Doichain

(working draft)

The Atomic “Double-Opt-In“ and email spam protection system on the blockchain

Version 0.0.7.1

13st of November 2018

Nico Krause
(nico@le-space.de)

www.doichain.org

Abstract:

We propose a decentralized peer-to-peer blockchain system that allows entities to securely request and store a proven permission to send an email to another entity. We call this permission "atomic Double-Opt-In".

DOICHAIN

Contents

1. Target Audience

This primarily technical document is mainly intended to be understood by blockchain developers. Several chapters are also readable for a non-technical audience. E.g.

2) Summary 3) Background & Motivation and others.

2. Summary

1) Problem and challenge

The medium email works perfectly fine in principle, as long as we are talking about private emails, which get sent to a single person or a small group. However emails are being used for marketing- and selling campaigns since the 1990s. These campaigns get more successful, the more people receive your emails. This creates a large temptation to send as much emails as possible, even if you can't prove without a doubt, whether the address owner did agree to receive this email or not. The circumstance and the relatively cheap price to send emails, encouraged a flood of huge numbers of emails. Therefore a law was developed, to only allow mass sendings of emails if the receiver gives permission to do so. This permission is usually granted with a Double-Opt-In procedure, in other words, a granted permission for advertising needs to be confirmed additionally. However, the problem is, that the independent third party, such as email providers or private smtp servers, can't verify this permission by newest data protection regulations. Both provider of the mail server and the receiver of the advertisement need to trust them, to manage the declarations of consent and to reliably document a removal of an agreement.

SPAM is a consequence of violating this rule, and is perceived as an unwanted interference by most people. Thus different methods got developed to fend off SPAM. But to this day no system is reliable and practical enough to differentiate between wanted and unwanted emails. Sometimes it occurs, that desired emails land in the SPAM directory or doesn't arrive at all. On the other hand SPAM reaches the inbox of the receiver. Thanks to Doichain this problem can be solved reliably, while taking into account all data security regulations.

2) Affected parties

1. **Owner of an email address**
   Goal: Declarations of consent can be managed and checked all the time. Withdrawing an agreement needs to be documented and executed reliably.

   Risk so far: Unsubscribing a newsletter leads you to getting tagged as active, thus being a valuable email address to sell.

   Consequence so far: Even more email spam.

2. **Email service provider**
   Goal: Smooth delivery of emails.

Risk so far: Damage to reputation because of address lists, that are filled with SPAM-traps.

3. **Advertiser**
   Goal: Proper DOI-confirmed declarations of consent in order to send advertising mails.

   Risk so far: The lead-generator delivers missing or incorrect DOI-declarations of consent.

   Consequence so far: SPAM-traps are delivered into the database, which does significantly damage the reputation and complicates email marketing or even renders it impossible.

4. **Mail Server provider and internet service provider who let customers freely use their mailbox.**
   Goal: Ensuring that valid emails are delivered into the inbox, while spam mails are sorted in the spam directory or blocked completely, all in the interest to satisfy their customers.

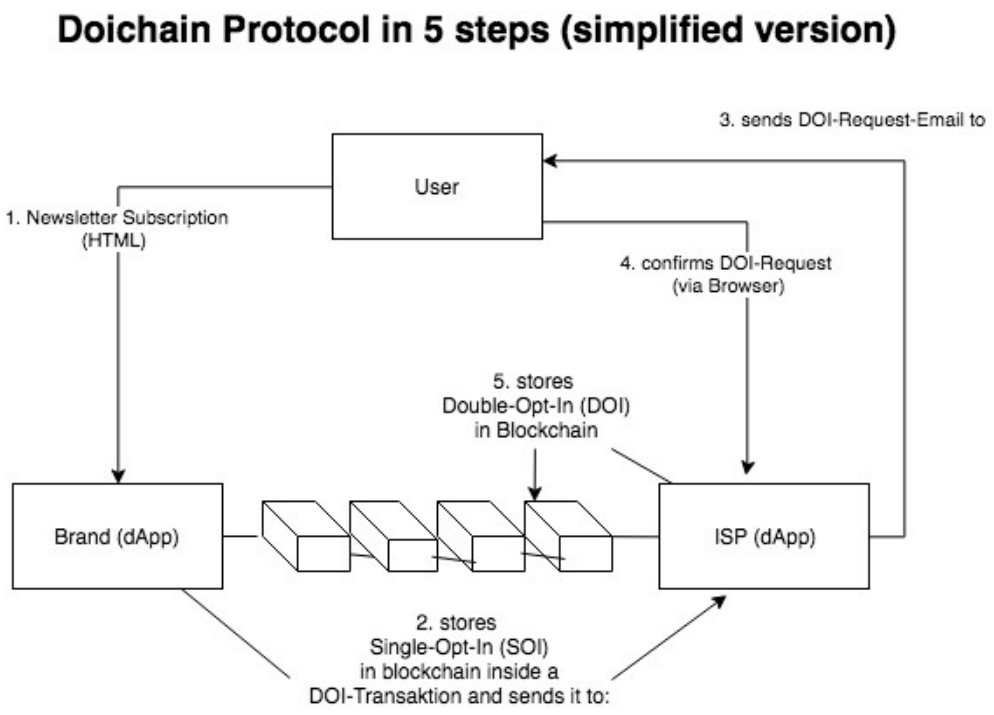
   Risk so far: Desired and awaited emails from customers are blocked and not delivered.

5. **Address generators, firms that specialise into campaigns like lotteries, in order to get agreements and addresses for their sponsors, who will use them in email marketing campaigns.**
   Goal: Generate the highest number of permissions possible with a relatively low amount of capital, for sponsors to use in their advertising campaigns.

6. **Courts**
   Goal: Definite prove, if a permission was granted for the delivery of this email at an exact point in time.

3) Solutions with the aid of a blockchain

1. The owner of an email address registers himself in an online-form
2. The Address generators send the email address(es) to a decentralized app (dApp). As Single-Opt-Ins the signed entry (SOI) takes place in the Doichain.
3. The receiving node's dApp sends and email to the receiver and prompts him to confirm the asserted agreement with a click on a link.
4. The email receiver clicks on the DOI-confirmation link.
5. The dApp now confirms the Single-Opt-In (SOI) in the blockchain with another signature. As a result the SOI becomes a DOI. In other words it becomes a valid agreement, confirmed by the email address owner.



DOICHAIN

3. Background and motivation

The initiators of Doichain are coming from a classical email marketing background. As widely known, email marketing industry struggles a lot with the quality of email addresses and the whole process of sending emails in conjunction with spam. The main motivation was to create an atomic prove for high quality mailings, by utilizing blockchain technology for requesting, confirming, storing and verifying an email permission. In a further step this system helps to identify spam and non-spam.

4. Doichain dApp

The dApp works as an interface for the blockchain, that is readable and writable by both humans and machines. This paragraph is intended for technical readers and blockchain developers.

1) The machine2machine interface:

A REST-API for three different modes

1) **Send-Mode:** Is used by parties which store and send SOIs (Single-Opt-In), request unconfirmed and export confirmed DOIs (Double-Opt-In). (For simplicity we call this party Alice)
   1. **URL /opt-in (POST, authentication required) (Parameter: sender email, recipient email, auth token, userId):**
      Is usually called by an application server of a website project, which transfers the senders and recipients email address to the Doichain dApp. It stores the SOI on the Doichain and requests the DOI from the corresponding public key Doichain address, which was queried from the recipients email DNS.

      The SOI also contains an encrypted HTTP(s) URL of the local dApp (Alice). So the confirming dApp (Bob) can request the email template to be sent to the final recipient (Peter). It is also used in order to inform the requesting (sending) dApp (Alice) about a successful DOI confirmation.

      Doichain extends the Namecoin blockchain with a new name_op. The new "name_doi" transaction gets transferred to the recipient trusted dApp with an enabled "confirm-mode".

      Internally it creates a signature over a string, containing the sender and recipients email address, using the wallet owners private key.

   2. **URL /opt-in (PUT) (Parameter: nameId of DOI, signature of DOI):**
      Is called by the confirm dApp (Bob), when a DOI gets confirmed successfully by a internet user (Peter), after receiving the DOI request email. It's simply necessary to instantly offchain-inform the requesting party (Alice) about a successfully given DOI. The confirmation is saved in the local database of the DOI requesting party (Alice)

   3. **URL /doi-mail (GET) (Parameter: nameId of DOI, signature DOI):**
      Is called by the dApp in confirm mode (Bob), after receiving the unconfirmed SOI transaction.

      This URL returns the email template, which will be sent over SMTP to Peter. Furthermore it contains other important data e.g. redirect-URL and email subject.

      In later improved releases this URL might also return an unsigned raw-transaction hash to be signed by Bob.

   4. **URL /export (GET, authentication required):**
      Is only called by the application server of the DOI requesting party and simply exports all successfully confirmed DOI's in JSON format. In the future this interface will undergo some significant extensions regarding filtering possibilities.

2) **Confirm-Mode:** Is used by the party, which receives unconfirmed SOI transactions (Bob/Peter) and transfers confirmed DOI's back to Alice
   1. **URL /opt-in/confirm/:hash (GET)**
      Is usually called by the browser of the internet user, who owns the recipient address (Peter), after he receives the DOI request email. The email was sent by the SMTP Server, which runs a Doichain dApp in confirm mode.
      The hash given by the URL gets decoded to an ASCII database id. This REST call signs and stores the confirmation of the DOI in the local database. It executes the name_op "name_doi", so the DOI becomes officially valid after 6 confirmations.

   2. **URL /walletNotify (GET) (Parameter: tx – the transaction id of an incoming SOI)**
      Is called by a script of Bob's Doichain node (walletnotify), as soon as a transaction arrives. Alice read Bob´s public key (Doichain address) from a DNS TXT value. (see Chapter: DNS-queries and attributes)

      The transaction is checked, if it contains a suitable "name_doi". The nameId and value gets extracted from the Doichain and saved in the local database. Furthermore the URL /doi-mail is called on Alice´s dApp (which sent out the SOI). It produces an email template for the DOI-request email that is sent to Peter.

3) **Verify-Mode:** Can be used by all involved parties to verify and validate a given DOI permission. In the future it can be used by a SMTP filter, to validate an incoming email containing special DOI-SMTP-headers.
   1. **URL /verify (GET) (Parameter: name_id, signature, sender_email, recipient_email)**
      Reads the DOI-entry from the Doichain via the name_id (given in the SMTP-DOI-header). It validates two signatures:
      1. The SOI-signature gets validated via public key given in the SMTP-DOI-Header using a string built from sender_email and recipient_email.
      2. The DOI-signature gets validated via public key given by the DNS TXT attribute using a string built from the first signature.

2) The human interface

1. Account Balance (Doi)
2. Query NameId's and DOI's
3. Wallet Transactions
4. Wallet Accounts & Addresses
5. Walletfunctions (e.g. sendToAddress, MultiSig)
6. Verify DOI's with sender and recipient address, public key and nameId
7. Doi Requests and Confirmations (inkl. Recipient and Sender)

5. Doichain Node

Is a Namecoin (Bitcoin) based software fork with new genesis block and other modifications. The Dochain team added a new rpc-command **“name_doi”** which takes two to three arguments:

name_doi "name" "value" ("toaddress")

name_doi registers the given name and value and transfers it to an optionally to a Doichain address:

Arguments:

1. "name"	(string, required) the name under which the DOI-transaction-data will be registered. A name for the doichain project should always use the namespace “e/”followed by a 256-bit, ECDSA valid, number represanted as a 32 byte (64 characters) string (Same as every Bitcoin privateKey). See also: https://en.bitcoin.it/wiki/Private_key
2. "value"	(string, required) value for the name - containing all necessary information of a valid soi/doi
3. "toaddress"	(string, optional) address to send the doi to

Result:

"txid" (string) the name_doi's txid

Examples:

doichain-cli name_doi "myname", "new-value"

doichain-cli name_doi "myname", "new-value", "NEX4nME5p3iyNK3gFh4FUeU..."

6. The Atomic Double-Opt-In

A Double-Opt-In (DOI) is a two step procedure required by European Data Protection Law when a website owner collects email addresses from website users in order to send them emails later on. Typically a website interacts with an internet user and offers a subscription to an email newsletter or other information in one or the other way. The internet user has to actively opt-in his wish on this website to be legal.

This opt-in is the first step and is also called the Single-Opt-In (SOI). As the user submits the newsletter subscription form, the application server of the website sends out a cofirmation email. The user confirms it typically by clicking a HTTP-link. The so created Double-Opt-In (DOI) can be legally registered on the websites newsletter database.

Interestingly, this process in practice is completely insecure and companies in this field are acting completely on trust when working with contractors or suppliers which acquire and sell DOI's.

Further on, legally acquired DOI's tend to be often forgotten by the entities itself which once in fact gave them. In that case other annoyences are inevitable. All in all, the whole procedure was so far neither trustable, nor reliable or satisfying for all players involved.

Doichain steps in here and for a first quick overview and easier understanding we assume a simple scenario:

1) Scenario with three actors: Alice (website owner), Peter (internet user), Bob (mail provider).

Alice owns a website with connected Doichain node and Doichain dApp. She wants to send a newsletter to Peter. We further assume the email of Peter is hosted with Bob's mail-server with connected Doichain Node and dApp.

1. At First, Peter submits his email address on Alice's newsletter form while the application server of Alice stores Peters email address locally. The email address of the sender (Alice) and the recipient (Peter) is posted to the local Doichain dApp by via REST-API.
2. The Doichain dApp, generates a unique Hash-key, which we call historically correct NameId. (Doichain is a fork of Namecoin blockchain). In Doichain we use the namespace “e/” for “email” (instead of d/ or id/ in Namecoin). The NameId references SOI and later DOI on the blockchain with Peters email address in the local database. In the current dApp prototype version, we generate and locally store a interims private- and public key pair for Peter, since we assume he doesn't have a Doichain wallet nor Doichain node yet.

   We create a signature over a text string using Peter's interim private key with the email address of sender and recipient (here Alice's and Bob's).

   Now, we execute the name_doi Doichain RPC command on the blockchain which initially stores the Single-Opt-In (SOI) inside the blockchain. It will be able to be found by the NameID and the signature can be verified by public key of Peter.

Bob's mail-server also has a connected Doichain node (and public-key), which we published via a DNS TXT record. We send the name_op transaction NameID which holds the SOI, to an address of this public key. As we would send money on the Bitcoin network, the transaction gets broadcasted all over the connected Doichain nodes in unconfirmed way.

3. Bob's Doichain Node informs his dApp via walletnotfiy, which then sends out an email via SMTP into Peters email inbox. The email template to be used was gathered via REST from Alice dApp straight over HTTP(S)). Peter receives the DOI-request email, confirms it with a HTTP-confirm link pointing to Bobs dApp, so the DOI cannot be faked. The dApp will execute a second name_doi command on the Doichain by signing the DOI additionally with the private key of Bob's mail-server connected Doichain node.

4. Another day, Alice decides to send out her newsletter and attaches two additional smtp header to her email: X-Doichain_NameId and X-Doichain-Public-Key of Bob's DOI reference hash in the Doichain and Bob's public key.

5. Bob's SMTP Server has an installed smtp filter to lookup Doichain NameId's. It receives the email from Alice and filters out the X-Doichain_NameId and X-Doichain-Public-Key SMTP-headers and verifies it by contacting the local Doichain dApp via REST URL /verify. If the signature of the DOI is valid and corresponds with the senders and recipients email address, the email is eventually delivered to Bob's Inbox.

2) Co-Registration Scenario

Alice is a Email-Marketing-Company and runs a landing page for a lottery game. The lottery is sponsored by several co-sponsors which expect a separate permission (DOI) to email the participants of the lottery.

The workflow in this scenario is almost the same as in the "Simple Scenario" except the following:

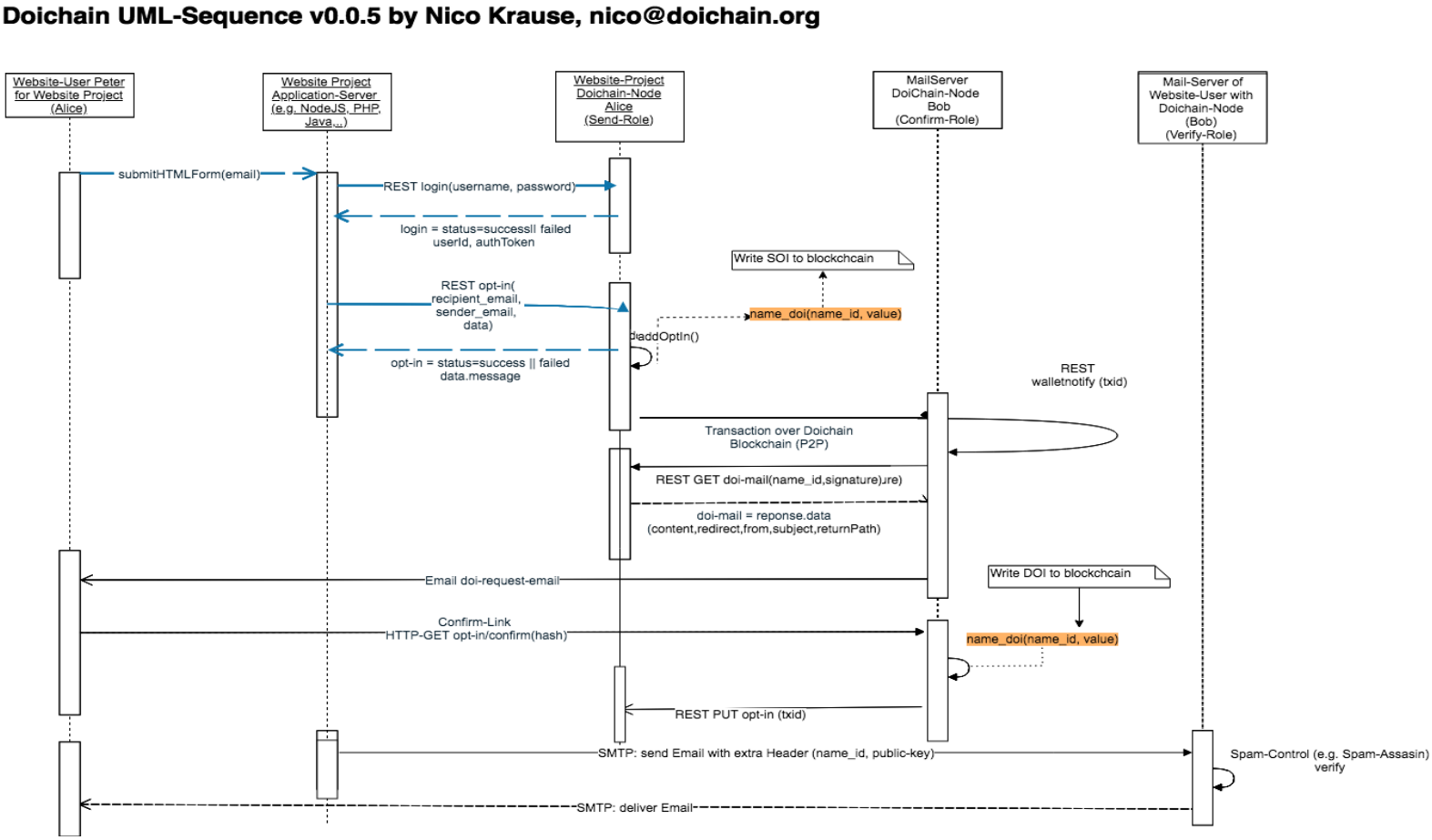
1. REST-API is able to take multiple ***sender_mail*** parameters
2. A separate DOI is requested and broadcasted for each ***sender_email*** and needs to be separately funded by Alice. The same nameId is used for all sender_emails. The first sender_email servers as main sponsor and is called "masterDoi". All following sender_emails are regarded as "Co-sponsors" every co-sponser can retrieve his DOI from the blockchain with an index attached at the end of a nameId e.g.

   masterDOI: e/445D042DDD7F8AF956FE309CDAFD10EA7535C344B672E0980E78BF5D12A90AD1

   DOI[1]: e/445D042DDD7F8AF956FE309CDAFD10EA7535C344B672E0980E78BF5D12A90AD1-**1**
   DOI[2]: e/445D042DDD7F8AF956FE309CDAFD10EA7535C344B672E0980E78BF5D12A90AD1-**2**
   DOI[3]: e/445D042DDD7F8AF956FE309CDAFD10EA7535C344B672E0980E78BF5D12A90AD1-3

3. Bob recognises the incoming name_doi transactions as co-registrations and sends out only 1 email for all of the co-sponsors.
4. Peters confirms the DOI-request-email as usual. All DOIs for all co-sponsors get confirmed.

3) Sequence Diagram



7. Further Technical Details

1) DNS-queries and attributes

A Doichain - dApp needs to find out the destination dApp of a SOI (unconfirmed DOI) transaction for sending out the Doi-request email to confirm. In order to do so we query a TXT attribute of the DNS of the recipients email domain:

doichain-opt-in-key=<Doichain Public-Key>
doichain-opt-in-provider=<Delegated Doichain Provider Domain>
doichain-testnet-opt-in-key=<Doichain Testnet Public-Key>
doichain-testnet-opt-in-provider=<Delegated Doichain Testnet Provider Domain>

The DNS-TXT attribute **doichain-opt-in-key** (Doichain Public-Key) is queried first from the DNS. If it does not exist **doichain-opt-in-provider** is read alternatively. It delegates a responsible DNS domain name which holds the responsible public-key for Doichain confirmations. In case neither a doichain-opt-in-key nor a doichain-opt-in-provider is found in the DNS of the recipients domain, the public-key of a so far centralized fallback server is used instead.

If developer are utilizing the Doichain Testnet a separate Testnet public-key can be added alternatively the the DNS:

doichain-testnet-opt-in-key
doichain-testnet-opt-in-provider

In a further version of Doichain it can or must be discussed if such a functionality should stay on centralized DNS infrastructure or if would be better to store those data also inside the Doichain

blockchain with a separate name value pair. E.g. with namespace ***"doi-d/{example-domain.com}"***

Aspects of centralization and decentralization utilizing a fallback server

Remarks:
Specifications to be added in further versions of this document:

decentralization of the fallback Doichain DOI confirmation node

public-key revocation and replacement / public-key domain - history stored doichain.