SELFII ID Platform

Yellow paper v0.97 - November 16, 2017

Overview

SELFII ID is a decentralised framework and utility token that provides identity and ownership features built on open standards for use in social networks and the world wide web.

Goals

1. **Simplify Identification:** Provide a one-click solution to securely sign into any service on any device; and allow users and services to verify identities with permission.
2. **Empower Users:** Allow users to control access to their personal information and decide how anonymous or permissive their identities are.
3. **Build Trust:** Eliminate spam by making it uneconomical and disincentivize abuse. Build confidence in the authenticity of user identities.
4. **Provide a Platform:** Allow third parties to implement an endless number of free and premium features for their services from ecommerce to digital rights to gig economies.

Implementation of SELFII ID's core features are provided by Ethereum smart contracts, providing a decentralised, trustless and open infrastructure.

Legal Disclaimer

This document is an introduction to the SELFII ID project (“SELFII ID”). The purpose of this document is to communicate the goals of Selfii International Limited (“SIL”), the current status of SELFII ID, as well as information about how SELFII ID relates to other projects operated by SIL.

The information within this document is provided “as is” without any representations or warranties (express or implied). SIL makes no representations or warranties in relation to the information within this document.

SIL is not bound by any representation of design, roadmap or expected performance of the application or related projects described herein. SIL reserves the right to alter designs and goals at their sole discretion.

Without prejudice SIL does not warrant that this document will be constantly available or available at all; or that the information in this document is complete, up to date, non-misleading, or accurate.

This document has not been reviewed or approved by a regulatory body. Based on the nature of the product, there is no requirement to do so. There are no actions active or pending against SIL under any regulatory agency in any jurisdiction.

Background

As the blockchain careers towards mass adoption for payments, ownership, e-commerce and decentralised services, the need for blockchain based identities is becoming increasingly inevitable for business-to-business, business-to-customer and user-to-user applications.

SELFII ID is an identity framework for everyday people for use in social media and the web that empowers users to own their identity, enforce their own privacy, build trust with other users, prevent spam and abuse, and perform transactions online.

SELFII ID was initially designed to solve real world social media problems for SIL's Selfii social network for trust, privacy and reputation and has evolved into a powerful framework for improving the user experience of the next iteration of the web as a whole.

The Selfii team are great believers of decentralised technology and its endless ability to empower individuals to make their own choices, own their own data, and prove ownership of assets, both virtual and real.

The team are invested in improving the user experience of blockchain enabled technologies. We envisage an improved web where one-click, passwordless sign-in and one-click purchases are the norm. A future where pseudo-anonymous strangers can be engaged with trust without impersonation, spam, abuse or fraud.

We are developing an identity framework for a real-world product to solve real-world problems. Eventually we will standardise and open the platform to all developers and service providers so that they may extend it with their own features for their own products.

This paper presents the SELFII ID identity framework and an initial set of fundamental features as well as its future potential as a widely adopted technology.

Contents

Initial Features

As a generalised framework SELFII ID offers many use cases, many of which are yet to be invented. However SELFII ID was designed with current projects and partners in mind to solve specific needs today. Services can benefit from one or all of SELFII ID's features depending on their needs.

SELFII ID is designed to be extendable to add new features and uses in the future that typically require an identity. Future features may include premium subscriptions, certifications, additional reputation attributes, ownership of virtual goods, granted access to events and services, and many more.

Blockchain technologies offer powerful benefits to both businesses and individuals. While businesses invest in implementation projects a few technical barriers still exist that keep the blockchain out of reach to the average individual.

SELFII ID offers individuals additional benefits to joining the crypto community as well as incentivized adoption provided by our integration partners; further encouraging growth of the new economy as a whole.

Identification

As every Ethereum user has at least one distinct account that is secured by them and the blockchain, consistent identification becomes both simple and safe. Using SELFII ID, individuals can sign into services with a single click to create a unique account with that service without needing a password.

Services then may ask for additional information as required. Once registered, a returning user can simply click to sign in again. Users no longer need to remember per-service usernames and

passwords nor do they have to complete multi step authentication (often across multiple devices) or struggle with increasingly frustrating human verification processes.

Services benefit from the additional security and increased conversions. They no longer need to retain and secure authentication credentials, greatly reducing the risk of account hacks; and they no longer lose users to multi step registration and human verification.

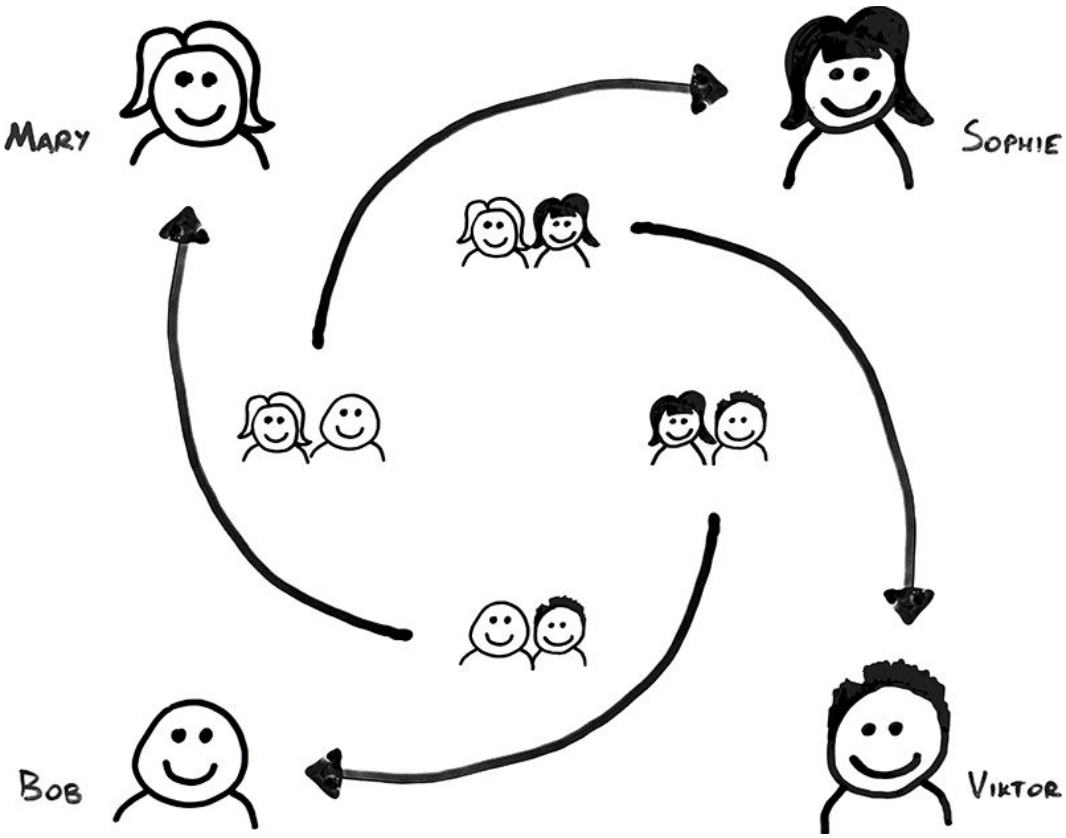
FaceScore

A pervasive problem in social media and meetup apps is "catfishing". This is when a disingenuous user builds or steals an identity using images of another person. Users may form relationships but are never quite sure they are talking to the person that is presented. Too often it turns out that the user is not who they presented. Users really want to trust that other users are really the person that they show in their pictures.

SELFII ID's FaceScore feature builds a confidence score that a user really is the person that they present in pictures by employing a web-of-trust based on users meeting each other and taking pictures together. FaceScore is automatic for users that opt-in and user adoption is not limited by the need for additional action such as swapping cryptographic keys or presenting identification documents.

FaceScore is a web-of-trust where solo selfies are the vertexes and group selfies are the edges.

FaceScore leverages the fact that when users meet each other they are inherently confirming that each individual is the person that they present in their pictures, making FaceScore a novel, exciting and valuable social networking feature that encourages users to meet.



While Mary and Viktor have never met and Bob and Sophie have never met, they have confidence from FaceScore that they are each the person that they present in their photos thanks to the web-of-trust built by their group photos with other users.

The real FaceScore web of trust scales to millions of users.

ID Verify

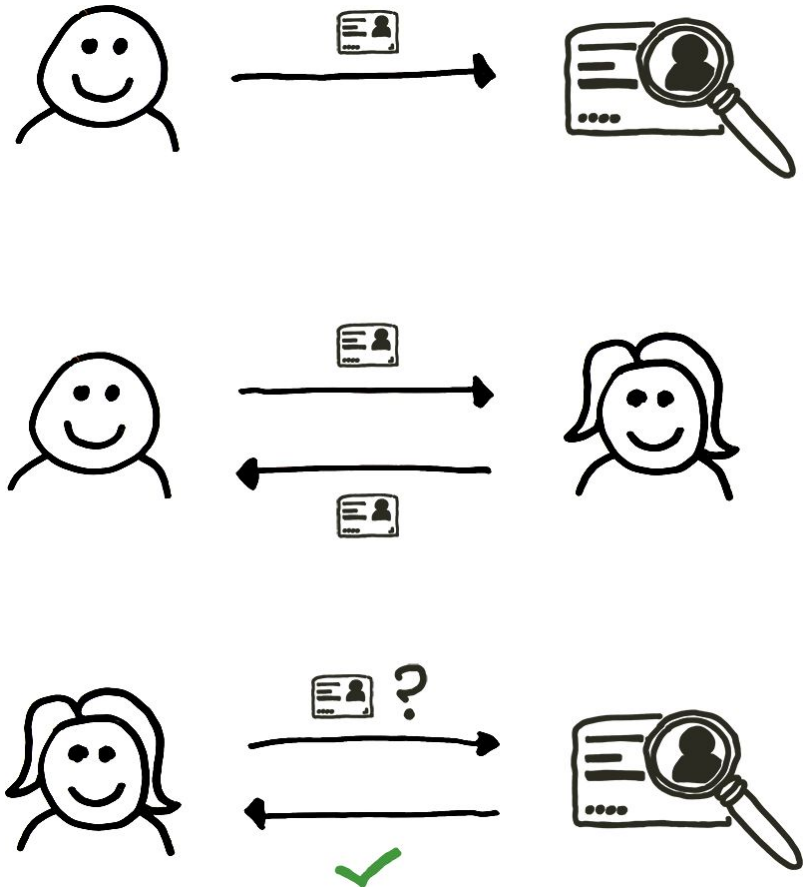
KYC ("Know Your Customer") refers to the process of confirming and assessing an individual's identity. Typically businesses perform B2C KYC on their customers and clients to meet regulatory needs. Regulators also require that certain types of business must also assess the risk that trading with a particular individual may make them vulnerable to money laundering, political bribery, and other types of fraud (aka "AML", Anti-Money Laundering).

SELFII ID's ID Verify feature is a private and secure, light KYC framework specifically designed for users and services to confirm a user's identity[1]. Users can perform simple KYC with a registered KYC provider via a SELFII ID client. Once completed, a user can then allow other users to verify their identity. As KYC has already been performed, verification occurs in real-time [2].

User to user verification allows individuals to verify each other's identity before meeting for business, meeting for pleasure, conducting business online, etc.

ID Verify is a paid feature of SELFII ID as performing KYC has an inherent cost. Applying the fee at the time of positive verification provides immediate value for money for the inquiring user. Additionally, KYC providers are encouraged to make the KYC process itself free of charge to encourage maximum user adoption.

[1] ID Verify will not fully comply with regulatory requirements for most jurisdictions, and it is not intended to do so. Our legal and technical members are working together to explore the feasibility of a regulatory compliant KYC that can be referenced on the blockchain, and we hope to announce something in the future. Compliant KYC would greatly simplify the KYC process for business, such as currency exchanges, ICO based crowdfunding, financial lenders, online casinos, etc.
[2] Depending on Ethereum transaction times.



1. Bob performs KYC by uploading an image of his passport and providing his telephone number to a provider of his choice.
2. Bob and Alice meet on a social network app. They share their identities.
3. Alice queries the KYC provider for verification of Bob's identity and agrees to the small fee if successful[3]. The KYC provider returns VERIFIED, UNVERIFIED, or UNKNOWN.

[3] Fees only apply if Bob has performed KYC and ID Verify returns 'VALID'.

Alternatively, services may perform ID Verify for its users at the service's expense. This may be beneficial when user to user id verification is critical to the service (such as an online dating app). Centralised services can perform ID Verify once per user and store the outcome to further reduce costs.

KYC providers that wish to participate can register their service on the blockchain. Full details of the service must be supplied, including a link to their website. Existing KYC providers are incentivised to participate by adding a long tail to the revenue generated by a single KYC process and by performing payments automatically and instantly.

Allowing multiple KYC providers ensures the process remains decentralised and allows providers to compete on price, ensuring there is no monopoly on SELFII ID's ID Verify feature.

ID Verify may be used within a participating service. It may also be used by users that meet on any off-chain services by using a SELFII ID compatible app.

Initially KYC providers will be added by SIL but this will be opened once an appropriate method of decentralised governance is in place to manage low quality providers; such as a voting mechanic for services or token holders to blacklist providers.

Stake

Stake is a powerful feature of SELFII ID that allows users to add SELFII tokens to an identity and give permission to services to lock the tokens for a period of time if the user breaks their community guidelines. In addition to locking the tokens, the tokens cannot be withdrawn. The lock period will be either 3 days, 3 weeks or 3 months depending upon the number of previous infractions.

Stake provides confidence to services that a user won't post inappropriate content. This confidence allows services to implement features that are beneficial to services and users alike.

In essence, users can voluntarily put their tokens ‘at stake’ as a commitment to being a good online citizen.

As the average good citizen can withdraw their stake at any time and even bad citizens can withdraw their stake after a lock period, the feature is effectively free.

Some services may implement features that depend upon the amount of stake that a user has committed such as publishing their content to a wider audience.

Spam & Abuse Prevention

Many services struggle to manage spam and abuse. User to user communication may need to be disabled; complex user to user relationship permissions may need to be implemented; and user to public communications (such as comments or forum posts) can cost a huge amount of time and effort to moderate. SELFII ID offers a simple solution that makes spam uneconomical and disincentives abuse by requiring a commitment of stake.

Good Citizens

Good citizens sign into a service that requires a stake to post public content and user to user messages. The amount of required stake is displayed either at sign-in time or at post time and only if the user has not already committed enough stake. A simple button click adds the required stake to the user’s identity and the user can proceed to post.

At any time, the user may remove their stake using a SELFII ID client app. While they do not have enough stake they may no longer qualify to post content or participate in certain features on the services that require it.

The user experience for good citizens is both simple and free.

Bad Citizens

Spam

Spammers use scripts (aka bots) to automate the process of registering to a service and publishing a post; usually with a URL to a product or malware. Typically services need to employ human verification (HV), costly manual moderation, and identity heuristics to block automated scripts, delete spam content, and ban spam users.

Human verification works to block bots by posing a challenge that is easier for humans to complete than for automated scripts. As spammers are financially incentivized to circumvent human verification they, or freelance hackers, work on doing so. When spammers have overcome a HV challenge, the HV providers must make the challenge more difficult. In effect, HV providers and spammers are locked in an arms race that results in HV challenges that are increasingly difficult for both bots and legitimate users to complete.

Identification heuristics work to prevent spammers by scoring users based on their email address, region and ip. Spammers are again incentivised to circumvent this process and do so by acquiring new domains and employing proxies to spoof their ip and location.

Both HV and ID heuristics result in false positives that further frustrate legitimate users and results in a drop off in conversions. Many users simply leave a service's website or app without registering.

Once a spammer is identified and blocked they can simply continue their campaign by automatically:

- Using new email and IP addresses to circumvent identification heuristics
- Generating new URLs for their products to circumvent spam filtering
- Paying hackers and / or employing low paid labor to circumvent human verification

Some spam campaigns employ labor to complete HV challenges. Typically a spam enterprise will automate all aspects except the HV, allowing labor to complete many HV challenges per hour. As the work is unskilled and internet based, spammers can acquire workers in the cheapest labour markets around the world.

Spam is off putting to users of a service; and costly for businesses to manage. In addition to automated spam prevention, services also need to manually moderate their community for spam that is successfully posted by bots and low paid labour. Depending upon the size of the community this may incur large HR costs.

Spammers typically sell low quality, low margin services but only need a few clicks to cover their minimal costs. As long as there are disingenuous services to sell, and the spamming process remains economical, there will always be spam.

SELFII ID attacks the spam problem; not by further amplifying the spammer vs service arms race with more complex HV or ID heuristics, but by making spam uneconomical from the outset.

By requiring a stake, services make spam campaigns unviable due to the cost of concurrency. While the required stake is minimal for good citizens, it is very expensive for spammers. While a spammer can start a campaign with a minimal stake, the moment they are blocked they will need to either:

- Acquire an additional SELFII identity and commit more tokens to its stake

Or:

- Wait for the block to be lifted

To run a continuous campaign, spammers would need to commit a vast number of SELFII tokens. As posts have a very low Click-Through Rate and therefore a very low Cost Per Impression (CPI), individual posts in a campaign have a very low value. The stake to CPI ratio effectively makes spam uneconomical.

Abuse

Handling abuse on a service that allows user generated content is a much more subtle affair than handling spam. Moderation is a nuanced effort that requires understanding of the spirit of community guidelines as well as the intentions of the author and context of posts.

Services employ various moderation strategies including pre-moderation, active moderation, and crowdsourced moderation.

Pre-Moderation

When pre-moderation is used, all posted content requires approval from a moderator. Pre moderation may be applied to all users (such as commenters on articles of a reputable news outlet) or to select users (such as new users or users with previous infractions). Users create a post and then wait for approval before it is published.

Pre-moderation is a fairly expensive strategy as all posts must be reviewed and the process can frustrate users that expect a real-time experience.

Active Moderation

An active moderation strategy requires moderators to keep up to date with all published content and decide whether posts require moderation or not. This allows users to post in real-time but abusive content will be dealt with some time after. How effective and expensive active moderation is will depend on the size of the service and it's community but is typically the most expensive strategy.

Furthermore, relying only on active moderation allows abusive posts to be seen by users before a moderator finds them.

Crowdsourced Moderation

Typically crowdsourced moderation relies on users to report abusive content. This can prioritise identified abuse to the top of the moderation list, or even automatically moderate content after a predefined number of reports.

This strategy allows moderators to focus on problematic content first. Smaller business may even rely entirely on abuse reports in order to reduce costs.

Crowdsourced moderation is the least expensive strategy but also the least effective in ensuring a pleasant user experience - users will view contravening and possibly inappropriate content before moderators become aware of it.

Risk Aversion

SELFII ID Stake reduces moderation efforts by disincentivizing users from being abusive in the first place. When users contravene the Community Guidelines of the service, the service may impose infractions. Multiple infractions result in a ban, during which the user's stake is locked. The committed tokens cannot be removed and the tokens cannot be assigned to another user until the lock has expired.

When a user posts, a psychological effect known as "Risk Aversion" applies pressure on the user to avoid being abusive. Even though the user's stake will not actually be lost, it will become unavailable if the user is banned, depending on the service and their community guidelines; and the severity of the infraction. In other words, the downside of an infraction has a relationship to a monetary value and that relationship is simple to understand.

Furthermore, when a user has faced a ban, a service may choose to require a higher stake to continue posting on the service which will either increase the user's risk aversion or dissuade them from continuing to use the service at all.

Abuse deterrence will not always work; there will even be users with a “Risk Loving” behaviour but the overall reduction in abuse, coupled with spam prevention, will allow services to move their moderation strategy closer to a crowdsourced strategy and reduce overall costs.

Users of the service will have lower exposure to abuse and an improved user experience, leading to higher user retention for the service.

Grey Lists

Services may post users to a grey-list and may choose to observe the grey-lists of other services. Thus when a user is banned they may be banned from a great many other services, further incentivizing good behaviour.

Smaller services that lack the resource to moderate their community may rely entirely on other service’s grey lists.

Status

When a user commits tokens to their stake they are committing to being a good online citizen. This commitment is not only interesting to services but also to other users.

The SELFII ID framework can use the amount of stake that an individual has to differentiate them into simplified levels that can be displayed by services. Displaying levels allows users to make decisions about how they interact with other users.

Users may create group discussions or forums that only permit other users with a minimum Status level. Similarly, they may choose to only accept comments or private messages from users with a minimum level. Levels provide confidence to users that user to user interactions will be free of spam and abuse.

With the same confidence, services may promote content from users with a higher Status level as the authors are disincentivized from posting inappropriate content.

Users can allow services and other users to view their Status. Status may have a great number of uses, including but not limited to:

- Abuse free group discussions and comments
- Filtering private messages and friend requests
- Publishing content to a wider audience
- Admission to exclusive events, clubs and forums
- Access to exclusive games, tables and bonuses in a casino

Each SELFII ID identity has Status that can be increased by committing additional tokens to the identity's stake. The specific amounts for each Status level is still under review but an example may be:

Status	Stake (SELFII)
Star	1
Bronze	25
Silver	100
Gold	1,000
Platinum	10,000
Prestige	100,000

Audience

As users with more stake have more to "lose" and are less likely to publish abusive content, services may choose to publish their content to a wider audience. Additionally, sponsors may provide additional tokens to users to increase their audience when advertising their products.

Fans of popular users may send tokens as a sign of appreciation, further growing their idol's popularity. This feature requires no further technical implementation in the SELFII ID framework and can be implemented by services as required.

Vouch

While Status can be used to prove a portion of a person's wealth, it does not necessarily indicate any qualities about that individual. Vouch is a feature of SELFII ID that allows users to vouch for and attribute qualities to other individuals.

After a user adds an individual to their known identities using SELFII ID Authentication, a user can vouch for them using a SELFII compatible client. After vouching, the user has the option to attribute qualities to that individual.

Qualities are a set of standardised tags such as 'Business', 'Party', 'Renter', 'Lessor', 'Agent', 'Friend', etc that, along with their infractions (or lack-of) provide some information about a person's reputation.

Services and apps can show the number of Vouches a user has received. Additionally they can show the number of each quality that has been attributed to them.

As well as being able to view another user's reputation, Vouch may also be queried for other use cases. For example, an exclusive social network group may admit users that have certain attributes vouched for by existing members.

Roadmap

1. ## Proof of Concept

   Prove the underlying tenets of the three core features of SELFII ID by implementing functioning smart contracts, and end to end integration with off-chain services, using a private blockchain.

2. ## Technical Design

   Author specifications from the conclusions drawn by the proof of concept and apply to a technical design document (yellow-paper).

3. ## Implementation

   Implement a viable service and perform a security audit of the infrastructure, ready for use.

4. ## Application

   Create dApp and mobile apps that allow individuals to manage their SELFII identities and stake.

5. ## Integration

   Integrate SELFII ID in partner applications to enable sign on; stake related features; and status queries.

6. ## Distribution

   Distribute SELFII tokens to the community for use.

Implementation

Below is a high level description of how the features of SELFII ID work.

Token generation and distribution is performed by an ERC20 compliant Token Contract on the Ethereum blockchain.

The SELFII ID framework is implemented as an upgradable library of contracts as required by specific features. Many of these contracts accept SELFII tokens.

The SELFII ID contracts are decorated with simple forwarding contracts to allow upgrades to fix security and performance issues, as well as to potentially add additional features in the future.

Identification

The SELFII ID Identity contract maintains identification of distinct addressable users with a simple mapping of identity id to ethereum account, and a unique public key. Once an identity is signed, authentication does not require any contracts to be executed, making authentication extremely fast and completely free of gas charges.

Identity Signing

In order to create and use a SELFII ID identity an Ethereum user must “Sign” their identity by publishing a public key with a SELFII ID compatible client. The client generates a public and private key and provides the public key to the SELFII ID token contract using the appropriate message. The private key is only held by the client and is used to prove ownership of the public key and therefore the identity. The keys are typically seeded by physically signing a device screen.

The token contract associates the public key with the SELFII ID identity and returns a signature id until the user un-signs the identity.

```
function identitySign(string publicKey, string identityId) returns (string signature)
```

As Ethereum's inherent authentication is used to identify the user at signing time, the user does not need to backup or duplicate a generated key pair. Users may sign a SELFII ID token as many times as they like, for example to use their SELFII ID token on a number of client devices. As long as the user has their Ethereum credentials they can use their SELFII ID identity for authentication.

This has the added benefit of the user not having to remember any passwords or passphrases, or having passwords stolen, hacked, or brute forced. Services do not hold any authentication credentials, so they are also immune to large scale hacks that harvest user credentials.

If users wish to have multiple identities clients can provide a feature to name them for easy selection.

ID Authentication

Services and individuals authenticate a user against their SELFII ID identity by encrypting a transient message with the user's public key and then asking the client to decrypt the message. If the client is able to decrypt the message then the service knows that the client holds the private key and is therefore the owner of the identity.

This mechanic allows the entire authentication process to be extremely fast. Services need only perform a read only transaction with the blockchain via a local node to get the public keys while clients do not need to interact with the blockchain at all.

Using the blockchain allows users to authenticate with a wide range of unrelated services using a single identity without requiring a centralised organisation to manage the identity or private data.

Additionally, as a public key is published on the blockchain and is not provided to services directly by the clients, man-in-the-middle attacks are obviated.

Services provide a button for authentication. If a user has multiple signed identities then they are prompted to select one, otherwise authentication is immediate. A Javascript library will be available to services for simple integration into their websites. Further libraries will be available for native client apps.

The button uses a registered protocol handler SELFII ID:// so that non dApp (decentralised application) clients can hand over the sign in process to a SELFII ID compatible app.

For users using a dApp compatible browser such as Mist or Chrome with MetaMask, the button works without the assistance of an external application.

When the user signs into the service, the SELFII ID compatible client sends their identity id and the signature that they wish to use. The service then queries the public key for the identity.

```
function identityGetKey(uint identityId, [uint signatureId]) returns (string publicKey)
```

The service encrypts a transient message with the public key and sends it to the client as an authentication challenge. When the client receives the challenge, it simply decrypts it with the signature's private key to prove ownership of the public key and identity. The final request to the service provides the unencrypted message[4] and the service trusts that the client owns the identity.

[4] The request is still encrypted with typical protocols such as SSL/TLS.

The service can also read any public metadata associated with the user, such as FaceScore, infractions, bans, committed stake, and KYC references, or metadata defined by future or third party features.

Any further data about the user can be requested from the user the by the service, depending on the service's needs and the user's intentions.

The service or client may also displays the SELFII ID features that the service wishes to use and allow the user to grant the service any appropriate permissions that it requests.

User to User ID Authentication

Users may share and authenticate their identities outside of any service. This is particularly useful when participants have met on a service but wish to verify each other's identities (see ID Verify below).

Once a User to User ID Authentication is complete, the results are written to the participant's SELFII ID identity to be observed across devices and applications.

1. Bob requests Alice's SELFII ID identity.
2. Alice copies her identity id from her SELFII ID client and supplies it to Bob as a string or QR code.
3. Bob adds Alice as an identity on his SELFII ID client using the string or QR code.
4. Bob clicks 'Authenticate' for Alice and is given a string to copy and give to Alice.
5. Bob gives the string to Alice. Alice clicks 'Authenticate my ID' in her SELFII ID client and pastes the string.
6. The SELFII ID client gives Alice a new string to give to Bob.
7. Bob enters the response into his SELFII ID client and the app confirms that Alice has authenticated.

We are currently performing research and design to improve this process. The use of a simplified Friends list feature with client to client messaging would reduce the above process to a couple of button clicks but may require the use of a centralised service. As the process will be standardised, other clients may implement such a feature thus negating any monopoly.

ID Verify

ID Verify works by connecting users and services with KYC providers in an asynchronous manner using the blockchain as a proof of record.

KYC Providers

KYC providers register their service by sending an appropriate message to the SELFII ID KYC contract, including a set of metadata about the provider:

- Company name
- Company address
- Company website
- URL to the company logo
- ID Verify Price
- URLs to the SELFII ID compliant HTTP API endpoints required to use the service

```
function kycRegisterProvider(string name, string address,...) returns (uint providerId)
```

KYC providers can update their details from the same Ethereum address that they used to register, at any time. Registration allows users to review providers before trusting an identity.

KYC Registration

Users perform KYC registration via a SELFII ID compatible client. Users can perform KYC at any time and may be invited to do so upon an ID Verify request from a service or user.

When a user decides to register KYC, they first select the required documents and enter the required details as described by the client. Typically a passport, address, phone number, and photo.

The user can then select a provider from all of those that have registered with the SELFII ID KYC contract. The full details of each provider and their fee for ID Verify is available so that the user can review them and choose which provider they would prefer to use.

The KYC provider must authenticate the user first. Finally, the user's documents and their SELFII ID identity are sent to the KYC provider. Once the upload has completed the KYC provider can complete their KYC process internally.

Alternatively, KYC providers may perform the KYC registration process via their app and established service and attribute the user, however they must support SELFII ID Authentication to do so.

Once the KYC provider has completed the KYC process they then send an appropriate message to the SELFII ID contract to certify the SELFII ID identity with their KYC. The certification is a reference to the service. A provider can only certify a SELFII ID identity using their registered Ethereum address.

```
function kycAttest(string identity)
```

Once a SELFII ID token has been certified, services can show that ID Verify is available for a user.

For simplicity, a user may only be certified by a single KYC provider at a time. This may change at a later date if regulatory KYC is introduced, in order to be able to cover various jurisdictions and regulatory bodies,

ID Verify

Users and services can validate a user's identity using both the SELFII ID KYC contract and an HTTP API hosted by the KYC provider. The HTTP API is standardised and documented.

The ID Verify process always requires that the querying participant provides the target user's identity. KYC providers never reveal a user's identity, only whether a provided identity is valid.

Once an ID Verify process has been completed, the result is written to the querying user's SELFII ID identity to observe across devices and applications.

Typically the ID Verify process is started on a participating service's app by clicking a button:

1. The user clicks a button on a participating service's app to perform ID Verify for a specific identity.
2. The SELFII ID client app opens with the identity, the KYC provider details, and the ID Verify price.
3. The querying user clicks 'Confirm' to pay the ID Verify fee and a 'PENDING' status is displayed.
4. Shortly after, the querying user is informed by the SELFII ID client app that ID Verify is ready for the selected user.
5. The querying user clicks 'Validate Now'.
6. The querying client updates to show whether the ID was validated or not.

KYC Providers can listen for ID Verify events if they wish to perform any preparatory processing, such as caching the result in anticipation of the query. Events contain the provider id so providers can filter out requests for other providers.

```
event kycQuery(uint providerId, string reference)
```

Individual services may implement a mechanism so that users must first give permission to other users to share identities. This allows users to keep parts of their identity private until such time that that user specifically wishes to give their Identity to another user.

Additionally, services that do hold user's identities and allow sharing with others may perform ID Verify on behalf of their users. Services may cache the ID Verify results to improve performance and reduce costs. The ID Verify should only be trusted as much as the user trusts the service.

User to User ID Verify

Alternatively, users may perform ID Verify outside of any service. Users may make contact on a service that does not retain user identities, or has no SELFII ID integration (such as Facebook); yet they can still perform ID Verify using a SELFII ID client after performing ID Authenticate.

The verification process is the same as starting it from a service but is instead started within the client app on an identity that the user has previously received. ID Verify cannot be processed on an identity that has not yet been authenticated as this would allow users to spoof another user's identity.

Technical

The process of ID Verify involves both the blockchain and standardised HTTP APIs. The following workflow describes how an ID Verify request is created and completed from a technical perspective:

1. The user opens a SELFII ID client to request ID Verify for a particular identity.

2. The user reviews the identity and selects a KYC provider.
3. The user clicks ‘Confirm’ to pay the verification fee.
4. The client displays a ‘PENDING’ status for the request.
5. The client sends the appropriate transaction with the SELFII ID KYC contract.
6. The SELFII ID KYC contract associates the transaction with the requesting user id and a generated, unique reference number.
7. The SELFII ID KYC contract returns the reference number.
8. When the transaction has completed, the client will display an ‘AVAILABLE’ status and a new button ‘Validate Now’ will be displayed.
9. The querying user clicks ‘Validate Now’.
10. The client authenticates with the KYC provider using SELFII ID Authentication.
11. The client queries the KYC provider’s API service, sending the identity and the reference number.
12. The KYC provider returns a response with:
    a. VERIFIED
    b. UNVERIFIED
    c. UNKNOWN

    Where UNKNOWN means that the KYC provider has no record of the identity.

13. The KYC provider confirms the ID Verify response with an appropriate message to the SELFII ID KYC contract.
14. If the response was VERIFIED the SELFII ID KYC contract pays the KYC provider for the confirmation.
15. If the response was UNKNOWN or UNVERIFIED, the fee is returned to the querying user.
16. The SELFII ID KYC contract writes the ID Verify result to the requesting user’s SELFII ID identity.
17. The querying client updates appropriately to show the response.

```
function kycReserve(uint providerId, string targetIdentity) returns(bool
                                    success)
```

If a service takes too long to respond to a KYC request, they will be flagged as offline to prevent unnecessary requests and lost gas prices. KYC providers are considered offline when they fail to respond within 20 blocks of the transaction being completed (around 5 to 8 minutes). This requires an expiration time for KYC requests, and KYC providers can be notified of the change in availability status with an event.

```
event kycUnavailable(uint providerId)
```

KYC providers can notify the SELFII ID KYC contract with an appropriate message when they are back online.

```
function kycOnline(uint providerId)
```

They can also notify the SELFII ID KYC contract if they are expecting to go offline, for example for scheduled maintenance.

```
function kycOffline(uint providerId)
```

If a network failure leads to a lack of response to the client from the KYC provider API *after* the ID was verified, the user will still receive verification via the blockchain and their SELFII ID identity.

Permissions

Users can grant permissions to services to allow them to write infractions, vouch and certifications. Permissions are granted using a compatible SELFII ID client or service.

When a service requires permissions, they display a standardised dialogue with the stated permissions. When a user clicks 'Accept', the SELFII ID client sends an appropriate message to the SELFII ID Identity contract. The Identity contract writes the permission in association with the user's identity.

```
function serviceGrantPermission(uint serviceId, string Identity, hex permission)
```

When a service authenticates a user, they can query a user's public identity metadata which includes granted permissions for that service, along with committed stake, FaceScore and any other public data.

Spam & Abuse Prevention

Stake

Spam and abuse prevention requires that users commit stake. Users can commit SELFII Tokens to their stake using a SELFII ID compatible client.

The user selects an amount of SELFII Tokens to commit to their stake. The client sends the amount to the SELFII ID Status contract, which holds the SELFII in association with the identity; and adds the specified amount.

```
function stakeCommit(string Identity, uint quantity) returns (hex status)
```

Users can remove their stake at any time using their client. The committed SELFII Tokens will be returned to the sender account.

```
function stakeRemove(string Identity, uint amount) returns (hex status)
```

Infractions

With a user's permission, services can apply infractions on a user. Services may reject use of the service if the permission is not granted by the user. Other services may simply rely on the user's existing infractions and ban status instead of requiring write access, depending on the size and influence of the service.

When a user gives permission to a service to be able to write infractions they are shown a Terms & Conditions dialogue that makes it clear that the service does not hold their SELFII Tokens. Instead, the user is signing a contract with themselves that the receipt of infractions may lock their SELFII Tokens.

Services that wish to write infractions must register their service by sending the appropriate message to the SELFII ID Reputation contract. Once a user has identified themselves with the service, the service can query the SELFII ID Status contract for the user's stake. The registration must be performed using the service's registered ethereum account.

```
function registerService(uint serviceId)
```

When a service writes an infraction, they send an appropriate message to the SELFII ID Reputation contract. The message includes the type and severity of the infraction. The contract

evaluates whether the infraction should lead to a ban depending on any pre-existing, un-served infraction and a confidence in the service.

```
function infractionAdd(string Identity, hex type, hex severity) returns (uint banExpires)
```

If the contract determines that a ban should be applied, it will write the ban expiration in association with the identity. The length of the ban will depend on the existence of any previous bans.

Available types, severities and ban lengths are yet to be defined.

Grey Lists

Grey lists can be optionally registered by services to share with other services a list of which users they have imposed bans or restrictions on.

To create or update a grey list with a user, the service sends an appropriate message to the SELFII ID Reputation contract including the user's SELFII ID identity, as well as the service id. Services may remove users later.

```
function infractionAddToGreylist(string Identity)
```

Services may read the grey lists of other services and use them in their decision making. Services can:

- Sync the entire grey list registry
- Sync the grey list of a specific service

- Lookup a specific user
- Lookup a specific user for a specific service

As lookups are read only operations, they can be performed locally, free of gas charges.

Services can subscribe to a contract event that informs them when a greylist has changed so that they can update their local copy.

```
event infractionGreylistUpdated(uint serviceId)
```

Vouch

Users can vouch for individuals and attribute qualities with the Vouch feature using their client app. Clients apply Vouch by sending an appropriate message to the SELFII ID Reputation contract.

```
function vouchToken(uint tokenId)
```

Clients and services can query the Reputation contract for the available qualities that can be attributed to users. Clients can display the available qualities to the user for them to attribute to another individual's identity.

```
function vouchGetQualities() returns (string qualities) // JSON
```

Qualities are attributed by the client by sending the appropriate message to the SELFII ID Reputation contract. Qualities are defined as bit flags, which are combined and attributed to the

appropriate SELFII ID token to reduce data costs and to enforce the predefined list of available qualities.

```
function vouchAttestQualities(uint tokenId, hex qualities)
```

Open tags and comments are not permitted as they may allow for leaking of personally identifiable information.