remme

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

Whitepaper

v.0.3
January, 2018

remme.io

Introduction 2

1. The Problem 2

2. Current Solutions 3

3. Public Key Infrastructure (PKI) 5

4. The Remme Solution 6
    4.1 Advantages of REMME: 7
    4.2 Additional advantages: 7

5. REMME product design and evolution 8

5.1. Certificate support 9

5.2. Two-Factor Authentication 9

5.3. Token economy and pricing model 10
    5.3.1 Fee distribution 11

5.4 Inter-blockchain token migration 11

5.5 Consensus algorithm 11
    5.5.1. Requirements 11
    5.5.3. Masternode Functionality 13

6. Architecture 13

7. Roadmap 14

8. Remme use cases 16
    8.1. Example 1 (managed system with public data) 16
    8.2. Example 2 (private blockchain) 17

Conclusion 18

Useful links 18

Annex 1: Bitcoin prototype 19

Annex 2: Consensus algorithms brief comparison 21

Annex 3: Blockchain frameworks comparison 22

Annex 4: Token sale 23

Introduction

Traditionally, user authentication and authorization has most commonly been accomplished through the use of user passwords. It is a method that has proven itself a convenient means of identifying users and securing their access to information and resources. However, the challenges and risks associated with password based authentication are numerous; ranging from a difficult user experience through to fundamental security issues which can result in the compromise of accounts and data secured by this method.

While there are alternate, more sophisticated approaches to authenticating and authorizing a user without the reliance upon passwords, these solutions require technically complex architectures which entail high costs and expertise to administer and maintain.

As a result, the use of password based authorization has proliferated, despite its flaws.

1. The Problem

By their very nature, passwords impose a burden and expectation of sound security practices on the end user and it is this very characteristic which often leads to compromise.

It is widely known in information security circles (and by users themselves): When forced to use antiquated password based authentication, users frequently take the path of least resistance - storing passwords insecurely, reusing passwords, using insecure passwords, sharing passwords, the list goes on.

Current systems attempt to account for these user behaviours in various ways, with varying degrees of success, but no solution can eliminate the inherent flaws in the password model entirely.

In enterprise environments, the use of passwords spawns a routine of endless training and security awareness programs, detailed usage policies (administrative controls) and centralized systems enforcing password length, complexity, and expiry (technical controls). Such systems will also monitor and track attempted logins, disabling access entirely when predefined thresholds are met.

While such countermeasures may reduce the rate of compromise, these are band-aid solutions to the fundamental flaws of the model. In addition to the aforementioned problems, the attack surface of passwords is large, providing bad actors with numerous avenues to compromise their target. Typical methods include:

- Brute force attacks
  - Dictionary attacks
  - Rainbow tables run against password hashes
- Packet sniffing
- Keyloggers
- Social engineering
  - Phishing
  - Coercion
  - Shoulder surfing
- Man in the middle attacks

The many shortcomings of password based access control are well known within the industry and many interim measures have been proposed and gained adoption in an effort to mitigate or reduce the impact of these issues. Yet, the problems remain.

One of the common challenges in deploying and managing password based access control systems is the delicate balance between usability and security; an increase in either area typically results in a decrease in its counterpart. As a result, most solutions focus on one area or the other.

Clearly, the password alone can not serve as a sufficient instrument for protecting user data and guaranteeing the security of the user session. Any proposed solution to this problem must not only address the numerous security flaws in the password model, but also aim to reduce the burden of security on the end user without sacrificing security or decreasing usability.

From an information security perspective, technical controls are always preferable to administrative controls as they are not dependent upon user behaviour.

2. Current Solutions

The Password Manager

One commonly used approach to reducing the overhead of password management is known as a password manager. This software, typically deployed as a standalone

client, a web based service, or a browser plugin, provides centralized secure storage of passwords including the ability to generate secure passwords, unique to each resource they protect. The obvious drawback to this approach is that the password manager itself is protected with a master password (Keepass, Lastpass, 1Password).

In the case of theft, loss or brute force of the master password, all accounts secured by the password manager are at risk. While this solution reduces the burden of password management, it does not eliminate it. More concerning, the work factor required for an attacker to compromise all user accounts stored within is greatly reduced as only a single password must be compromised for a bad actor to gain complete control over its target.

Two-Factor Authentication

Two-factor authentication (2FA) provides a supplementary solution to password security challenges. Using this method, security is strengthened by requiring two of three means of authentication: something you know (password, passphrase), something you have (phone, hardware token, swipe card) or something you are (biometrics).

The most widely used example of two-factor authentication is a bank debit card. The card itself is the first factor (something you have) with the PIN serving as the second factor (something you know). There are many different implementations of two-factor authentication, each with its own strengths and weaknesses.

Standard implementations of 2FA include:

- Software tokens: One-time passwords that are generated every n seconds are the most common option. Usually implemented with TOTP protocol.
- One-time passwords sent via an out-of-band message. Usually, SMS, email or instant messenger is used.
- Hardware tokens.

While 2FA adds an additional layer of security to password based access control, it does not solve the fundamental issues and challenges of password management noted above.

3. Public Key Infrastructure (PKI)

When passwords are used for authentication, the process requires that, the password is disclosed - which makes it vulnerable to compromise. This is comparable to proving ownership of your home by handing a stranger your house keys. Surely, a copy of the deed would accomplish the same without sacrificing personal security. In this scenario, a digital certificate serves the same purpose: Providing proof of authorization and ownership, without potentially compromising the very asset it is protecting.

Utilizing Public Key Infrastructure (PKI), authentication can be accomplished through the use of SSL/TLS certificates. While this method is widely adopted in government and enterprise solutions for the protection of sensitive information (classified, financial, etc.), the complexity and cost of deploying Public Key Infrastructure has prevented widespread adoption of this technology to replace password based access control systems. This technology enables the secure exchange of data over untrusted environments such as the internet while providing the ability to verify the identity of either or both parties.

The PKI framework encompasses security policies, procedures, standards and software which collectively allow for the secure exchange of information. However, we should understand that the core components of PKI are digital certificates and public key cryptography.

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.

Public key cryptography (the use of public and private keys) provides the services of authentication and confidentiality. However, the mere existence of public and private keys is not enough to ensure trust.

There must be a comprehensive system in place to irrefutably confirm the authenticity and integrity of the keys and certificates. The PKI framework accomplishes this goal through the use of trusted authorities with different roles as shown below. At a high level, the typical deployment of PKI includes the following elements:

- Certification Authority (CA) - a trusted party provides services for issuing digital certificates.

- Registration Authority (RA) - a trusted party responsible for accepting requests for digital certificates and authenticating the entity making the request.
- Validation Authority (VA) - a trusted party providing a service used to verify the validity of a digital certificate

As a whole, the PKI framework provides the following services:

- Authentication and authorization
- Non-repudiation
- Message integrity
- Confidentiality

Known as a hybrid system, PKI utilizes both symmetric and asymmetric key algorithms and methods to provide these services.

4. The Remme Solution

REMME is bringing PKI infrastructure to the blockchain, providing a full end-to-end, cost effective solution to the shortcomings of password based access control systems.

Remme will focus on several core issues. Firstly, due to its complexity and cost, the decision to deploy PKI has historically been either government regulated or business-driven. Furthermore, third party PKI services have not been a cost effective solution for client authentication. Thus, PKI as a replacement to password based client authentication has not been widely implemented. Remme intends to reduce the barriers to entry and foster the widespread adoption of PKI as a replacement to password based authentication.

Additionally, in its current centralized form, there is potential for collusion between software vendors and the CA pertaining to the inclusion of a specific CA onto a list of trusted software. Decentralization through the use of blockchain technology eliminates the need for trust to be placed in a third party CA.

Some time ago, there was a brilliant vision proposed named “web of trust” (PGP is a widely known implementation), where most of the noted problems could be resolved through the decentralization of the aforementioned PKI roles. Unfortunately, while still in use, this largely remained a vision and did not see widespread adoption.

The Remme solution will provide our customers a means of deploying and managing PKI with the utmost level of security and all the advantages of a blockchain based, decentralized and distributed system including immutability, high availability and fault tolerance.

4.1 Advantages of REMME:

- Elimination of passwords resulting in increased security, lower management overhead, and a drastically improved user experience.
- Decentralized, immutable point of trust for different systems: Easy single-sign-on implementation with worldwide authorization available out of the box.
- Reduced cost, complexity and overhead in PKI deployment and management.
- There are no legal limitations or government cooperation issues. In fact, the use of PKI with 2FA is in line with information security best practices and helps satisfy the criteria of due diligence.
- There is no centralized database of certificates and keys that could be compromised.
- There is no technology lock-in and no API limitations. Easy integration with existing systems is provided.
- There are no additional fees for different certificates in different CAs.
- Trustless by design, Remme removes the need for dependency on a third party CA.
- Fast and protected public key distribution and certificate revocation process.

4.2 Additional advantages:

- Protection against Phishing and MITM attacks.
- Eliminates the numerous attack vectors of password based systems.
- Integration with a multitude of multi-factor authentication solutions.
- Full anonymity provided by default.
- Complete transparency for the issuance, tracking and management of certificates.

For non-technical users, this framework likely appears complicated. However, from the end user perspective, it is quite simple: there is no need to concern oneself with password management or to perform the mental gymnastics necessary to create, store and recall secure passwords. Furthermore, for administrators, there is no need

to design and deploy Public Key Infrastructure or to pay numerous authorities for certificates required for different purposes. Nor is there a need to manage the lifetime of each password/certificate/key.

5. REMME product design and evolution

The Remme team developed an early proof of concept by creating a decentralized PKI on the Emercoin blockchain (EMCSSL). Although this is a production-ready system, testing revealed its performance to be inadequate for conducting efficient PKI operations.

Another concept, based on Bitcoin blockchain UTXOs (see Annex 1) was then tested, but it was decided to move forward with our own custom blockchain for two main reasons:

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

The Remme blockchain will provide the scaling and performance necessary to consistently perform efficient PKI operations while also providing a financial framework which meets our requirements.

For such a system the three main requirements are as follows:

1) The certificate pricing policy should be bound to an existing currency. For example, the US dollar.
2) The blockchain should provide high throughput for the management of PKI operations.
3) The possibility to deploy a private blockchain should be available to meet organizational requirements.

To meet these requirements, Remme developed a custom blockchain derived from Hyperledger Sawtooth (see the detailed information on blockchain selection in Annex 3) and a custom consensus algorithm heavily inspired by Ethereum Clique and Dash masternodes network.

Hyperledger Sawtooth provides a framework for the development of modular, custom blockchains - every component of the framework including the consensus

algorithm can be replaced with custom code. The consensus algorithm is, in a nutshell, Clique – a simple PoA protocol which runs on top of the pre-defined list of nodes with an addendum which makes it somewhat similar to the Dash masternodes concept: Every user can enter the list of signing nodes by locking a predefined number of tokens and will be excluded from the list if their node works against the network rules or does not meet availability requirements.

5.1. Certificate support

The Remme system has been designed to support the industry accepted X.509 certificate standard with support for the following use cases:

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) Certificates signed by an organization: In this case an organization (which is our client) may use its own CA to sign and manage certificates of its clients and employees.

To support REMME-based certificates, Remme is developing server-side software which will continuously verify the status of the certificate on the blockchain. There are numerous ways of achieving this such as implementing plugins for enterprise systems or incorporating self-signed certificates into the system's 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.

5.2. Two-Factor Authentication

The choice of technology for the second factor is dependent upon the requirements of the system to be protected by REMME technology. For example, if a system requires the physical presence of an authorized person, then the best option for the second factor would be to use biometric data, such as a fingerprint or eye retina scan.

Alternatively, if the system uses a local sensor, then the second factor would be the physical connection to the local network. This scenario could be extended to integrate with existing network policy servers and 802.1x implementations. If the system is remote, it is optimal to use a secondary out-of-band device (a software or hardware token). The probability of simultaneous malware infection of two devices is extremely low and this approach also provides a secondary measure of protection in the event of a certificate becoming compromised.

For SMB clients with limited budgets or infrastructure, 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 chosen application. 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. It is also possible to use an entirely hardware-based solution for generating access tokens such as RSA Securid, YubiKey, Yubico or Trezor.

5.3. Token economy and pricing model

As one of the requirements is to bind the price of a certificate to a fiat currency, the platform requires a means of updating the prices for its services with respect to the exchange rate of the token.

To address this requirement, REMME will maintain a highly available server which will periodically (every 10 minutes) update prices. Every masternode will query this server to obtain the price of a token at a given moment in time.

[fig 3]

The REM token is required for all operations within the ecosystem. This includes:

- Initialization of the certificate creation process: An amount of tokens is reserved for certificate revocation transactions ($1).
- Establishing a node: As per our roadmap, this will be available after Q3 2018. (Approximately 250,000 REM tokens will be required for the establishment of a masternode).

- Fee for transferring REMME tokens between users on the REMME blockchain (0.1%).

5.3.1 Fee 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:

- Under $5m collected during the public token sale: 70% of these fees are stored on the account of a masternode which created the current block, 30% are transferred directly to REMME and are used to maintain services such as Bitcoin anchoring, cross-blockchain transfers etc.

- Between $5m and $10m collected during the public token sale: 80% of these fees are stored on the account of a masternode which created the current block, 20% are transferred directly to REMME and are used to maintain services such as Bitcoin anchoring, cross-blockchain transfers etc.

- Over $10m collected during the public token sale: 90% of these fees are stored on the account of a masternode which created the current block, 10% are transferred directly to REMME and are used to maintain services such as Bitcoin anchoring, cross-blockchain transfers etc.

5.4 Inter-blockchain token migration

Inter-blockchain token migration is required to provide a migration mechanism which will enable users to use the REM token, initially released on the Ethereum platform, on the REMME blockchain. As for the current state of the project, the optimal choice is a system similar to Waves cryptocurrency gates.

The planned functionality is that one of the nodes will monitor 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), the same amount of tokens is sent to the user's account on REMME. The system works in the same manner when transferring REM to the Ethereum blockchain.

5.5 Consensus algorithm

5.5.1. 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:

- Consistent response time.
- Hardware independent.
- Incentivize validating users to meet availability requirements.

5.5.2. Masternodes and proof-of-service

Masternodes will utilize the proof-of-service algorithm, which was first introduced in the Dash masternodes network. Proof-of-service features include:

- No dependency on on hardware capabilities.
- Can incentivize users to stay online by ranking them by their uptime.
- Suitable for public networks.

This algorithm can be also described as a form of proof-of-stake, with the additional factor of node availability (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 the one introduced in Dash, in that masternodes (validators) will be responsible for producing new blocks and validating transactions.

The main entity in this algorithm is the global list, containing the full list of nodes which are eligible to sign new blocks. When a new node is introduced into this list, it is placed at the end of the list. Node selection is governed by the order nodes were introduced to the list and the algorithm specifies the desired and the maximum time in which blocks should be produced. Blocks produced faster than in the desired time will be rejected. If these requirements are not satisfied, the node is removed from the global list and must re-enroll to enter the list again.

Producing a block is rewarded by fees from transactions included within the block (see pricing policy for 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 uses timing between blocks to confirm nodes are operating as expected.

5.5.3. Masternode Functionality

To make the working process of the algorithm clear, some scenarios are outlined below:

- Application for masternode status
  To become a masternode, a node sends a specially formatted transaction. The transaction is verified to confirm 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 tokens are locked.

- A Masternode successfully produces a new block
  Once it is eligible, a masternode must produce a new block and send it to the network within the specified time frame.

- A Masternode misses its turn, pushes a new block too fast (faster than the desired time) or issues an invalid block
  The node is removed from the list and the block will be produced by the next eligible node.

- Management of masternode availability
  Each node must periodically send a heartbeat to the network. If any node fails to send a heartbeat within time constraints at a rate of fifty percent or higher from a sample of six checks, it is removed from the global list.

- Decommissioning of a masternode
  In this case, the node owner sends a specially formatted transaction to the network. Once the transaction is processed, the locked tokens are released and the node is removed from the global list.

6. Architecture

Based on the system requirements outlined above, the following components form the high level REMME architecture:

- REMME Core: The main functionality of this component is to securely and reliably store CA generated or self-signed certificates and their revocation

status. When deployed on public networks, it is also responsible for payment processing.
    - Masternodes: In the framework of the REMME blockchain, masternodes are responsible for the production of new blocks. In private networks, the list of masternodes is maintained by system administrators. In a public network deployment it is decentralized as described in the "Consensus algorithm" section.
    - Nodes: Most clients are expected to operate in light node mode. In this case the user is not required to store the whole blockchain, only the data needed to verify the user's operations. Anyone can run a node which stores the whole blockchain for verification purposes.

- Client software (server-side integration): Required to verify that REMME certificates have been issued and are valid.

- Bitcoin anchoring system: Provides improved auditability.

- Oracle for pricing list updates: This is needed as there is a requirement to ensure certificate prices are bound to a fiat currency and remain stable. It is also necessary because the system needs to know prices for Bitcoin and Ethereum as it is linked to them when it comes to transaction fees.

- The system for token cross-blockchain migration.

7. Roadmap

- Q4, 2015
    - Idea development, feasibility study and concept validation.

- 2016
    - REMME Core MVP V 0.1 with 2FA on Telegram based on Emercoin blockchain.
    - 5 pilot projects.

- Q2 2017
    - Blockchain Intensive Hackathon by Microsoft winner.
    - Memorandum of Understanding with Ukrinmash.
    - Strategic partnership with Infopulse.

- Q3, 2017

- REMME Core MVP V 0.2 - CRL infrastructure on Bitcoin blockchain, certificate generation in a browser.
- Memorandum of Understanding with DepositPhotos.

- Q4, 2017
  - December 4, 2017: Pre-sale for the REMME community whitelist.

- Q1, 2018
  - REMME Core public Alpha.
  - Legal structure.
  - Product security audits and pen-tests.
  - Extending team of software engineers .
  - Public sale phase.

- Q2, 2018
  - At this stage, private blockchains for integration with enterprise systems will be implemented. The planned functionality is:
    1. Blockchain-based certificate data storage.
    2. Integrations with different clients' systems.
    3. Bitcoin anchoring for higher auditability.
  - Open source integration libraries for websites and web applications.
  - Additional 2FA options, such as Signal, Status, WeChat, Trezor.
  - Ongoing product security audits and pen-tests.
  - 20+ integrations.

- Q3, 2018
  - Public testing. Masternode support is planned at this stage of development. Submissions for masternode status will be processed by the existing masternodes and, if the node is eligible by all criteria, it will be included in the list of approved nodes. Following further development of the PoA algorithm, masternodes from the list of approved nodes will be added periodically by REMME maintainers. Prices will be updated (see "Pricing policy") from a REMME maintainer's trusted node, with new prices taking effect after 10 confirmations of the block they were introduced in. At this point in the product development, the system will support cross-blockchain token transfers allowing holders of REMME tokens to take full advantage of the service.
  - REMME Core integration with Active Directory and SCADA systems.
  - Extending of decentralized ecosystem of nodes.
  - Ongoing product security audits and pen-tests.

- Opening of sales office in London: hiring dedicated sales team.responsible for the EU, extending existing marketing team and hiring support team.
- 50+ integrations.

- Q4, 2018
  - At this stage, the release of the public system is planned and the process of master node management on the public network will be fully automated and REMME maintainers decommissioned.
  - Consensus algorithm update.
  - REMME Core adoption for IoT (with a focus on automotive and smart cities).
  - Ongoing product security audits and pen-tests.
  - Opening of sales office in New York: Hiring dedicated sales team. responsible for the US, extending existing marketing team and hiring support team for the US market.
  - 100+ integrations.

- Q1, 2019
  - Opening of sales offices in Tokyo and Singapore: Hiring dedicated sales team responsible for Asian market, extending existing marketing team and hiring support team for this region.
  - Holding special cybersecurity events, once every 3 months.
  - Special cybersecurity lessons and classes in Ukraine for training specialists for contributing to the REMME ecosystem.

8. Remme use cases

8.1. Example 1 (managed system with public data)

A cryptocurrency exchange service adopts REMME for client authorization to replace the default password based authentication system. The number of users on the site is 25,000, and of this number, 20,000 users are active. Daily workload is 2,000 users per day.

Monetization: The service provider purchases the required number of certificates (valid for 1 year). Additional certificates are purchased as required and existing certificates are renewed yearly.

Certificate validity: Only for the agreed upon service.

2FA: REMME provides software to enable messenger-based 2FA, additional 2FA options can be negotiated.

User capabilities:

- Generate a certificate.
- Quickly revoke the certificate in the case of private key being compromised.
- Automatically obtain a new certificate upon expiration of the existing certificate.
- Select a preferred 2FA method.

Administrator capabilities:

- Manage costs and payments (with REMME tokens).
- System monitoring (issued and revoked certificate numbers).
- Root certificate management.
- Issuance of a new certificate upon expiration of the existing certificate.

8.2. Example 2 (private blockchain)

A state company plans to integrate REMME for user authorization on its internal services. Capability: up to 1,000 users.

Monetization: Payments for integration and support.

Certificate validity: Only for this service, using different certificates for different components of the system.

2FA: Hardware token.

User capabilities:

- Generate a certificate.
- Quickly revoke the certificate in the case of the private key being compromised.

Administrator capabilities:

- System monitoring (issued and revoked certificate numbers, active sessions).
- Root certificate management

- Manage validity of certificates for different components of the system.
- Issuance of a new certificate, revocation of existing certificates.
- Editing of user data stored on the blockchain.

Conclusion

The goal of REMME's revolutionary technology is to help organizations such as infrastructure companies, IoT, medtech, financial and blockchain companies protect sensitive data without developing and maintaining costly and complex infrastructure. It is a distributed Public Key Infrastructure management technology built on top of the X.509 certificate standard that uses SSL/TLS to protect the entire channel from an attack.

The REM token is required for all operations within the ecosystem, and as such, functions as a utility token. The Proof-of-Service consensus model has been selected for its traits of high throughput, scalability and security.

Two-factor authentication provides an additional layer of security, ensuring the account remains secure even if the private SSL/TLS certificate has been compromised. The choice of technology for the second factor depends on the requirements of the system to be protected by REMME technology.

REMME will leverage blockchain technology as a vehicle of transmission and a provider of consensus and immutability to deliver a turnkey solution to the flaws of the password based access control model.

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 the 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, the Bitcoin network serves the following functions:

- Certificate revocation management. Each certificate is bound with the output of a specific Bitcoin transaction. The certificate is considered invalid when this output is spent.
- Certificate 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 the REMME standard and the Bitcoin address of the certificate holder. The data of the certificate will be used to form the string mentioned above and confirm the validity of the signature using the given Bitcoin address.

In the framework of the system, the 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.

The canary string is defined 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 the signmessage function of the standard Bitcoin implementation (Bitcoin Core) and included in the corresponding field of the certificate.

The full process of creation of a REMME compatible certificate is as follows:

1) Generate a key pair.
2) Create a Bitcoin transaction to be used for revocation management.
3) Create a certificate and update its subject fields according to REMME specifications.
4) Create a canary string.
5) Sign the canary string with the `signmessage` function.
6) Include the canary string in the corresponding certificate field.
7) Sign the certificate.

To verify the certificate:

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

This proposed system enables blockchain based PKI built upon Bitcoin distributed storage. It is worth noting that this system is designed to be blockchain agnostic as it is based upon concepts which are supported on most existing blockchains.

Annex 2: Consensus algorithms brief comparison

Proof-of-work
Based on performing computationally expensive work to solve tasks with a solution which can be easily verified. Not suitable for REMME due to our requirement 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 tokens 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
Similar to proof-of-stake, but also involves the activity of accounts (i.e. the number of transactions) and node uptime measurements. Given that the algorithm can be manipulated by holding a large number of tokens or by completing a large number of transactions (the primary goal of the desired system is uptime), this system is undesirable.

Proof-of-service
Introduced by Dash in its masternodes network, proof-of-service incentivizes masternodes to achieve the highest amount of uptime possible. The longer a masternode remains online, the higher its position in the masternodes ranking and the higher its income. REMME uses a 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 submit blocks in a pre-defined order.

Annex 3: Blockchain frameworks comparison

Requirements

- Consensus must satisfy the requirements outlined in this document or should be replaceable without great effort.
- The blockchain itself should be able to store arbitrary data.
- Light nodes are required.
- Both private and public (anyone can create a masternode under specific conditions) network modes.

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:

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

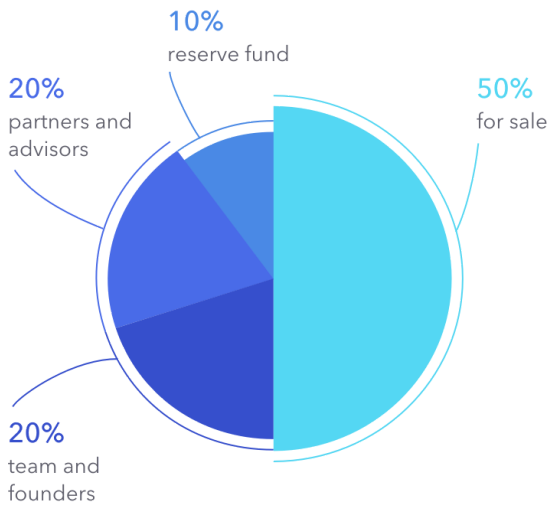
Annex 4: Token sale

The REMME token (REM) is designed to perform many functions. As it pertains to the token sale, the REM token acts as a pre-order access key which guarantees access to the software program once it is developed.

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

Detailed 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: No restriction

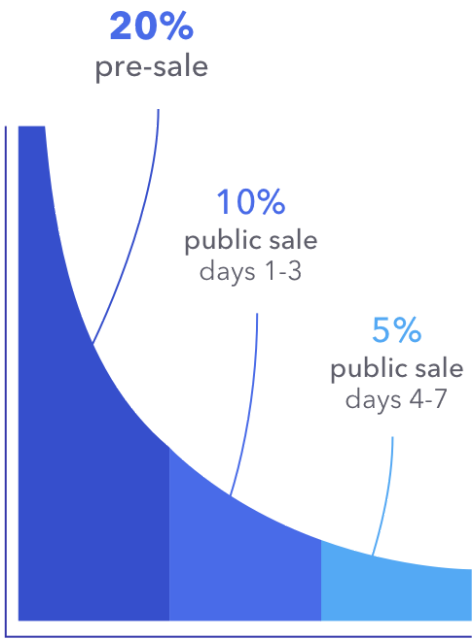
Token distribution:



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.