Nidaba: a distributed scalable PKI with stable price for certificate operations

Denis Rystsov

29 April 2014 (attested by Bitcoin)

Abstract

A distributed PKI would solve the problem of trust to the authorities in the key technologies behind the Internet. It's creation is a complex task, since the system has to be scalable, has to provide a stable price for certificate registration and has to prevent cyber squatting.

Nidaba has all the mentioned properties, moreover it protects a blockchain from being forked behindhand and it has the same level of protection as Bitcoin does.

This paper describes an architecture behind Nidaba.

1 Introduction

Nidaba is a distributed PKI[1] based on the Bitcoin protocol[2]. On the low level Nidaba is a hash table where a key is a unique name (domain) and a value is a public key. An element of the hash table — a pair of a unique name and a public name — is called a certificate.

Nidaba can be used to build a distributed certificate authority (CA), a distributed OpenID[3] or a distributed DNS.

Let's describe how to build a Nidaba based distributed CA. First we choose a pair of public and private keys and announce them both. The pair is a marker: all ssl-certificates signed with the private key should be processed in a special

[1] Public key infrastructure, http://en.wikipedia.org/wiki/Public_key_infrastructure

[2] A Peer-to-Peer Electronic Cash System, http://bitcoin.org/bitcoin.pdf

[3] http://en.wikipedia.org/wiki/OpenID

way. An owner of a site who wishes to use the distributed CA should choose a private key, put a corresponding public key to Nidaba using a canonical URL of the site as a unique name, issue a certificate using the openly known private key and use the certificate as a normal ssl-certificate. If a site's visitor supports a distributed CA she recognizes the marker and checks whether a certificate's public key matches a value retrieved from Nidaba by the site's canonical URL. If the visitor doesn't support a distributed CA she would be warned that the site uses an untrusted certificate authorities.

The distributed CA would decouple authentication from a distribution channel. If this idea is applied to DNS it produces a distributed DNS: we just need to sign an address of a domain with a private key corresponding to a public key stored in Nidaba with domain as a unique name and to distribute the signed address in a Kadmilia-like network.

Also it is easy to use Nidaba as a distributed OpenID: if a person uses a name of his public key in Nidaba as a username then to prove his identity a site might ask the person to sign a provided message with his private key.

Before we discuss the architecture of Nidaba we should answer several questions: "Why should the PKI be distributed?" and "Why should we provide a stable price for certificate registration?".

1.1 Why should the PKI be distributed?

User registry system is a vital part of many technologies like DNS and HTTPS; if any is compromised it may lead to disastrous effects, so they must be secure and reliable. If the technology is centralized it carries some risks that can't be eliminated by technical means.

A company behide the centralized solution may fail or change it’s strategy. If Facebook goes bankrupt then all sites that use Facebook as an OAuth provider are also affected, because the users of these sites can't prove thier identity.

Moreover a government may force a company to compromise data of its users. For example if a government influences the ICANN and any trusted certificate authority it can organize an invisible man-in-the-middle attack on any opposition site.

1.2 Why should we provide a stable price for certificate registration?

This question splits into two: "Why should we pay for a distributed solution" and "Why should the price for a certificate registration be stable?".

A distributed solution works if at any moment of time there is a sufficient number of active nodes. If work of the nodes is voluntary or is being paid from donations, there is a risk of "tragedy of the commons": each participant of the network would think that her contribution to the greater good is minor and ignores it.

If the price is not controlled there is a risk that it would be too high or too low. If it is too high nobody continues using the system; if it is too low the activity of cyber squatters and vandals would increase. So there is a need for rules to hold a price in a fixed corridor.

1.3 Analogs

There are several user registry systems based on a Bitcoin protocol, Namecoin and Twister are among them.

Namecoin[4] is deflationary cryptocurrency similar to Bitcoin, Namecoin protocol extends Bitoin protocol with several domain operations. Each domain is unique and has an associated value. To register a new domain you need to get an intrinsic currency, NMC, and to pay the network and transaction fees. Network fee was introduced to decrease cyber squatter's activity, now its value is too low to do it. Mining reward is exponentialy decreasing and when it comes to zero miners would start working only for transaction fee, so there is a risk of the "tragedy of the commons" situation: a miner gains more profit right now if she includes transactions with any transaction fee, but this behavior drives average transaction fee below mining profitability in the future.

Twister[5] - is a distributed microblogging service. One part of it is a distributed name registry, anyone can register a name free of charge and people also keep their nodes active free of charge. Hashcash-like scheme is used to protect from cyber squatters.

The Namecoin economy is not stable, it may suffer from the tragedy of the commons and there aren't mechanisms to control the price of name registration.

[4] Wikipedia. Namecoin, http://en.wikipedia.org/wiki/Namecoin

[5] Twister - a P2P microblogging platform, http://arxiv.org/pdf/1312.7152v1.pdf

Twister hasn't economy and also may suffer from the tragedy of the commons since all activity is voluntary.

Twister and Namecoin are not scalable, ignore cyber squatting and don't adjust to raising computer power which may lead to vandalism.

2 Price stability

Lets see how Nidaba provides price stability.

Miners get reward for the work for the benefit of the network. A miner can't transfer this reward to an another person, but can use it to register a new certificate or to prolong an existing one. This scheme allows to create a market between reward holders and persons willing to register or to prolong a certificate.

To register or to prolong a certificate it is necessary to destroy some quantity of reward. This quantity (price) is constant on a small interval of time, low in the article there are rules describing how this quantity is changed over time.

Base price is a minimal quantity of reward to prolong a certificate for a minimum interval of time. If we speak about a specific certificate, the price is called a price of ownership of the certificate. A price of ownership is equal or greater than the base price. It may be higher because of an auction for the name. A price of ownership is expressed as a multiplier to the base price.

Base price is adjusted over time to reflect an increase of average miner's performance, but at the start of the network it is chosen equal to the amount allowing an average miner to earn the reward to prolong a certificate for one year in one month.

A miner's reward for block consists of two parts. The main part is proportional to the current difficulty. Since the current difficulty is proportional to the current network power then an expected value of miner's profit depends only on her own power. Thus a miner who joined the network at the beginning and a miner who did it when the network is mature would have gained an equal reward if they had the same power.

The second part of the reward for a block is equal to 10% of means spent on registration and prolongation in the block. It adds an incentive for miners to include transactions into a block.

An owner of a certificate must prolong it once in several blocks. For the convenience each certificate has an account which is being used to automatically withdraw means for the prolongation. An owner may transfer an arbitrary amount of means to prolong a certificate for some time ahead.

To avoid an overproduction of reward it has an expiration date, after which it can't be transferred to a certificate account. An expiration date adds the reward more similarities with futures contracts than with gold. So a reward is also referred below as futures or means. Besides an expiration date there is an account fee to avoid overproduction: once in several blocks a small percent of means (account fee) are being withdrawn from a certificate account. An accumulated account fee for a year is equal to 50% of the means on the account in the beginning of the year. This rule makes more profitable to prolong a certificate for a short period of time several times a year than once for a whole year.

This scheme provides a permanent and uniform demand for reward.

Given rules provide an equilibrium of price for certificate operations: if there is an overproduction of futures then the price per future (price per certificate operation) goes down, some miners leave the network because the mining becomes unprofitable for them, the network power decreases until the supply of futures comes down to the demand level. If the demand for futures rises then the price per future goes up, it attracts new miners, the network power increases until the supply of futures meets the demand level.

3 Base price adjusting

An average miner's performance increases over time, so the production cost (time) to generate futures decreases and the price for certificate operations decreases too. This violates the principle of price stability. So we need to adjust base price according to increase of an average miner's performance to keep price per certificate operations stable.

There is a trick to figure out an average miner's performance: a registrar may do an arbitrary amount of work when she register or prolong a certificate, if she hits the top 10% productive registrars (measured by that work) she will receive a 50% of payment for certificate operation back.

It is possible to measure an average performance based on a submitted information. According to increasing of the average performance the base price must be increased. A price adjusting is being run if the last price adjusting was a year ago or an average performance has changed more than twice.

4 Cyber squatting

It is impossible to avoid argues about the names. If it is a centralized system the company behind the system may be an arbiter (like in a UDRP case with DNS). This scheme is impossible with distributed systems.

Nidaba uses a penny auction to resolve the name issue:

1. after a reward holder registers a certificate for an user, the user is the owner of the certificate, the price of ownership matches the base price but the certificate's name is automatically put up for auction
2. during the auction any person may open an account associated with the certificate's name and transfer an arbitrary amount of means to the account
3. when there is enough means on the account, the account's owner may offer a new highest price of ownership and become the owner of the certificate
4. the price of ownership is being withdrawn from the owner's account once in several blocks, if there aren't means on the account the owner looses ownership
5. a person may bid if the sum of means on person's account and already withdrawn means is greater than *duration of the auction · new price of ownership*; after a bid her account is immediately withdrawn, so the amount of the withdrawn means is exactly equal to the product
6. after 3 months since the auction's start and 2 weeks since the last ownership change the auction is over and the current ownership is final
7. no auction participant receives means back

Because Nidaba wouldn't gain popularity immediately it is important to make he duration of auction larger at the beginning, for example one year instead of 3 months.

5 Time and correction of difficulty

Nidaba uses Bitcoin to fix a moment when a block was mined.

Any person can fix a moment but only the earliest would get a reward for it, so it is more likely that a miner would fix a block rigth after he mined it.

To fix a Nidaba block a person calculates its SHA256 hash code, uses the value as a hash of a public key, calculates the corresponding address in the Bitcoin network and transfers a small fraction of bitcoin to the address. A transfer of bitcoin to the Nidaba's block address in the Bitcoin network is called an *observation*. *Observations* are Bitcoin's transactions and all Bitcoin transactions are ordered therefore all observations are ordered and it is valid to use a relations "before" and "after".

When the person receives a conformation about of the *observation*, she wraps data about the observation (hash of a Bitcoin's block with the transaction, a path from Merkle tree root to the transaction and hash of the fixing Nidaba's block) into a *observation transaction* and broadcast it to the Nidaba network.

Let an *observation transaction* contains a hash of Nidaba's block X and a observation o. It is included into the Nidaba chain if the following conditions are met:

1. there are no more than 6 block after X
2. an observation transaction for X with observation before o wasn't included to the chain
3. the most earliest observation of any block before X is before o
4. the most earliest observation of any block after X is after o

When there are more than 6 blocks after X the block creation time is defined as the most earliest observation of the X; if no one provides a observation then the block creation time of X is equal to the creation time of the next block.

If the Nidaba block are being mined too often or too seldom then a difficulty correction occurs.

6 Forks

Miners work independently on adding a new block to the chain, if they add a block simultaneously the chain of blocks becomes a tree. In this case it is important to provide an algorithm to choose a main branch. The algorithm should have several properties.

A chosen branch should be the most hardest to fork.

A choice of the main branch should be independent from the other branches on the tree. It means if we take a tree and choose a main branch then for any sub

tree with the main branch the choice of a main branch must not change if apply the algorithm once again.

A choice of the main branch should not change if we add the same tail to the all branches (an independence from the future).

A choice of the main branch should not depend on the past, it means if we add a arbitrarily prefix before the fork occurred the choice of the main branch should not change. This property is not necessary but very convenient, because we can ignore the common prefix when we compare branches.

Nidaba uses the algorithm which meet the demands. Besides that the algorithm provides that the difficulty of the forking the chain behindhand (if a person decides to fork a chain on Tuesday and to fork all operations since Monday) is close to the difficulty of forking the Bitcoin chain.

Lets consider a fork.

$$\begin{array}{ccccc} & & & & B_4^1 \leftarrow B_5^1 \\ & & & \swarrow & \\ B_1 & \leftarrow & B_2 & \leftarrow & B_3 \\ & & & \nwarrow & \\ & & & & B_4^2 \leftarrow B_5^2 \end{array}$$

A person should calculate a score for each branch, the branch with a highest score is the main. Let's we have a set of branches $\{B_1 \leftarrow B_i^j\}$ and we want to find out a main branch. At first we calculate a maximum of block creation time $T = \max\{\mathrm{t}(B_i^j)\}$ then we calculate a score and choose a branch with the maximum.

$$\text{score of } B_1 \leftarrow B_n = \mathrm{f}(B_1 \leftarrow B_n, T) = \sum_{i=1}^{n} \mathrm{v}(B_i) \cdot q^{T-\mathrm{t}(B_i)}$$

Where $\mathrm{v}(B_i)$ is a sum of reward of block B_i and all means transferred with it; q is a fixed constant.

6.1 Correctness

Nominally a score of a particularly branch depends on the other branches by a way of the T parameter, so we may suppose that adding a branch with a low score may affect the choosing of the main one. Lets show that is wrong.

Lemma 6.1. *Adding of a new branch to the tree affects the choosing of the main branch if and only if the added branch has maximum score.*

Proof. Suppose we added a new branch $B_1 \leftarrow X$ to the set and recalculated scores. If the score of the new branch is maximum, when the new branch is the main. Otherwise we have two cases: $\mathrm{t}(X) \leq T$ and $\mathrm{t}(X) > T$.

With the first case the new value of T matches the old one, it leads that the new values of score matches the old ones and the choise of main branch does't change.

With the second case lets mark the new value of T as T' and explore how the change of T affects the values of score:

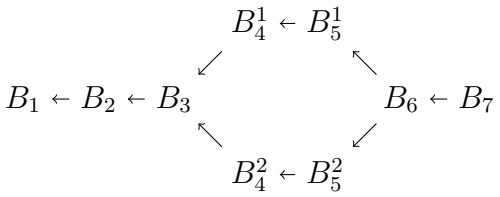
$$\begin{aligned}
\mathrm{f}(B_1 \leftarrow B_n, T') &= \sum_{i=1}^{n} \mathrm{v}(B_i) \cdot q^{T' - \mathrm{t}(B_i)} \\
&= \sum_{i=1}^{n} \mathrm{v}(B_i) \cdot q^{(T'-T)+T-\mathrm{t}(B_i)} \\
&= q^{T'-T} \sum_{i=1}^{n} \mathrm{v}(B_i) \cdot q^{T-\mathrm{t}(B_i)} \\
&= q^{T'-T} \mathrm{f}(B_1 \leftarrow B_n, T)
\end{aligned}$$

It turns out that with increasing of T a score for every branch is being multiplied by the same coefficient. Therefore, the branch with a maximum score does't change and our assumption is true.

More over this lemma implies a property of independence from the other branches. □

Lets demonstrate that the other properties are also held.

Lemma 6.2. *The algorithm of choosing the main branch is independent from the past and future*



Proof. Lets examine the difference between the branches.

$$\begin{aligned}
&\mathrm{f}(B_1 \leftarrow B_4^1 \leftarrow B_7, T) - \mathrm{f}(B_1 \leftarrow B_4^2 \leftarrow B_7, T) = \\
&\left(\sum_{i=1}^{3} \mathrm{v}(B_i) \cdot q^{T-\mathrm{t}(B_i)} + \sum_{i=4}^{5} \mathrm{v}(B_i^1) \cdot q^{T-\mathrm{t}(B_i^1)} + \sum_{i=6}^{7} \mathrm{v}(B_i) \cdot q^{T-\mathrm{t}(B_i)}\right) - \\
&\left(\sum_{i=1}^{3} \mathrm{v}(B_i) \cdot q^{T-\mathrm{t}(B_i)} + \sum_{i=4}^{5} \mathrm{v}(B_i^2) \cdot q^{T-\mathrm{t}(B_i^2)} + \sum_{i=6}^{7} \mathrm{v}(B_i) \cdot q^{T-\mathrm{t}(B_i)}\right) = \\
&\qquad = \sum_{i=4}^{5} \mathrm{v}(B_i^1) \cdot q^{T-\mathrm{t}(B_i^1)} - \sum_{i=4}^{5} \mathrm{v}(B_i^2) \cdot q^{T-\mathrm{t}(B_i^2)}
\end{aligned}$$

It depends neither on the tail ($B_i|_{i=6,7}$) nor on the prefix ($B_i|_{i=1,2,3}$) therefore the properties of independence are held. □

7 Certificate management

Certificate management includes raising bids on a auction, changing the public key and disavowing from the certificate ownership.

By default an authentication is done via the private key of the certificate and a person is authorized for all operations with the certificate. Such approach is vulnerable, if a malefactor steals the private key then the only one left to the owner is to issue a disavowal from certificate's ownership and to start a new auction for the same name (or to reattach to the current auction if it hasn't finished).

The default authentication may be changed. Nidaba allow to set up an another pair of a public and private keys to access to the certificate operations. If a malefactor gets a certificate's private key the owner would change the certificate's public key (a disavowal of a public key). If a malefactor gets the private key to control operations, the owner would disavow from certificate's ownership and start a new auction for the certificate's name.

The authentication isn't limited to those schemes. Nidaba uses a simple language to describe an authentication and authorization. For example, it allows to describe the following schemes.

Schema A. An owner of a certificate's account can allow anyone to bid for her sake in a auction if the last bid was done for sake of another account. Also an

owner can limit a bid to be no more than 20% above the last bid and below $5x$ of base price.

Schema B. An owner of a certificate can introduce a list of public keys and provide an access to certificate management operations only if there are quorum of keys. Also the list itself may be altered if the quorum are reached. Beside this a disavowal of the certificate ownership can be setup to be valid if it is signed by the quorum of current list or by quorum of list which was actual no more than N block ago. This scheme fits the companies:

1. all power isn't consolidated in one hands
2. if there is a turnover the control over certificate remains inside the company
3. if there is an unexpected change of list of keys the formal owners can issue an disavowal of ownership
4. a compromise of former employees private keys can't compromise the company

8 Scalability

Solutions based on a Bitcoin protocol have a scalability issue because each miner has to store locally all the network information (the blockchain). Nidaba solves this issue, but before we describe a solution we need to realize why the blockchain grows.

First of all, chain's size depends on the count of certificates it stores. It is impossible to predict the grow of registrations, so it is impossible to predict the grow of chain's size. However, if we deny new registration then the size of the chain still continues to grow due to a prologation of already registered certificates, since each certificate should be prolonged regularly then the grow of the chain is linear.

Nidaba's solution to the scalability issue is based on a assumption that if a chain holds only a fixed number of certificate then its linear grow is compensated by an annual grow of average storage capacity. So a Nidaba's chain has a limit of certificates it can hold, when a number of registered certificates approaches the limit a two new independent chains are introduces.

New certificates are registered in the new chains. If a certificate owner forgets to prolong a certificate in the old chain then a new certificate can be registered on it.

If chains are independent then two certificates with the same name may be registered in the different chains. The conflict is resolved by a convention. Among certificates with the same name on the different chains the one whose auction started earlier is believed to be valid and the rest are invalid and should be ignored. We can use the order relation on blocks of different Nidaba chains because a block creation time is a Bitcoin transaction and a order relation are defined on them.

9 Shares

Nidaba uses shares as a mechanism to add an incentive for some people (stockholders) to develop, maintain, advertise the network and to create a satellite projects. Shares are very similar to bitcoins: they can be spitted, merged and transferred (sold) to an another person, but a share allows a stockholder to receive dividend in a form of reward. Each block an additional reward equals to the 10% of the block's reward is being transferred to stockholders proportional to the number of shares they hold.

When a new chains are introduced they inherit a the same distribution of shares as the parent chain.

An initial distribution of the shares may be done via an external IPO, via an internal IPO, by a committee as reward to core developers or via mining as a additional form of reward.

The external IPO is an auction in which the developers of the network sell shares to the public just after the network has been started. This strategy allows any one to become a stockholder and enriches the developers of the network but after the network is developed.

The internal IPO is an auction that is very similar to the auction for a certificate's name. With the internal IPO a user should buy means and bids them in a auction. It allows any one to become a stockholder, but enriches the first miners of the network.

A committee of honest and well-known members of community publicly distributes shares between core developers as a reward and makes bounty for new features. It allows anyone who influences the network to become a stockholder, it add an incentive for developers to work, but enriches them after network is ready and proportional to their efforts.

If the shares are distributed via mining it allow any miner to become a stockholder and enriches the first miners.

10 Name resolving

There are different strategies for name resolving. Each of them is a trade off between security, required storage capacity and latency of resolving. The strategies aren't unique for Nidaba and can be used over Namecoin, Twister and other systems.

10.1 Full copy

A user stores locally all blockchains. This strategy provides maximum security and lowest latency, but requires a lot of stores capacity and the initial initialization takes a lot of time.

10.2 Partial copy

A user downloads all blockchains but doesn't store public keys. When the user needs to resolve a name she knows in which block it should be, so after a request to third party she can check whether a response is correct.

This strategy requires less storage capacity, provides the same level of security and the same time of initialization, but increases the latency of resolving.

10.3 Headers only

A user downloads and stores only headers. When the user makes a request to third party it receives a certificate and its coordination in a Merkle tree, so the user can check that the returned certificate is in a blockchain and hasn't expired yet.

The strategy is less secure because a malefactor may register a certificate with the same name on a parallel chain and return it or she may return a certificate without an information that later in a blockchain it was compromised (a owner issued a disavowal of ownership).

With this strategy a third party must sign its responses so the user could prove that a third party is a malefactor, moreover a user could do a lot of requests to different third parties to increase a security.

This method requires a far less storage capacity, the initialization is done faster, the latency is the same with the partial copy, but a partial trust to third party is needed.

10.4 Full trust with check

A user doesn't store any Nidaba data and asks a third party to resolve a name. A response must be signed to provide to user a chance to prove that a third party is a malefactor. An advantage over DNSSEC is that there are many of third parties so the user has an alternative if one of the parties is compromised. Moreover the user can do several requests to some of them to decreases a risk of disinformation.

11 Conclusion

We have described a distributed and resistant to cyber squatters PKI. This description contains several technics that can be used with another technologies based on a Bitcoin protocol or with altcoins. Among them are

1. a technic based a demand and supply principle to provide a stable price per reward on an expernal market
2. a way of adjusting internal price per some operation to a increasing performance of an average miner
3. an idea to use Bitcoin to determinate a moment when block was mined
4. a method to increase resistance to behindhand forking to a Bitcoin level
5. a solution to the scalability issue of the Bitcoin protocol based technologies

However this paper describes rather an idea of a distributed PKI than a particularly implementation and all constants mentioned above is choosen to look reasonable. If an implementation follows this article the constants should be checked on a appropriate model.

Attesting

This paper is attested by Bitcoin. It means that anyone can check when it was created. To do it a person should calculate a Bitcoin address based on the sha256 hash code of the paper and check when a transfer to the address was done.

To find out the address you should use the following command

```
sha256sum -b nidaba.pdf | awk '{print $1}' | python
   hashToAddress.py
```

The sources of hashToAddress.py are listed on the next page

```
import hashlib

def digest(method, hash):
  h = hashlib.new(method, hash.decode("hex"))
  return h.hexdigest()

ripemd160 = lambda x: digest("ripemd160", x)
sha256 = lambda x: digest("sha256", x)

__b58chars = "123456789" + \
  "ABCDEFGHJKLMNPQRSTUVWXYZ" + \
  "abcdefghijkmnopqrstuvwxyz"
__b58base = len(__b58chars)

def b58(hash):
  data = hash.decode("hex")
  long_value = 0
  for (i, c) in enumerate(data[::-1]):
    long_value += (256**i) * ord(c)
  result = ""
  while long_value >= __b58base:
    div, mod = divmod(long_value, __b58base)
    result = __b58chars[mod] + result
    long_value = div
  result = __b58chars[long_value] + result
  nPad = 0
  for x in data:
    if x != '\0': break
    nPad += 1
  return (__b58chars[0]*nPad) + result

footprint = raw_input()
footprint = "00" + ripemd160(footprint)
footprint = footprint + sha256(sha256(footprint))[:8]
print b58(footprint)
```

License

References

[1] Satoshi Nakamoto, *"Bitcoin: A Peer-to-Peer Electronic Cash System"*. 2009.

[2] Miguel Freitas, *"twister - a P2P microblogging platform"*. 2013.