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

Denis Rystsov
rystsov.denis@gmail.com
nidaba-pki.org

3 May 2014 (attested by Bitcoin)

Abstract

A distributed PKI would solve the problem of trusting the authorities in the key technologies behind the Internet. Its 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 the 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 the key is a unique name (domain) and the value is a public key. An element of the hash table — a pair of a unique name and a public key — is called a certificate.

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

[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

Let's describe how one may 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 way. The 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, then issue a certificate using the openly known private key and use the generated certificate as a regular SSL certificate. If the site's visitor supports distributed CA, they recognize the marker and check whether the certificate's public key matches the value retrieved from Nidaba by the site's canonical URL. If the visitor doesn't support distributed CA, they would be warned that the site uses an untrusted certificate authority.

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 the public key stored in Nidaba with the domain as a unique name; and to distribute the signed address in a Kadmilia-like network.

Nidaba can also be easily used as a distributed OpenID provider: if a person uses the name of their public key in Nidaba as the username, then, as a proof of their identity, a site might ask the user to sign a provided message with their 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 of them is compromised, disastrous effects may follow. Therefore, they must be secure and reliable. If the technology is centralized, there are some risks that can't be eliminated by technological means.

The company behind the centralized solution may fail or change its strategy. For instance, if Facebook goes bankrupt, then all the sites that use Facebook as an OAuth provider are also affected, since the users of these sites are unable to prove their identity any longer.

Moreover, a government may force a company to compromise the data of its users. For example, if a government influences the ICANN and some 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 can be separated in two parts: "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 given moment of time there is a sufficient number of active nodes. If the work of the nodes is voluntary or is paid from donations, there is a risk of "tragedy of the commons": each participant of the network could think that her contribution to the greater good is negligible, and give up.

If the price is not controlled, there is a risk that it would be too high or too low. If it is too high, there will be no users. If it is too low, the activity of cyber squatters and vandals would increase. So there is a need for rules to hold the price in a fixed corridor.

1.3 Analogs

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

Namecoin[4] is a deflationary cryptocurrency similar to Bitcoin, which extends the Bitoin protocol with several domain operations. Each domain is unique and has an associated value. To register a new domain, one needs to pay the network and transaction fees with the intrinsic currency, NMC. The network fee was initially introduced to decrease the cyber squatters' activity, but now its value is too low to do it. The mining reward is exponentially decreasing, and when it reaches zero, the miners would start working for only the transaction fee. So there is a risk of the "tragedy of the commons" situation: a miner gains more profit right now if they include 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, where anyone can register a name free of charge. The users also keep their nodes active free of charge. Hashcash-like scheme[6] is used to protect from cyber squatters.

The Namecoin economy is not stable, and it may suffer from the tragedy of the commons. There are also no 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

[6]Hashcash - A proof-of-work system http://en.wikipedia.org/wiki/Hashcash

Twister has no economy, and may also suffer from the tragedy of the commons, since all the 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

Let's see how Nidaba provides price stability.

Miners get rewarded when they work for the benefit of the network. A miner can't transfer this reward to 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 users willing to register or prolong a certificate.

To register or to prolong a certificate it is necessary to destroy some quantity of reward. This quantity (the price) is constant on a small interval of time. The reader can find the rules that describe how this quantity is changed over time later in the article.

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

The base price is adjusted over time to reflect the increase of an average miner's performance, but at the start of the network it is chosen to be equal to the quantity that allows an average miner to prolong a certificate for one year, if they mine for one month.

A miner's reward per block consists of two parts. The main part is proportional to the current difficulty. Since the current difficulty is proportional to the current total network power, the expected value of a miner's profit depends solely on their own power. Thus a miner who joined the network at the beginning and a miner who did it when the network was mature, would gain an equal reward if they had the same power.

The second part of the reward per block is equal to 10% of the total reward 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 convenience, each certificate has an account assigned to it, from which the payment for the prolongation is automatically withdrawn. The owner may transfer an arbitrary amount of currency to prolong a certificate for some time ahead.

To avoid overproduction of reward, it has an expiration date, after which it can no longer be transferred to a certificate account. The expiration date makes the reward more similar to futures contracts than to gold. So, from here onwards, the reward is also referred to as *futures* or *means*. Besides the expiration date, there is an account fee to avoid overproduction: once in several blocks, a small percent of means (account fee) are withdrawn from the certificate account and destroyed. The 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 it more profitable to prolong the certificate for a short period time several times a year than one time for the whole year.

This scheme provides a permanent and uniform demand for reward.

The 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. Mining becomes unprofitable for some of the miners, and they leave the network. The total power decreases until the supply of futures goes down to the demand level. If the demand for futures rises, then the price per future goes up. It attracts new miners, and 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 the base price according to the increase of an average miner's performance.

There is a trick to figure out an average miner's performance: we encourage a registrar to do an arbitrary quantity of work when they register or prolong a certificate. It is possible to measure the average performance based on the submitted information. According to the increasing of the average performance, the base price must be increased. The price is adjusted if the average performance has changed more than twofold since the last adjustment.

To add an incentive for a registrar to do the addition work the system return back 50% of their payment for the certificate operation if they hit the top 10% productive registrars (measured by that work).

4 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 right after he mined it.

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

When a user receives the confirmation about of the *observation*, they wrap the 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 an *observation transaction*, and broadcast it to the Nidaba network.

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

1. There are no more than 6 blocks after X
2. No observation transaction for X with observation that is *before* o is included present in the chain
3. The earliest observation of any block before X is *before* o
4. The 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 earliest observation of X. If no one has provided an observation, then the block creation time of X is equal to the creation time of the next block.

Just like Bitcoin, Nidaba maintains a constant speed of block generation, if the Nidaba blocks are mined too often or too rarely, then a difficulty correction occurs to adjust the speed.

Later in the article we use an intrinsic time. The intrinsic time is measured by the Nidaba blocks that passed since some event. Also an intrinsic time can be refferenced by regular *weeks* or day, in that case we need to multiply a regular time interval by the Nidaba' block generation rate. For instance, Bitcoin's int. day is equal to 144 blocks, since the Bitcoin's block generation rate is 6 blocks per hour.

5 Forks

Miners work independently on adding new blocks 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 the main branch. The algorithm should have several properties.

1. The chosen branch should be the hardest to fork
2. The choice of the main branch should be independent from the other branches on the tree. It means that if we take a tree and choose a main branch, then for any subtree within that main branch, the choice of the main branch must not change if the algorithm is run once again
3. The choice of the main branch should not change if we add the same tail to the all branches (independence from the future)
4. The choice of the main branch should not depend on the past. It means that if we add an arbitrarily prefix before the fork occurred, the choice of the main branch should not change. This property is not necessary, but is very convenient, since we can ignore the common prefix when we compare branches

Nidaba uses an algorithm which meets the demands. Other than that, the algorithm also guarantees that the difficulty of forking the chain behindhand (e.g. if a user decides to fork a chain on Tuesday, and to fork all the operations since Monday) is close to the difficulty of forking a Bitcoin chain.

Let us consider a fork.

$$\begin{tikzcd}
 & & & B_4^1 \arrow[dl] & B_5^1 \arrow[l] \\
B_1 & B_2 \arrow[l] & B_3 \arrow[l] & & \\
 & & & B_4^2 \arrow[ul] & B_5^2 \arrow[l]
\end{tikzcd}$$

The user should calculate the score for each branch, and the branch with the highest score is chosen to be the main one. Suppose that we have a set of branches $\{B_1 \leftarrow B_i^j\}$ and we want to determine the main branch. First, we calculate the maximum of block creation time $T = \max\{\mathrm{t}(B_i^j)\}$, then we calculate the score and choose the branch that has the maximum value.

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

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

5.1 Correctness

Nominally a score of a particular branch depends on the other branches via the T parameter. Therefore, we may suppose that adding a branch with a low score may affect the choice of the main one. Let us show that this is not so.

Lemma 5.1. *Adding of a new branch to the tree affects the choice of the main branch if and only if the added branch has the highest score.*

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

In the first case, the new value of T matches the old one, from which it follows that the new values of score match the old ones, and the choice of main branch doesn't change.

For the second case, let's 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, the score for every branch is multiplied by the same coefficient. Therefore, the branch with the maximum score doesn't change, and the lemma is true.

Moreover, this lemma implies the property of independence from other branches. □

Let us demonstrate that the other properties are also held.

Lemma 5.2. *The algorithm of choosing the main branch is independent from the past and the future*

$$\begin{tikzcd}
 & B_4^1 \arrow[dl] & B_5^1 \arrow[l] & \\
B_1 & B_2 \arrow[l] & B_3 \arrow[l] & & B_6 \arrow[ul] \arrow[dl] & B_7 \arrow[l] \\
 & B_4^2 \arrow[ul] & B_5^2 \arrow[l] &
\end{tikzcd}$$

Proof. Let us examine the difference between the branches.

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

The difference 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. □

6 Cyber squatting

It is impossible to avoid disputes about the names. In a centralized system, the company behind the system may be the 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 a user, the user becomes the owner of the certificate. The price of the ownership matches the base price but the certificate's name is automatically put up for auction

2. During the auction, any user may open an account associated with the certificate's name and transfer an arbitrary quantity 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 withdrawn from the owner's account once in several blocks. If there aren't means on the account, then the owner loses their ownership

5. A user may bid if the means on that user's account plus the already withdrawn means is greater than *duration of the auction* · *new price of ownership*. After a bid their account is immediately charged, so the sum of all the withdrawn means is exactly equal to the product

6. After 3 int. months since the auction has started, and 2 int. weeks after the last ownership change, the auction is over and the current ownership is final

7. No auction participant receives their means back

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

7 Certificate management

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

By default, an authentication is done via the private key of the certificate. The user is authorized for all the operations with the certificate. Such an approach is vulnerable, since if a malicious party steals the private key, then the only option available to the owner is to issue a disavowal from the 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 public and private keys to access the certificate operations. If a malicious party gets a certificate's private key, the owner could change the certificate's public key (a disavowal of a public key). If the malicious party gets the private key to control operations, the owner would disavow the 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 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 their sake in a auction. The owner may limit a bid to be no more than 20% above the last bid to prevent a malefactor to make an astronomical bid to sabotage the auction. In addition the owner may deny to make a bid for their sake if they is already the owner of the certificate to avoid the malefactor doing several sequential bids to push the price of ownership higher. Also the owner may limit the bid for their sake to be no more that $5x$ of base price.

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

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

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

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

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 block creation time is a Bitcoin transaction and an order is 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 satellite projects. Shares are very similar to bitcoins: they can be split, merged and transferred (sold) to an another person. But a share allows the stockholder to receive dividends in the form of reward. Each block has an additional reward that equals to 10% of the block's reward and is transferred to stockholders in proportion with the number of shares they hold.

When new chains are introduced, they inherit 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 an 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 anyone to become a stockholder and enriches the developers of the network 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 anyone 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 and adds an incentive for developers to work, but enriches them after network is ready in proportion with their efforts.

If the shares are distributed via mining it allows 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 all blockchains locally. This strategy provides maximum security and lowest latency, but requires a lot of storage capacity. The initial initialization also 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, they know in which block it should be, so after a request to third party they can check whether the 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. They may also return a certificate, but without the information that later in a blockchain it was compromised (the 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 security.

This method requires far less storage capacity, the initialization is done faster, the latency is the same with the partial copy, but some degree of 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 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 the risk of disinformation.

11 Conclusion

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

1. A technique based on the demand and supply principle to provide a stable price per reward on an external market
2. A way of adjusting internal price per some operation to the increasing performance of an average miner
3. An idea to use Bitcoin to determinate the moment when a 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 particular implementation. All constants mentioned above are chosen to look reasonable. If an implementation follows this article, the constants should be checked on an 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.