation of authentication, non-repudiation and message integrity), data encryption (confidentiality during data storing, transfer and processing) and authorization in one complex system.
The core of PKI - the public key encryption systems (a private key for encryption, a public key for decryption) based on strong mathematical approaches. But simple presence of public/private key is not enough for trust. There should be a complex and comprehensive system with all functions listed above.
Typically PKI consists of a lot of controls beginning from policies and standards through administration and management to software and hardware. PKI can be realized in different architectures: simple, network, hierarchy etc. But we should understand that the heart of PKI is digital certificates. A digital certificate is a document designed to affirm the identity (user, system etc.) of the certificate subject and bind that identity to the public key contained in the certificate.

The typical scheme of PKI includes next elements:

- Certification Authority (CA) - a trusted party provides services for issue digital certificates.
- Registration Authority (RA) - a trusted party responsible for accepting requests for digital certificates and authenticating the entity making the request. Sometimes RA also called subordinate CA.
- Validation Authority (VA) - a trusted party provides a service used to verify the validity of a digital certificate. It's clear that different VA should has database of valid certificates, revoked certificates and communication with different CA.

As we can see, functioning of PKI based on trusted authorities with different functions.

From our point of view we should focus on several core issues. First of all, PKI now is government regulated or business driven ecosystem depends on a sector of application. Government CA and PKI at all usually are not acceptable for wide public or SMB use according to different limitations and application lockin. For example, specific CA works with specific tax reporting software. Services of business CA very often expensive and there is a collusion between software vendors and CA for including specific CA into a list of trusted for this software. For example, web browsers don't accept all certificates issued by different CA. Sometimes ago it was a brilliant vision named "web of trust" were most of noted problems could be resolved with teamwork of count of CA/VA/RA. Unfortunately, it left just as vision according to the disagreement s CA to work in a single network of trust.

Our team works on solving those problems by implementing decentralized public key infrastructure based on blockchain technology. The chosen approach will give our end customers the way of managing their PKI with a high level of security and all advantages of decentralized and distributed system, including fault tolerance.

Advantages of REMME:

1. There is no centralized database of certificates and keys that could be compromised.
2. There are no technology lockin and API limitations. Easy integration with existing systems.
3. There are no additional fees for different certificates/credentials in different CA.
4. There are no possibilities for collusion between software/hardware vendors and limited count of CAs.
5. Fast and protected public key distribution process.
6. Fast and protected certificate revocation process.
7. Single point of trust for different systems: easy single-sign-on implementation, decentralized worldwide available authorization.
8. There are no legal limitations and government cooperation issues.

Additional advantages:

1. Acceptable for different types multi-factor authentication.
2. Full anonymity.
3. It allows to track all issued certificates, provides complete and transparent control.

REMME is bringing blockchain to PKI infrastructure providing immutability of data stored there.

For simple user it could look complicated, but all is simple: you don't need to remember count of login and passwords, you don't need to pay five or ten authorities for certificates used in tax, legal, bank, technical or other types of software, you don't need to control the live time of each password/certificate/key.

1. REMME product evolution

Our team tested the possibilities of creating a decentralized PKI on Emercoin blockchain (EMCSSL). This is a production-ready system, but it lacks speed: our focus-group said that it was taking from 10 to 30 minutes for a certificate to become active. Another conception based on Bitcoin blockchain UTXOs (see Annex 1) was also tested, but we decided to move forward for two main reasons:

1) The costs of using existing public blockchains are unstable because they take fees for all operations in their native tokens, which have high volatility;
2) Many of them are quite slow by their nature (Bitcoin, Emercoin) or prone to overloads (Ethereum).

So the solution is to build our own decentralized system with the financial system, that suits the product needs.

For such system there are three main requirements:

1) The pricing policy should be stuck to an existing currency, for example, US dollar.
2) The blockchain should have the high throughput to maintain quick certificate management.
3) The possibility to create a private blockchain to be integrated into different organizations.

To maintain those requirements we designed a system based on a custom blockchain based on **Hyperledger Sawtooth** (see the detailed information of blockchain selection in Annex 3) and a custom consensus algorithm heavily inspired by Ethereum Clique and Dash masternodes network.
Sawtooth is a framework for custom blockchains designed to be modular: every part of the framework including the consensus algorithm can be replaced with custom code. Consensus algorithm in a nutshell is Clique – simple PoA protocol which runs on top of the pre-defined list of nodes – with an addendum which makes it somewhat similar to Dash masternodes network: every user can enter the list of signing nodes by locking a certain amount of tokens and will be excluded from the list if their node works against the network rules and does not hold uptime.

1.1. Certificates support

As in the first version of the system X.509 certificates are used. The following use cases are supported:

1) Self-signed certificates. In this case certificate data, such as public key, signature, expiration date and revocation status are stored on the blockchain.
2) Certificated signed by an organization. In this case an organization (which is our client) may use its own certificate to sign and manage certificates of its clients and employees.

To support REMME-based certificates it is planned to integrate a server-side software which will continuously check for the status of the certificate on blockchain. There can be different ways, such as implementing plugins for content management systems or incorporating self-signed certificates into the system trusted certificates list. In the framework of this system the following fields of a certificate will be exploited:

Field	Usage
Organization name	REMME
UID	The address of a user on REMME blockchain.

1.2. 2FA

The choice of technology for the second factor depends on the characteristics of the system, which is protected by REMME technology.
For example, if a system requires a physical presence of an authorized person, then the best option for the second factor would be to use biometric data, such as a fingerprint scan or an eye retina scan.
If the system uses some local sensor, then the second factor would be the physical connection to the local network check. If the system is remote, it is optimal to use a secondary device (a phone or another PC). The probability of simultaneous malware infection of two devices is lower than of one. It would also help to protect the account with a compromised certificate.
Additionally, an instant messenger can be installed on the second device to receive secret keys via messages from a protected system. In this case, the reliability of the second factor will be equal to the reliability of the messenger

account. For example, it is possible to use Telegram (or other messengers), email, or email + PGP key.
Special consideration should be given to the standard TOTP (time-based one-time password) method, which generates one-time codes within certain time intervals (e.g., every thirty seconds). Such a scheme is implemented, for example, in the Google Authenticator application. You can also use an entirely-hardware solution for generating access tokens, for instance with the help of YubiKey, Yubico or Trezor.

1.3. Token economy and Pricing model

As one of the requirements is to stick prices to a fiat currency, the platform needs some way to update the prices for its services with respect to the exchange rate of the token. To address this requirement REMME will maintain a server which will periodically (10 minutes) update prices and will be available for masternodes. Every masternode can query this server to get the price of a token for a moment of time.
[fig 3]
REM token is needed for all internal operations inside of ecosystem and includes:

1) Initialization of a certificate creation process (an amount of tokens gets blocked for revocation transactions) ($1);
2) Establishing a node (will be available after Q3, 2018, see below) ($10 000);
3) Fee for transferring REMME tokens between users on REMME blockchain (0.1%);

1.3.1 Fees distribution

Depending on the number of tokens collected during the token sale REMME will hold a portion of transaction fees to finance the project development:

1) **Under 5m collected during the public token sale**: 70% of those fees are stored on the account of a master node which created the current block, 30% are transferred directly to REMME and is used to maintain services such as Bitcoin anchoring, cross-blockchain transfers, etc.
2) **From 5m to 10m collected during the public token sale**: 80% of those fees are stored on the account of a master node which created the current block, 20% are transferred directly to REMME and is used to maintain services such as Bitcoin anchoring, cross-blockchain transfers, etc.
3) **Above 10m collected during the public token sale**: 90% of those fees are stored on the account of a master node which created the current block,

10% are transferred directly to REMME and is used to maintain services such as Bitcoin anchoring, cross-blockchain transfers, etc.

1.4 Inter-blockchain token migration

Inter-blockchain token migration is needed to prepare a migration mechanism, which will enable users to use REM token, initially released on Ethereum platform, on REMME blockchain. As for current state of the project the optimal choice is centralized system similar to Waves cryptocurrency gates. The idea is that one of the nodes is tracking transactions in both Ethereum and REMME. When a user sends tokens to this node on Ethereum (with the user's REMME address enclosed in metadata, smart-contract may be needed), the same amount of token is sent to the user's account on REMME, and the system works in the same way for transferring REM to Ethereum blockchain.

1.5 Consensus algorithm

1.5.1. Requirements

As REMME is built on a custom blockchain it is necessary to design a consensus algorithm suitable for the REMME network.
In general, project-specific requirements are:

1) Provide stable response time of the system;
2) Be independent of user's hardware;
3) Incentivize validating users to stay online (2 and 3 together lead to p.1).

1.5.2. Algorithm

The solution for this task is the algorithm called "proof-of-service", which was firstly introduced in Dash masternodes network. It:

1) Do not depend on the hardware capabilities;
2) Can incentivize users to stay online by ranking by their uptime;
3) Suitable for public networks.

This algorithm can be also described as a form of proof-of-stake algorithm with the additional factor of uptime (for the detailed comparison of consensus algorithms see Annex 2).
The core of this algorithm is the network of masternodes. Our concept will be slightly different from one introduced in Dash, masternodes (validators) are responsible for producing new blocks and validating transactions.

The main entity in this algorithm is the global list: it contains the full list of nodes which are eligible to sign new blocks. When a new node is introduced into this list it is placed into the end of the list.
Nodes are selected from the list stored in the blockchain in the order they are introduced to the list.
The algorithm specifies the desired and the maximum time in which blocks should be produced. In the other case, the node is moved out of the global list and have to enter this list again. Blocks produced faster than in the desired time should be rejected.
Producing a block is rewarded by fees from transactions included to the block (see Pricing policy for the details). Note, that the system can produce empty blocks, which will not be rewarded as there are no fees. This is required by the consensus algorithm, as it is using timings between blocks to check if nodes are operating correctly.

1.5.3. Cases

To make the working process of the algorithm clear, some cases of what is happening are provided:

1) **Submitting a request to become masternode**
   To become a masternode a node sends a special transaction. On this transaction network checks, if the candidate node holds a sufficient amount of tokens and is reachable on the provided IP address. If these conditions are satisfied - the masternode is added to the end of the global list and the amount of tokens is blocked.
2) **Masternode successfully produces a new block**
   When the turn comes a masternode should produce a new block and send it to the network within the specified amount of time.
3) **Masternode misses its turn or pushes a new block too fast (faster than the desired time) or issues an invalid block**
   The node is moved out of the list and the block should be produced as in 4.
4) **Masternode goes offline**
   Each node should periodically send "heartbeat" package to the network. If the node fails to send a package in time more than 3 times in a row of 6 checks it is removed from the global list.
5) **The node owner wishes to stop it manually**
   In this case, the node owner sends a special transaction to the network.

Once the transaction is processed the locked amount of tokens is freed and the node is removed from the global list.

2. Architecture

Based on the system requirements, the following list of system components is formed:

1) **REMME Core.** The main task of this component is to safely and reliably store self-signed certificates and their revocation status. When used in the public service, it is also responsible for payment processing.
    a) In the framework of REMME blockchain there is a separate type of nodes, called **master nodes**, which are responsible for handling new blocks production. In private network the list of master nodes is maintained by system administrators. In a public service it is maintained automatically as described in "Consensus algorithm" section.
    b) Anyone can run a node which stores the whole blockchain for verification purposes.
    c) Most of the clients are supposed to work in light node mode. In this case user is not required to store the whole blockchain, just some chunks of data needed to verify user's operations.
2) **Client software** (server-side integration) needed to verify provided REMME certificates.
3) **Bitcoin anchoring** system for improved auditability.
4) **Oracle for pricing lists updates.** Needed as there is a requirement of keeping prices bond to a fiat currency. Also, it is needed because the system needs to know prices for Bitcoin and Ethereum as it is linked to them within the part of transactions fee.
5) The system for **token cross-blockchain migration**.

3. Roadmap

1. **Q4, 2015**
    - Idea growth and concept validation
2. **2016**
    - REMME Core MVP V 0.1 with 2fa on Telegram based on Emercoin blockchain.

- 5 pilot projects.

3. **Q2 2017**
    - Blockchain Intensive Hackathon by Microsoft winner.
    - Memorandum of Understanding with Ukrinmash.
    - Strategic partnership with Infopulse.
4. **Q3 2017**
    - REMME Core MVP V 0.2 - CRL infrastructure on Bitcoin blockchain, certificate generation in a browser.
    - Memorandum of Understanding with DepositPhotos.
5. **Q4 2017**
    - December 4, 2017: Pre-sale for the REMME's community whitelist.
6. **Q1, 2018**
    - REMME Core public Alpha
    - Legal structure
    - Product's security audits and pen-tests
    - Extending team of software engineers
    - Public sale phase.
7. **Q2, 2018**
    - At this stage, private blockchains for integration with enterprise systems will be implemented. The planned functionality is:
        - Blockchain-based certificate data storage.
        - Integrations with different clients' systems.
        - Bitcoin anchoring for higher auditability.
    - Open source integration libraries for websites and web applications
    - Additional 2FA options, such as Signal, Status, WeChat, Trezor
    - Product's security audits and pen-tests
    - 20+ integrations
8. **Q3, 2018**
    - Public testing. At this stage a user can nominate itself to become a holder of a master node, which will be processed by the existing master nodes and, if the user is eligible by all criteria, they will be included to the list of approved nodes. At this stage, as it is the further development of PoA, master nodes from the list of approved nodes, will be added periodically by REMME maintainers. Prices will be updated in a centralized manner (see "Pricing policy") from REMME maintainers trusted node. New prices will take their place after 10 confirmation of the block they were introduced in. At this stage, the system will also have cross-blockchain token transfers so

everyone who purchased REMME tokens will be able to take the full advantage of the service.
- REMME Core integration with Active Directory and SCADA systems
- Extending of decentralized ecosystem of nodes
- Product's security audits and pen-tests
- Opening sales office in London: hiring dedicated sales responsible for EU, extending existing marketing team and hiring support team.
- 50+ integrations

9. **Q4, 2018**
    - At this stage, the release of the public system is planned, so the process of master node management on the public service will be fully automatic without involving REMME maintainers.
    - The consensus algorithm update.
    - REMME Core adoption for IoT (with a focus on automotive and smart cities)
    - Product's security audits and pen-tests
    - Opening sales office in New York: hiring dedicated sales responsible for the US, extending existing marketing team and hiring support team
    - 100+ integrations
10. **Q1, 2019**
    - Opening sales offices in Tokyo and Singapore: hiring dedicated sales responsible for Asia market, extending existing marketing team and hiring support team for this region
    - Holding special cybersecurity events, once each 3 months
    - Special cybersecurity lessons and classes in Ukraine for training specialists for contributing REMME ecosystem

4. Use cases

4.1. Example 1 (managed system with public data)

A cryptocurrency exchange service adopts REMME for clients authorization to replace the default system with login-password pairs. The number of users on the site – 25000, from this number 20000 users are active. Daily workload – 2000 users/day.

Monetization: the service pre-pay a required number of certificates (valid for 1 year) so that end users do not have to pay for certificates. Every year the service pay for the new certificates for its users.

Certificate validity: only for the service.

2FA: REMME provides the service with the software to provide messengers-based 2FA.

Users' capabilities:

- Generate a certificate.
- Quickly revoke the certificate in case of secret key compromising.
- Automatically get a certificate when the old one has expired.
- Select a preferred 2FA method.

Administrators' capabilities:

- Reliable payment system (with REMME tokens).
- System monitoring (issued and revoke certificates number).
- Root certificate management.
- Issuance of a new certificate when the old one has expired.

4.2. Example 2 (private blockchain)

Some state company wants to integrate REMME for users' authorization on its internal services. Capability: up to 1000 users.

Monetization: payments for integration and support.

Certificate validity: only for this service, use different certificates for different components of the system.

2FA: hardware token.

Users' capabilities:

- Generate a certificate.
- Quickly revoke the certificate in case of secret key compromising.

Administrators' capabilities:

- System monitoring (issued and revoke certificates number, active sessions).

- Root certificate management.
- Manage validity of certificates for different components of the system.
- Issuance of a new certificate when the old one has expired.
- Possibility to revoke users’ certificates as quickly as possible.
- Edit users’ data stored on the blockchain.

Conclusion

The goal of the REMME high-end secure system is to help organizations like infrastructure companies, IoT, medtech, financial and blockchain companies protect sensitive data. It is a distributed Public Key Infrastructure (“PKI”) management technology built on top of the X.509 certificate standard that uses SSL/TLS to protect the entire channel from an attack.
REM token is needed for all internal operations inside of ecosystem, so it is obviously utility token.
The Proof-of-Service consensus is used for the mix of high throughput, scalability and security.
The two-factor authentication is an additional layer of security ensuring that only the owner of the account can access it, even if the private SSL/TLS certificate key has been compromised. The choice of technology for the second factor depends on the characteristics of the system, which is protected by REMME technology.
During the development of the REMME system, a number of standard and well-known components that are time-tested and proven effective by numerous audits were used.
REMME will use the blockchain technology as a vehicle of transportation and a source of consensus to offer a solution to this problem: decentralization.

Useful links

1. Verizon data breach investigation report
2. http://www.certificate-transparency.org/
3. Proof-of-work description
4. Proof-of-stake description
5. Ethereum Clique PoA protocol description
6. X.509 specification
7. Our Github repositories
8. Waves gate to Ethereum

Annex 1: Bitcoin prototype

To solve the problems of centralization in current PKI systems, the concept of a system based on Bitcoin protocol was proposed. This concept formed the basis of the second version of REMME. Hereinafter, the use of self-signed X.509 certificates is implied. In this system, Bitcoin software serves the following tasks:

1) Certificates revocation management. Each certificate is bond with an output of a certain Bitcoin transaction. The certificate is considered invalid when this output is spent.
2) Certificates authentication. Each certificate stores a digital signature of a string (signed by the certificate holder, the string itself will be referred as "canary string") defined by REMME standard and the Bitcoin address of the certificate holder. Having that, the data of the certificate can be used to form the string mentioned above and check the validity of the signature using the given Bitcoin address.

In the framework of the system following fields of certificates are used:

`UID`	Bitcoin address of the certificate holder.
`L`	Digital signature of the canary string signed with Bitcoin `signmessage`.
`OU`	The identifier of a transaction used for certificate revocation management.
`ST`	The number of an output of the transaction mentioned above.

Canary string is formed from the following pattern: `https://REMME.io/certificate/$CERT_SN/$PUBKEY_HASH/$CERT_OU/$CERT_ST`, where:
`$CERT_SN` - certificate serial number;
`$PUBKEY_HASH` - SHA256 hash of the certificate public key;
`$CERT_OU` - the identifier of the used transaction;
`$CERT_ST` - the number of the used transaction output.
This string is signed with `signmessage` function of the standard Bitcoin implementation (Bitcoin Core) and included in the corresponding field of the certificate.
So the full process of creation of the REMME compatible certificate looks as follows:

1) Generate a key pair.
2) Create a Bitcoin transaction to be used for revocation management.

3) Create a certificate and fill its subject fields in according to REMME specification.
4) Create a canary string.
5) Sign the canary string with the `signmessage` function.
6) Include the canary string to the corresponding certificate field.
7) Sign the certificate.

To check the certificate:

1) Fetch the certificate.
2) Ensure that the associated transaction output is unspent (i.e. the certificate is valid).
3) Form the canary string from the certificate data.
4) Check the signature of the canary string included to the certificate using Bitcoin `verifymessage` function.

The proposed systems enable its users with the certificates management system based on the Bitcoin distributed storage. It is worth noting, that this system is highly portable because it is based on terms, which are essentially the same for most of existing blockchains.

Annex 2: Consensus algorithms brief comparison

Proof-of-work
Based on doing computationally expensive work to solve some task with the solution easy to be checked. Do not apply, because it is necessary to be independent of hardware resources.

Proof-of-stake
In this algorithm, the probability of issuing a new block is proportional to the amount of money held by a node. The drawback is that the network can be controlled by nodes which hold more tokens. This algorithm also incentivizes users to keep their tokens which is unacceptable for the REMME system.

Proof-of-importance
Basically the same as proof-of-stake, but also involves the activity of accounts (i.e. the number of transactions) and node uptime. Given that, the algorithm can be tricked by holding a large number of tokens or doing a lot of transactions (the primary goal of the desired system is uptime).

Proof-of-service
Introduced by Dash in its masternodes network. Requires masternodes to be online as long as possible. The more time a masternode is online, the higher position it gets in masternodes ranking and the higher is its income. REMME use a certain variation of proof-as-service as a consensus algorithm.

Proof-of-authority
Designed to be used in private networks and includes a list of approved nodes which are submitting blocks in a pre-defined order.

Annex 3: Blockchain frameworks comparison

Requirements

1) Consensus should fit the requirements in the related document or should be replaced without a great effort.
2) The blockchain itself should be able to store arbitrary data.
3) Light nodes.
4) Both private and public (anyone can create a masternode on certain conditions) network modes.
5) No additional network fees.

Comparison

Blockchain	Consensus	Data storage	Light nodes	Private	Public	No fees
Ethereum	Proof-of-work or proof-of-authority (private)	+ (smart-contracts)	+	+	+	-
Exonum	Proof-of-lock	+	+	+	-	+
Rootstock	Proof-of-work	+ (smart-contracts)	+	-	+	-
Fabric	Any	+	+	+	-	+
Sawtooth	Any	+	+	+	+	+
Dash	Mixed (proof-of-service + proof-of-work)	-	+	-	+	-
Stratis	Proof-of-stake	+	-	+	+	-

NOTES:

1) Dash masternode network is suitable (and was our inspiration) but dash source code will require a significant rework.
2) Stratis is bound to .NET technology stack.
3) Exonum is said to have public networks
4) Fabric can be reworked to be unpermissioned via pluggable modules
5) Stratis lacks documentation on their website

Annex 4: Token sale

REMME Token (REM) is designed to perform a function of a pre-order access key that enables access to the software program ones it is developed. REM Token enables pre-order by functioning as a key enabling access.

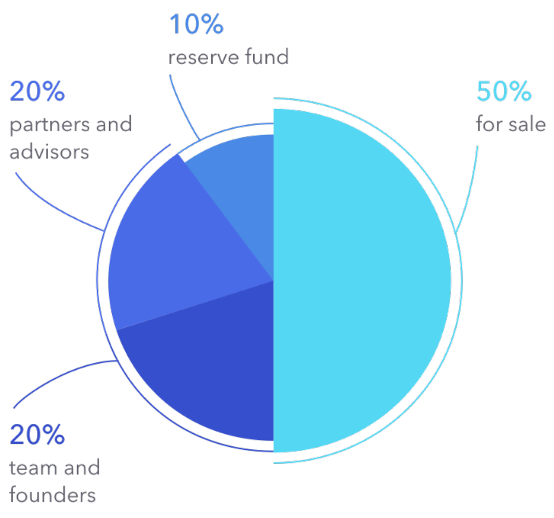
REM Token Utility

REMME uses blockchain technology to create a distributed certificate management system that has no single point of failure. And REM token is the superpower boosting the whole ecosystem, operating like a license or digital key and granting its holders access to REMME PKI and DApps. Token holders will be able to use the REM token in a variety of ways: certificate generation, revocation, node creation, developing DApps, fees covering conversion of fiat payments, etc.

Details overview

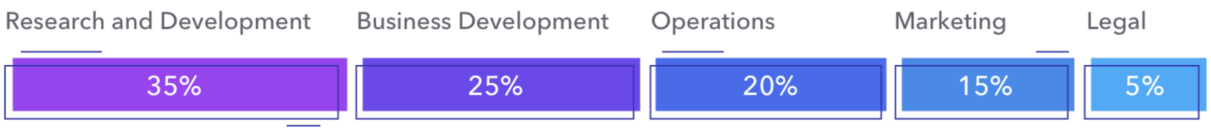
Total token supply: 1 billion
Hard cap (including pre-sale stage): $20 M
Soft cap: $480,000 (reached during Pre-Sale)
Initial price: 1 REM = $0.04
Type: ERC-20
Currencies accepted: ETH, BTC
Pre-sale dates: December 4 - December 25
Public sale start: February 13th, 2018 (20:00, UTC)
Whitelist: yes. Start date TBA
KYC: basic
Country restriction: everyone can participate

Token distribution

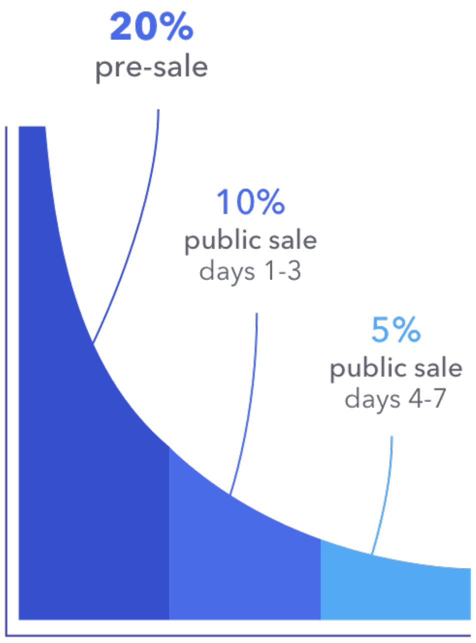


Lock-up:
1. Partners and advisors - 10% of tokens will have 6 months vesting with 3 months cliff.
2. Team and founders - 2 years vesting with 6 months cliff.

Use of proceeds



Bonuses



Unsold tokens

Will be locked for up to 3 years:

- 50% locked for 1 year;
- 25% locked for 2 years;
- 25% locked for 3 years.