Universal Proof of Uniqueness

Last Printed: 12/07/2019

Link to Living Draft with Comments

Overview

BrightID is a social identity network. It is superficially similar to a social network, but its purpose is to allow people to prove to applications that they aren't using multiple accounts.

BrightID solves the unique identity problem through the creation and analysis of a social graph.

BrightID is a public good that exists for the benefit of humanity. It is a nonintrusive, decentralized, open source technology seeking to reform identity verification--and thus lay the groundwork for a free and democratic society.

Principles

Nonintrusive

Data should be shared with peers, not a central organization. BrightID does not want your data. Names and photos are only shared between people making connections.

Decentralized

The solution should not be controlled by a single centralized organization; such an organization could commit hidden fraud.

Open

The solution should be open and auditable.

Reusable

The solution should be easy to build on and reuse.

As more applications integrate with BrightID, it becomes more likely that their users will already be verified, reducing the friction for adoption.

Properties of the social graph

The basic unit of BrightID is a connection between two people that is cryptographically signed by both. This allows the connection to be portable while minimizing the risk of fraud.

Portability of connections is essential for creating a decentralized network of computer nodes that each has a complete copy of the graph. Decentralization allows for a wide variation of analysis methods and for methods to be audited by other nodes.

Analysis

In order to make a determination about someone’s uniqueness in the system, the graph is analyzed. There are many possible methods; different methods can be compared or aggregated. We believe many different methods will be employed concurrently by various nodes on the network.

Metadata

In the methods we tried[1], we found it useful to consider additional data in the form of *seeds* and *groups*. Seeds are preselected points in the graph from which trust flows. Groups--in the sense we used them--are small, combined efforts by connected users to help someone become verified. Groups provide richer possibilities for interconnectivity than single connections and we analyzed the graph of interconnected groups.

Thresholds

SybilRank, an algorithm on which part of our research was based, was tested with the Spanish social network Tuenti. The algorithm was used to rank vertices (users) in the graph according to their likelihood of representing duplicate users (sybils). The ordered list was given to workers who manually checked and removed suspicious accounts. Having such an ordered list resulted in workers finding many more duplicates than through user reporting[2].

A manual check like the one used in the Tuenti example may not be practical, so a verification method needs to find a threshold above which users are considered unique and automatically mark them as verified. A higher threshold may result in more false negatives (unique people being mislabeled), while a lower threshold may result in more false positives (sybils being mislabeled).

Injecting Simulated Attacks

One way to automatically find a threshold is to simulate different kinds of sybil attacks and inject them into the graph at various locations before running the ranking analysis. After analysis, the rankings of the simulated sybils can be compared to the new rankings of previously verified users. The threshold is set to an acceptable level of false positives and false negatives.

---

[1] Notes and results of our tests.
[2] SybilRank

Combining Results

Applications are free to choose the most appropriate algorithms, parameters, and thresholds.

Verification methods may sample results from several other verification methods (potentially running on several nodes) and combine them.

Verification Persistence

A user typically doesn't lose a verification for falling below a threshold unless an important local change has also occurred--for example, leaving or changing a primary group or the loss of a nearby seed group.

Verification

A BrightID user acquires a verification *sticker* as a visual indication that they have the corresponding verification. The user may then visit an application that uses that verification and share a copy that has been signed by a BrightID node.

Contexts

A *context* represents a unique combination of an identification system and a verification system.

Applications can register one or more *contexts* with nodes. Contexts are a convenient way to create a boundary within which each person has at most one identifier.

For example, a dating application could have a context called "bumble," or an Aragon DAO could have a context called "xyz_dao". A person could have at most one verified Bumble identifier and one verified XYZ DAO identifier.

Nodes can individually choose which contexts they support. Applications can choose which nodes they trust to manage their context.

Level of Centralization

The number of applications or independent nodes participating in a context determines its level of centralization.

Centralized

An application could choose to use its own id system and operate its own BrightID nodes for verification. The resulting context would be centralized and opaque. Users would have to trust that those in charge of the application were not adding sybils themselves.

A chat forum for a centralized organization might choose such an approach. The organization could use BrightID verification to permanently ban disruptive users and the risk to application users that the organization would want to add sybils is minimal.

Decentralized

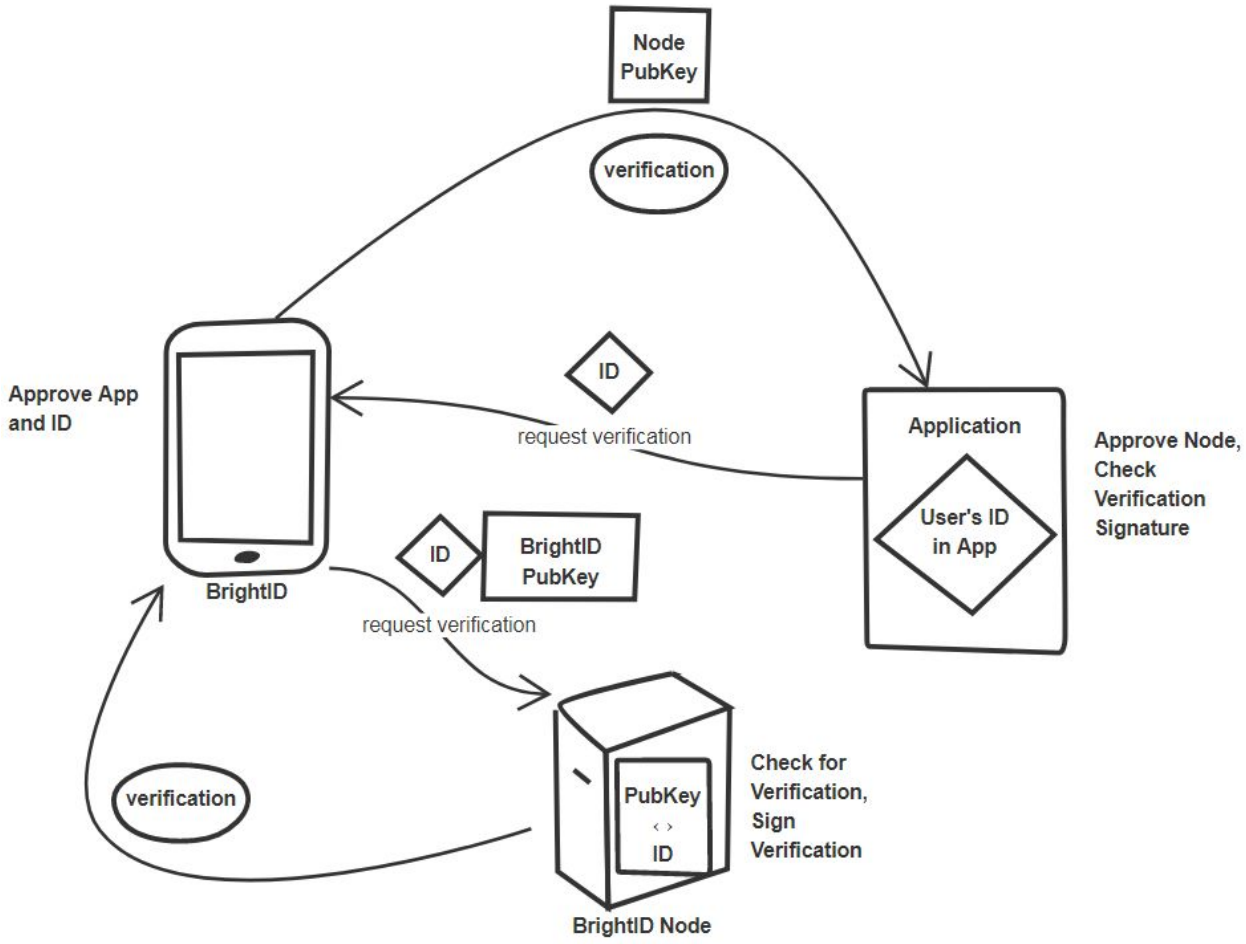
Several applications could choose to share a decentralized id system. For example, they might use blockchain addresses as identifiers. To create a shared context, participating applications also have to agree on a verification system.

An application could achieve greater decentralization by choosing to allow multiple unrelated nodes to manage its context.

One of the benefits of using a decentralized approach is that nodes can audit each other to make sure that none of them is introducing sybils. This transparency can help users be more confident that no internal fraud is happening.

An example of an application that would benefit from a decentralized approach would be a Universal Basic Income application where there could be risk of internal fraud by a centralized organization.

Process



A BrightID node stores a mapping between ids in a context and BrightIDs. An application only needs to store which ids have been verified. If it considers a user's verification to have expired, it can request an updated one. The application never needs to know a user's BrightID. This reduces the risk of accidental linkage of application user profile data to a BrightID.

To accommodate users changing the id they use, the signed verification response from a BrightID node includes a list of "revocable ids" which tells an application which ids a user previously used which are no longer mapped to a verified BrightID. The application may remove those ids from its list of verified users. Since the revocable ids list is part of the verification response signed by the BrightID node, it can't be tampered with. Revocable ids are never

removed from the verification response, only added to it. Verification responses are timestamped, which prevents users from reusing old verification responses.

Smart Contract

A user's id in an application can be an Ethereum address. There is a BrightID smart contract that checks nodes' signatures of verification reponses and maps verifications to Ethereum addresses under contexts. A user (or application or BrightID node) can pay the gas to submit a signed verification response to the smart contract after which other smart contracts can check whether an Ethereum address in a context represents a verified BrightID user. This is done without exposing any BrightIDs to applications or publishing them on the blockchain.

Auditing

If a context uses multiple independent nodes, each node shares signed verification requests with the other nodes via the peer-to-peer protocol.

Successful verifications can be posted to a public register such as a blockchain where they can be read by applications and audited by nodes.

Alternatively, verification responses can be returned directly to an application. The application can check verification responses from multiple nodes as a way to audit them.

Reputation (Stars)

People collect reputation points known as "stars." Once a person has been verified as unique, they begin to accumulate stars on a regular basis and have the opportunity to collect more by making connections to other people who become verified as unique. Stars can be spent for rewards from participating applications. Stars can't be transferred to other users.

Purpose

Stars have two main purposes:
1. Guide users to make actions that help them become verified.
2. Help users avoid becoming complacent about making connections to people they don't actually know or trust by requiring something to be staked.

Provisional Stars

To encourage people to take early actions that will lead to becoming verified, new users earn provisional stars. When a user becomes verified and sponsored, a certain number of provisional stars become real stars.

Staking

When a user makes a connection, they stake some of the stars they are due to receive in the future. If, after a certain time, the user they connected to doesn't become verified as unique, they lose the stars they staked.

Unstaked connections

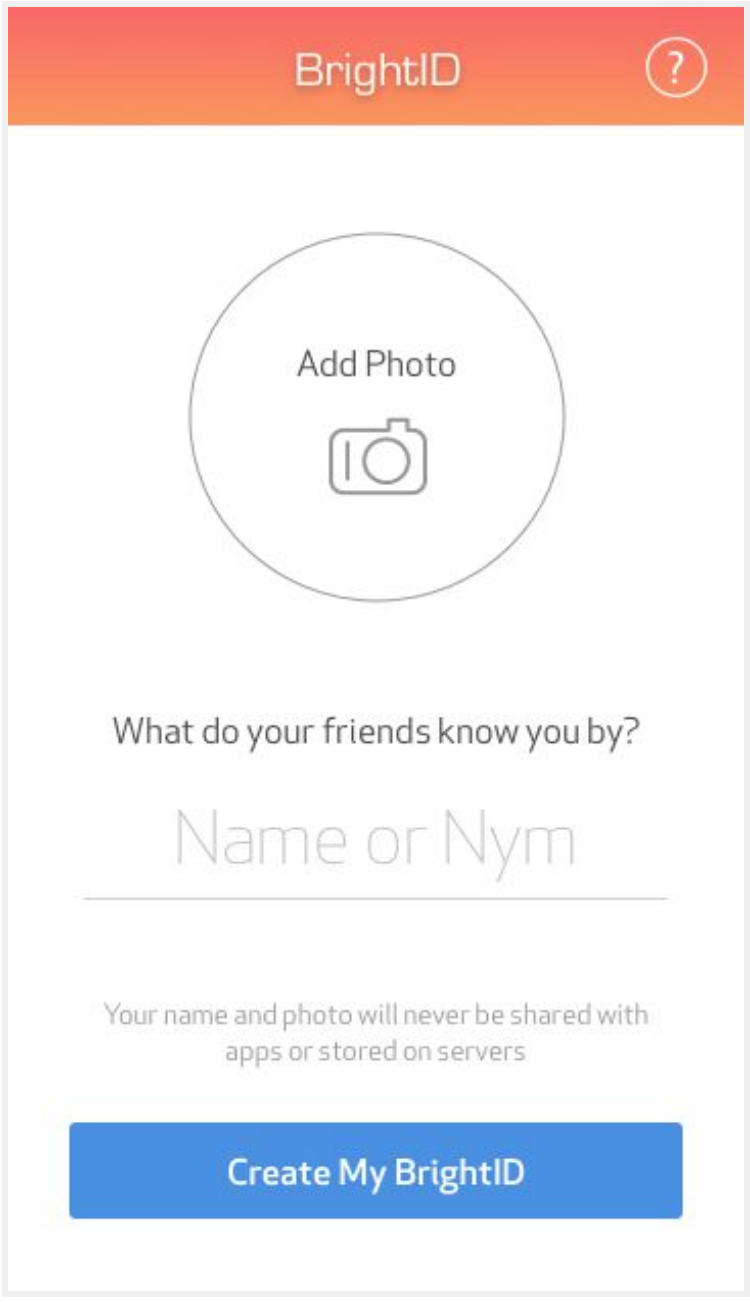
An unstaked connection is also possible. Unstaked connections are useful to avoid situations where someone is being pressured to make a connection they don't feel comfortable with. The second user will not see the fact that the first user marked a connection as unstaked. An unstaked connection won't contribute to either user becoming verified.

Indirect Staking

Making a connection to a verified person also involves staking. If that person in turn connects to not-yet-verified users, anyone connecting to them takes on part of the risk indirectly. A reward or penalty that results from a new user's verification status propagates several steps through the graph, weakening with each step.

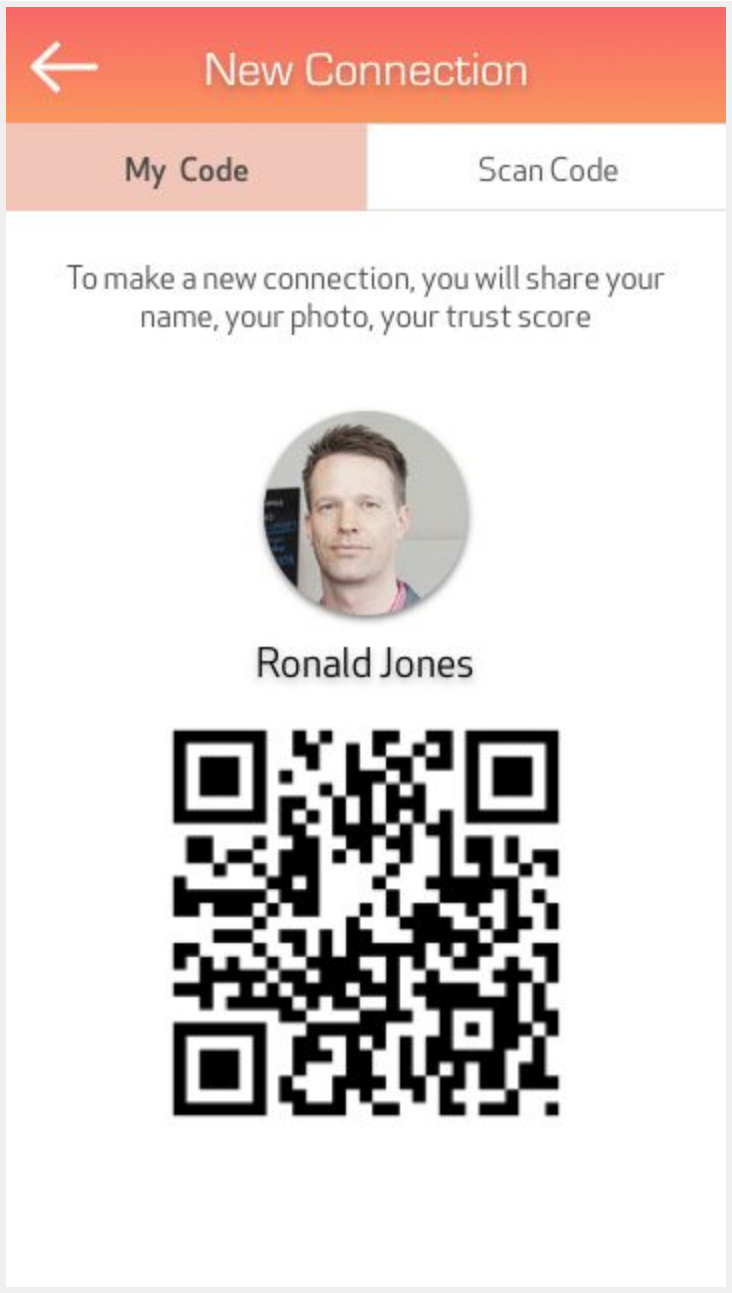
Mobile Reference App

The first mobile app to allow users to interact with the BrightID network was created through a grant from Aragon. It allows users to connect to each other, form groups, receive verifications and send them to applications. Other capabilities include data and identity recovery, primary groups, and reputation management. Some of the screens and UI flows of the application in its current and planned form are described below.



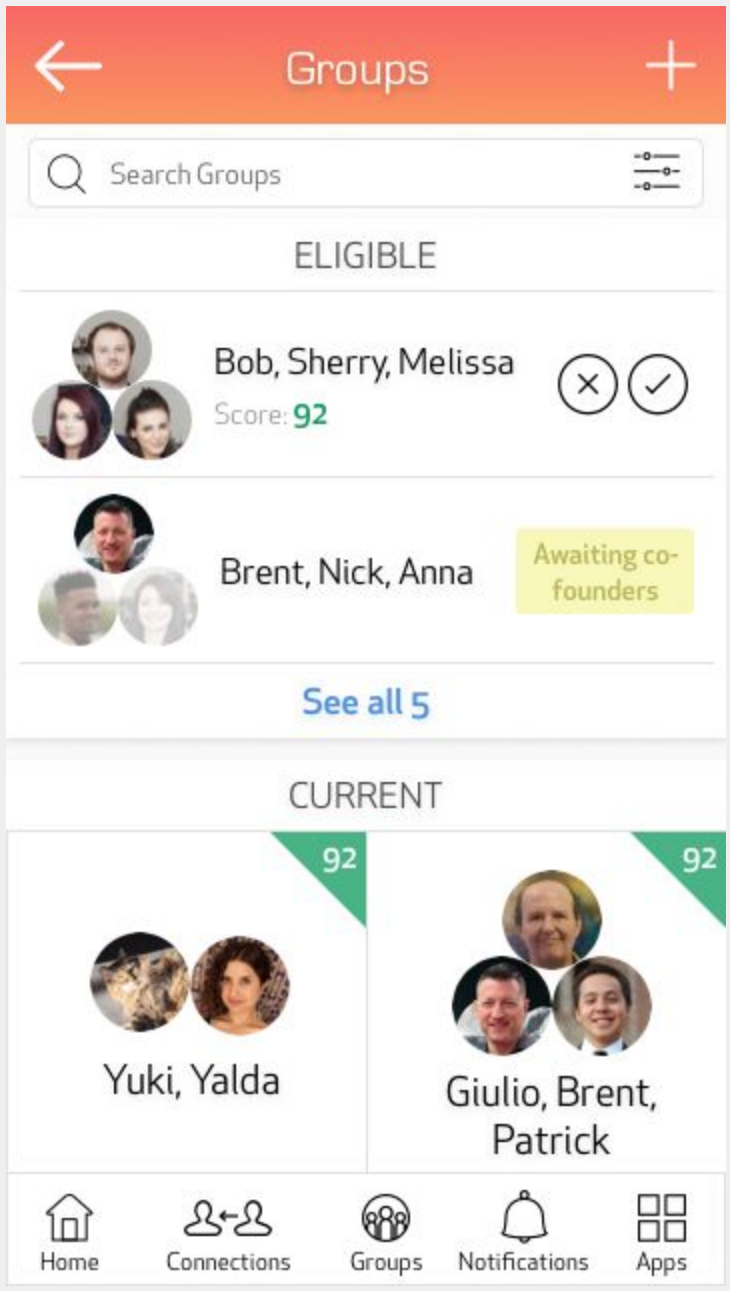
Signing Up

A new user is asked to add a photo and enter a name for themselves. This is used by other users to help them manage their connections. The user's name and photo are privately shared with their own connections, but never stored on servers or sent to apps.



Making a Connection

Each user opens BrightID and taps “connect” on the home screen. One user creates an on-screen code that the other user scans. They each confirm the connection. The other user’s join date, number of connections, and reputation (stars) are shown. If something looks wrong, a user can cancel the connection or choose not to stake reputation on it.



Groups

Groups help verify unique individuals. Each user needs to belong to at least one group to be verified. (See primary groups.)

Users are automatically joined to a group if they are connected to 50% or more if its members.

People can found new groups to help each other become verified. Each group is uniquely identified by its three cofounders.

Flagging Users

Flagging a user can be done from a connection list or group member list. A user can be flagged as fake, duplicate, or deceased. When a user flags another user, their previous connection is marked as removed on BrightID nodes.

A group member that is flagged by two other group members is removed from that group.

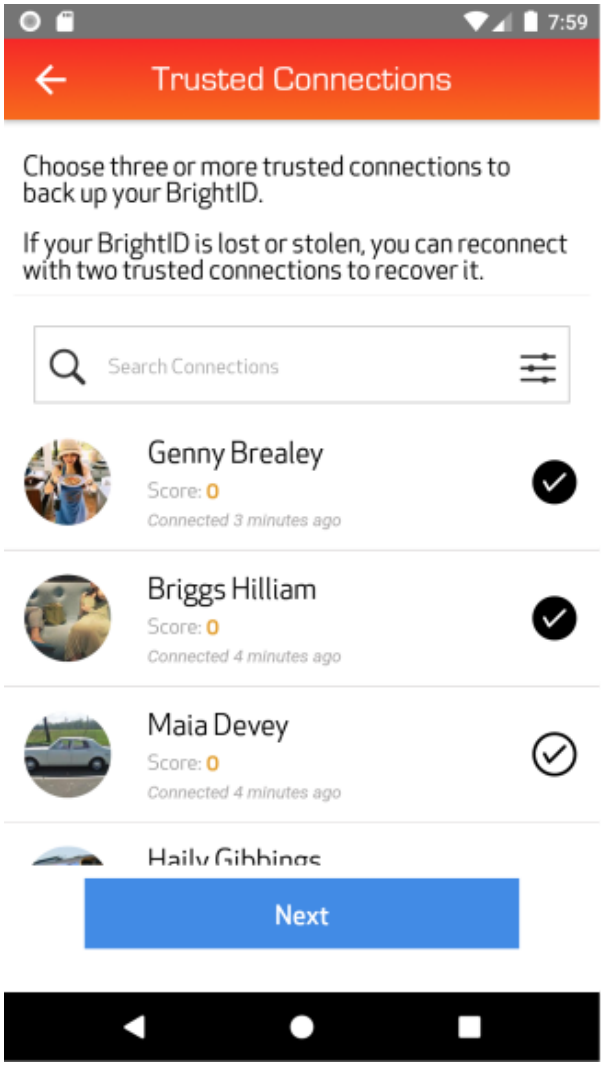
Applications

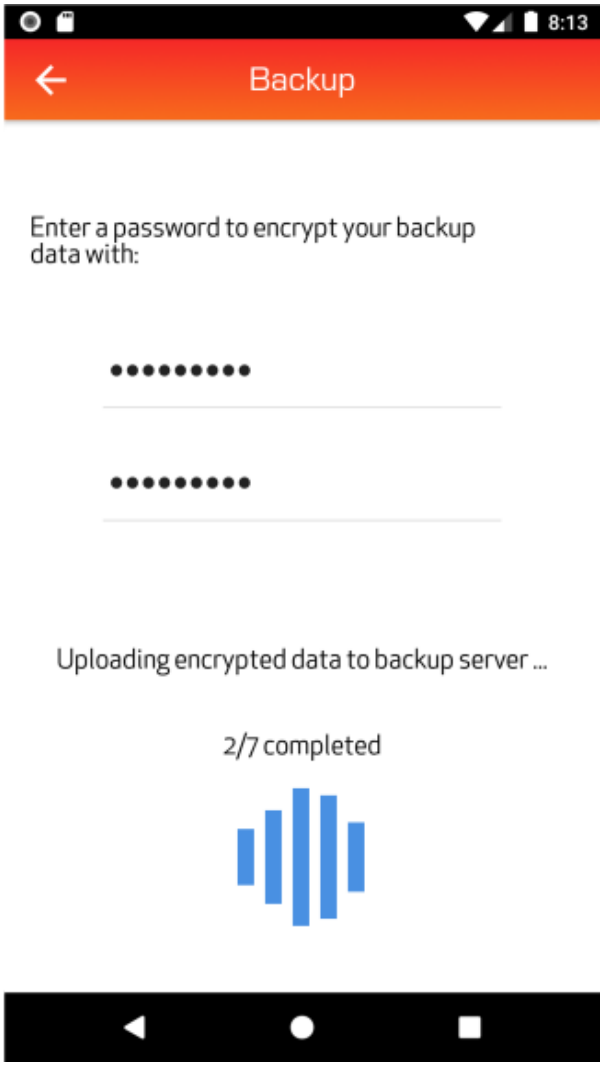
Verifications are requested and sent to applications as outlined in the verification section. After a user has been verified to use an application, it appears in the “apps” screen.

Recovery

Each BrightID has a signing key pair associated with it. If a user loses access to the signing private key, or the key is compromised (e.g. a device is lost or stolen, or the data is erased), it can be easily replaced by reconnecting to two members of a set of trusted contacts. Having a quick and easy recovery method makes bribery less effective: a user could simply accept a bribe and then replace their signing key, rendering the previous one useless.

Backup



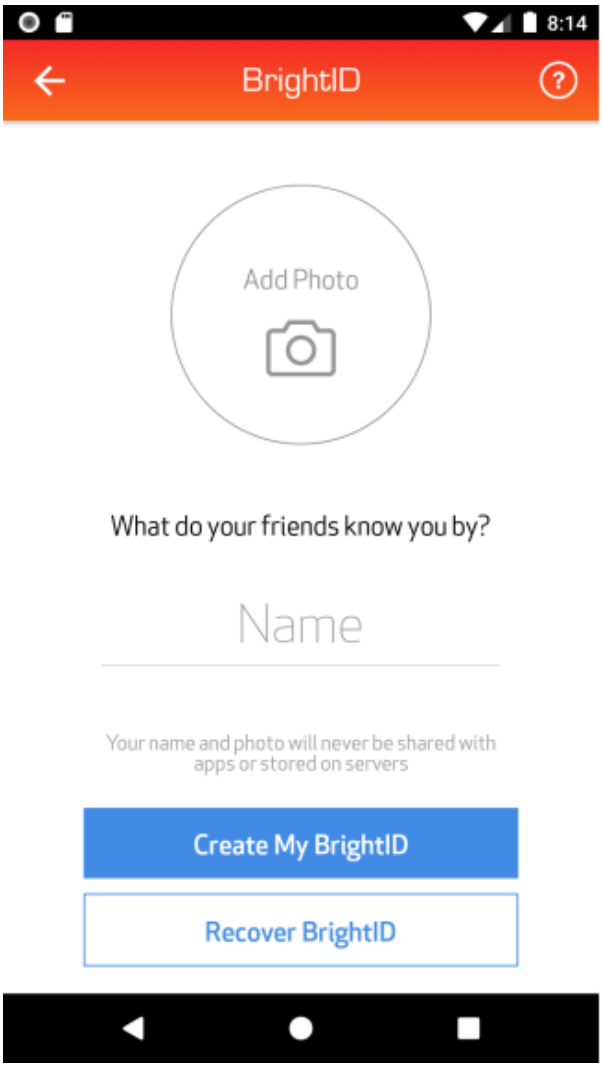


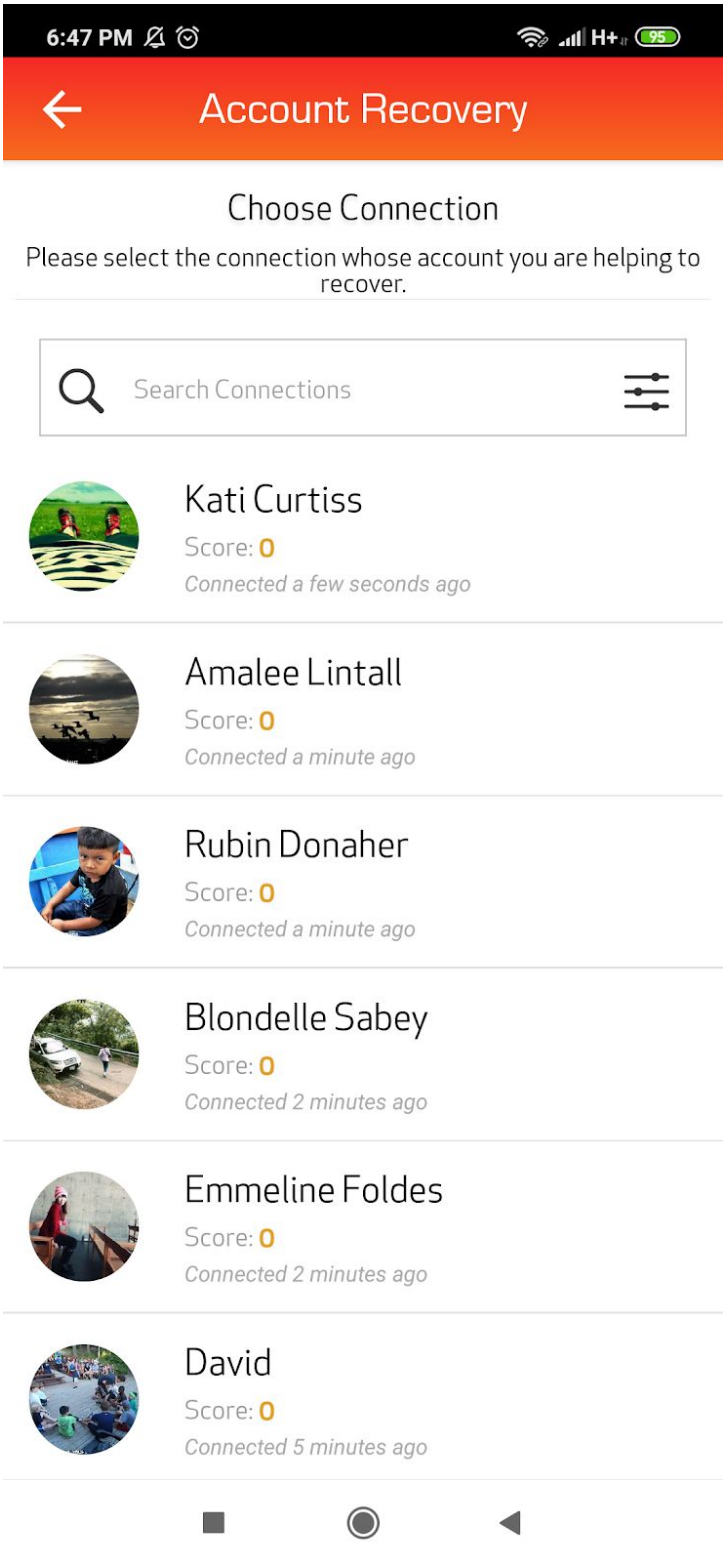
A new user is prompted to select at least three people from their connection list to serve as “trusted connections.” The user trusts them to not collude to take over their account.

At this time, the user also sets a password with which to encrypt the private data on their device.

Recovery Phase

A fresh installation of BrightID has an option to recover a BrightID. In this phase, the user is prompted to make connections to two of their trusted connections.

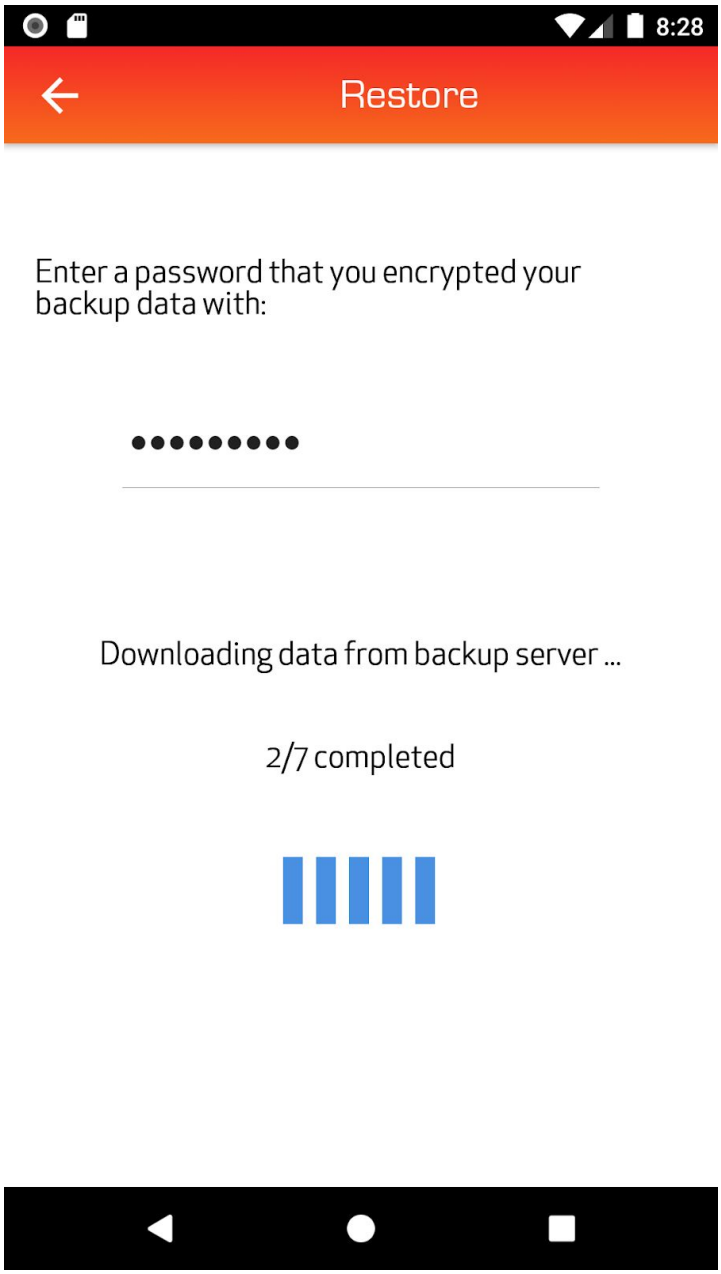




Trusted Connections

Their trusted connections will clearly see that this is a special “recovery” connection. They will be shown a list of people for whom they are serving as trusted connections. From this list, they will select the person who is currently recovering their account.

After making two recovery connections, the substitution of the new signing key for the old is recorded on all BrightID nodes and the old signing key is now useless.



Backup and Recovery of Private Data

A user's device holds the names and photos of a user's connections. This private data is never stored on nodes, but can be stored password-encrypted on a trusted connection's device or in the cloud.

If one of the trusted connections has this backup, it is sent to the user during the recovery phase. The user can then unlock it with a password.

BrightID will initially provide a centralized service for storing and retrieving password-encrypted backups.

Primary Groups

Primary groups are an important part of receiving verifications. Verifications are first assigned to groups through graph analysis. A person receives the same verifications as their primary group.

Each person chooses one primary group. Other members are notified when this happens. Over 50% of the members of the group must authorize a person's choice of primary group before the choice is allowed. In addition, any member may mark a group as unusable as a primary group and any member may veto another member's use of a group as a primary group.

A primary group represents the closest personal contacts (e.g. immediate family members) for a particular person. BrightID users should mark groups and veto other users accordingly.

Seed Groups

Some social graph analysis systems have a notion of pre-trusted people, known as *seeds*[3]. Seeds are used by the system to differentiate between honest regions of the graph and sybil regions created by attackers to resemble honest regions.

Selecting seeds is especially important during the rapid growth phase of the network where subgraphs of users may arise that are not well-connected to the main graph.

BrightID will promote the research of different seed selection methods and also the creation of tools that make seed selection scalable. Some principles to consider when creating a seed selection process are outlined below.

[3] SybilRank

Group Structure

When analysis is done on a graph of groups (as is the case in several of our initial systems[4]), it makes sense for a seed to be a group of people. This also allows a seed to have a continuous lifespan.

Additional requirements

Joining a seed group can have requirements beyond connecting to more than 50% of existing members. For example, a new member may require an invitation from an existing member and there may be an authorization process similar to a primary group.

New Seed Groups

Regions

New seed groups are meant to serve communities that are becoming interested in using BrightID, but are having trouble becoming verified due to lack of connectivity to other areas of the graph. Such communities can be regional or virtual (e.g. connected by a common interest).

Accountability

In creating new seed groups, there needs to be a balance between accountability and agility. At least one existing seed group or seed group member needs to take responsibility for the new group, but the process for creating a new group must be rapid enough to accommodate the natural growth of the network.

Auditing Seed Groups

It's important to measure the effectiveness of a seed group and aggressively revoke its seed status if it isn't effective[5]. To be able to do this, repeated measurements in the community being served--including a baseline measurement--must be taken. These should be taken by members

[4] Notes and results of our tests.
[5] An ineffective seed group could be an indication of its use by an attacker to facilitate the creation of sybils.

of other seed groups not directly responsible for the group being audited. Measurements include tracking a sample of people in the region and whether they would rank high enough (in a rank-based sybil detection system) to achieve certain verifications.

An open question is: “Who has the power to revoke seed status from a group?” A logical answer would be that any seed group above another group in the stream of responsibility has this power. The methods used to review and revoke must be manageable at scale.

Seed Group Tools

Members of seed groups will have added functionality in the mobile app to help them manage their responsibilities.
There will also be an additional seed monitoring tool that allows users to view seed groups, the chain of responsibility, and how audit data has changed for regions over time.

Additions to the Mobile App

Inviting to a Seed Group

Seed group members will be able to invite their connections to the seed group. Receiving an invitation is beyond the normal requirement of a new member needing to be connected to over 50% of current group members.

Designating a New Seed Group

A member of an existing seed group should be able to designate a group they belong to as a seed group. The member needs to specify a unique region for the seed group that it’s meant to serve. The existing seed group becomes the parent of the new seed group.

Members of the parent group (and any ancestor groups) are notified of the new group, since they are responsible for monitoring it. These members are able to see the group in their list of groups even if they don’t belong to it.

Revoking a Group's Seed Status

Any member of a seed group's parent or ancestor group can revoke its status. They enter a reason which is recorded for the seed monitoring tool along with the BrightID of the member that did the revoking. Revoking a group's seed status causes any groups underneath it in the hierarchy to also have their statuses revoked.

Auditing a Region

When making a connection, any user can designate the connection as an "audit" and provide a region. This causes a snapshot of that person's verifications to be stored along with their BrightID and the region for processing by the seed monitoring tool.

Seed Monitoring Tool

The seed monitoring tool allows anyone to view the hierarchy of seed groups and which regions they are meant to serve. In the hierarchy, it also shows revoked seed groups with their regions and the reason they were revoked.

Users can evaluate the progress of regions over time as measured by the presence or absence of verifications as recorded by audited connections. Audited connections can be filtered by the BrightIDs of the recorders to exclude noise or fraudulent reports.

The goal of monitoring a region is to evaluate whether its corresponding seed group is doing a good job and aggressively revoke seed status from underperforming groups.

Peer-to-Peer

The peer-to-peer system in BrightID is a decentralized way of sharing cryptographically signed operations that affect the social graph or adjacent systems stored on nodes. The operations relate to connections, groups, verifications and sponsorships. Anyone may run a BrightID node and receive this information.

The graph analysis that underlies verification is done locally and independently by each node. Other nodes and applications can audit the verifications produced by nodes.

What follows is a high-level description of the peer-to-peer system being developed.

Requirements

Nodes should periodically agree on the state of the graph and certain related data (e.g. groups) so that they can compute identical verifications if desired. Verifications from untrusted nodes should be audited.

A malicious node has nothing to gain from selectively hiding data, so byzantine fault tolerance isn't a requirement.

Design

Time Periods and Phases

Nodes operate according to pre-agreed time periods which are divided into an analysis phase and synchronization phase. Time periods are long enough to allow a complete analysis of the graph by nodes. Operation sets are designated for each time period. Client-assigned timestamps can be used to create an ordering within operation sets.

Operation Forwarding

Signed operations are forwarded to other nodes using a gossip protocol. Depending on their type, received operations may be added to a local operation set to be applied at the beginning of the next synchronization phase or they may be applied immediately.

Operations Applied Immediately

Operations related to creating users, founding groups, flagging users and authorizing primary groups are applied immediately to provide feedback to users. This is possible because they don't affect the verifications currently being returned by nodes. Operations related to verification and sponsorship requests can also be applied immediately.

Creating Users

An operation to add a new user can be added to an operation set at the same time it's immediately applied to the graph; a timestamp isn't needed to add a user since there is no operation to delete a user.

Add user operations can be ordered in their own section at the front of an operation set alphanumerically by the user's BrightID. When the operation set is applied, most of the users will already have been added, but some may be missing on some nodes and will need to be added before connection and group operations can be processed.

Founding a Group

The individual action taken by each cofounder to found a group can be applied immediately. Deleting a group in the founding stage can be applied immediately (it can be later re-founded with the same three co-founders).

The conversion of the group from the founding stage to the active stage is not applied immediately, but added to the next operation set. The operation contains all three signed requests by the cofounders.

Flagging Users

Flagging a user for removal from a group can be applied immediately. The second, confirming flag triggers the actual removal. The removal operation isn't applied immediately, but is added to the next operation set. The operation contains both signed flag requests.

Primary Groups

A request to use a group as one's primary group can be applied immediately. Each affirmative authorization vote can be applied immediately. A veto can be applied immediately.

The addition or removal of the primary group marker for a user is not applied immediately, but is added to the next operation set. The operation contains all the signed authorization votes or the single signed veto vote.

Verification Requests

When a user requests a verification to send to an application, it should be forwarded to any other nodes that handle that verification.

Sponsorship Requests

The signed request for a sponsorship returned by an application should be forwarded to other nodes so they can record the mapping.

Designating a Trusted Connection for Recovery

The signed request for updating a set of trusted connections for recovery should be forwarded immediately.

Updating a Signing Key (Social Recovery)

Signed recovery connection requests should be forwarded to other nodes immediately so they can also update the signing key after social recovery.

Operations Not Applied Immediately

Other operations aren't applied immediately, but are added to the current operation set. These operations may affect analysis and are therefore not applied until the current analysis phase is completed. They include adding and removing connections and joining and leaving groups.

Visibility To Users

Even though adding or removing connections aren't applied to the graph immediately, they are applied in the BrightID mobile app and will appear as having been completed to the users that initiated them.

Other users viewing group membership won't see that another user has joined or left a group until the operation set is applied at the end of the next synchronization phase.

Verifications won't be updated until after the next synchronization phase.

Time Period End

At the end of each time period, the current operation set is frozen and new requests begin to go into the operation set for the next time period.

The current analysis phase has ended. The synchronization phase begins.

Synchronization Phase

A synchronization phase begins after the operation set freezes. A new set of verifications have just finished being computed and can now overwrite the old ones in the database.

The synchronization phase gives nodes time to order the operations in the frozen operation set and compute a checksum after each operation. A checksum is a hash of the previous checksum and the current operation.

Nodes use the synchronization phase to send each other missing operations and to decide which operations to move from the frozen set into the next set (or vice-versa) in order to achieve consensus. Once consensus has been reached about a frozen operation set, it can be applied on all nodes.

Automatically Adding Users to Groups

After operations have been applied there will be a deterministic way that nodes decide the order in which people are automatically added to groups.

Analysis phase

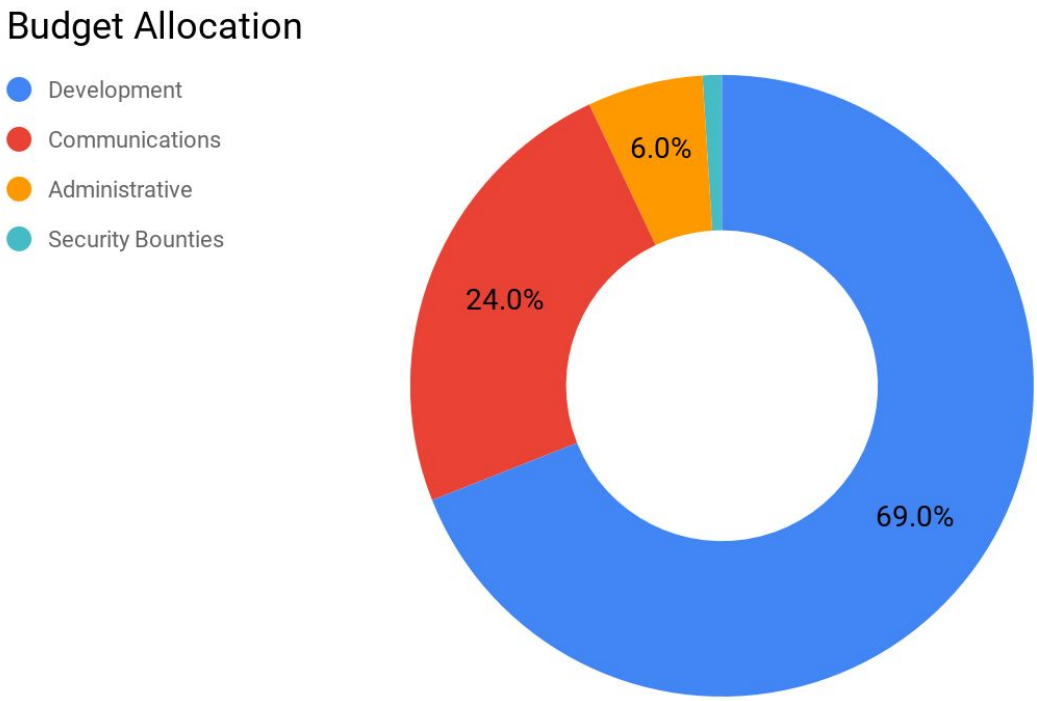
After a synchronization phase ends and the corresponding operation set has been applied, analysis begins on the new graph state. Resulting verifications are written to attributes in the database distinct from those that hold the current verifications that are now being returned to users. These will be copied over the current verifications at the beginning of the next synchronization phase.

Only after the analysis phase is finished will the time period be allowed to end and enter the next synchronization phase.

BrightID Main DAO

A decentralized autonomous organization (BrightID Main DAO) is used to represent the interests of applications using the BrightID network[6,7]. BrightID Main DAO ensures that the open-source software underlying BrightID and related efforts such as research, outreach, and documentation are supported.

The following initial budget allocation was approved by BrightID Main DAO members.



[6] BrightID Main DAO on Aragon.
[7] The Constitution of BrightID Main DAO.

BrightID Main DAO Budget Categories

Development

Fund the development of

- Applications that allow users to make connections, manage cryptographic keys and authorize other applications.
- The peer-to-peer protocol used by BrightID nodes
- Other open-source, general purpose utilities for BrightID

Research

Fund the research of new techniques for analyzing the social graph and users who wish to run nodes to incorporate these techniques. Award security bounties.

Seed Group Development

Bring remote communities into the BrightID network and fund research on seed selection processes.

Communications

Communicate with businesses, offer support, create public-facing documents including user and integration guides, host and attend events, engage in community building.

Sponsoring Users

Sponsoring users is a way for BrightID to have a continuous stream of funding while allowing BrightID to remain a public good. It spreads the burden of funding BrightID to many participants while hopefully avoiding a tragedy of the commons.

To use BrightID's verification system, a user must be sponsored--which happens just once per user per lifetime.

Sponsorship Contexts

Every sponsorship has at most one context, which indicates which application controls it.

When a sponsorship is newly created, it has no context assigned to it. The owner of the sponsorship can call a function on the smart contract to assign it a context. To prevent abuse, the context can only be assigned once.

By assigning a context to a sponsorship, it is added to the pool of sponsorships that the context (usually associated with an application) can use.

Person-to-Person Sponsorship

A person can sponsor another person. The BrightID app generates a custom signature using a person's BrightID signing key, which they can then set as the context in the sponsorship web app. When such a person makes a connection, they will have the option of sponsoring the person they're connecting to using their custom context.

Sponsorship Records

Sponsorship records are split between the sponsorship smart contract and the BrightID nodes.

On BrightID Nodes

BrightID nodes have the sponsorship mappings of BrightIDs to contexts. The number of mappings to a context indicates how many sponsorships have been used for a context. If a person's BrightID is mapped to a context, that person has been sponsored.

Signed sponsorship mapping requests are shared with other nodes via peer-to-peer.

In the Smart Contract

The number of sponsorships assigned to a context in the smart contract indicates the number of sponsorships that exist total for the context (both used and unused).

Sponsorship Check

Before a user's BrightID app requests a verification from a node, it first gets a signed sponsorship request from the application (context) requesting the verification (if it has sponsorships available). The BrightID app sends this to the node with the verification request. If the user was already sponsored or the user can't be verified, the sponsorship isn't used and can be used again by the sponsoring app. If the user is verified and needed a sponsorship, the user will now be sponsored; and the signed sponsorship request will be sent to other nodes through peer-to-peer sharing.

If a user doesn't have a sponsorship and the application requesting the verification didn't provide one through a signed sponsorship request, the verification request will fail. The user should get verified through a different application that has sponsorships available.

Auditing

Since sponsorships are checked at verification time, sponsorship mappings can be audited by auditing verifications.

Sponsorship Purchases

Sponsorships can be purchased from a smart contract for a fixed price[8]. Sponsorships are not transferable after a context is assigned.

Each BrightID user needs to be sponsored once in their lifetime. Their sponsorship is good for all applications.

Roadmap

Early supporters of BrightID will fund eight months of expenses needed to prepare BrightID for wide adoption.

Each of the eight milestones below represents an important new release that will change the way BrightID is used. Each is intended to generate interest and bring more people onto the platform. From a technical standpoint, each milestone allows BrightID to scale the number of verified unique people it can support.

BrightID will grow, collect feedback and release new versions at the same time development work is occuring.

[8] The choice of token and purchase price is set by BrightID Main DAO. Initially, the price will be 1 DAI. Scalability and price stability (the price should increase to match inflation) are the most important considerations.

brightID

Eight Month Roadmap

Month 1

Sponsorship

Applications and users can sponsor new participants, which funds BrightID.

Month 2

Improved UI

Improved mobile UI based on beta testing.

Month 3

Backup & Recovery

Social and cloud recovery of keys and contacts.

Month 4

Groups & Seed Tools

Guard against social engineering through "primary groups."

Allow for rapid growth through seed group creation tools.

Month 5

Reputation & SDK

"Stars" show how users are progressing and are redeemable through apps.

Making connections in other apps is possible with an SDK.

Month 6

Peer-to-Peer

BrightID operates on many nodes sharing data in a decentralized manner.

Month 7

Sybil Defense Modeling

Improve existing sybil defenses through extensive modeling.

Month 8

General Release

Scalability and usability testing followed by a mainstream release.

BrightID Team

Adam Stallard
Project Lead

Adam has worked in distributed systems since 2010 and client-server architecture since 2002. BrightID is his passion project.

LinkedIn, Twitter, Github

David Wisner
Mobile Lead

David is a Javascript developer with 6 years of experience building web apps. He graduated from the University of Washington with a degree in psychology.

LinkedIn, Twitter, Github

Aleeza Howitt
Research, Communications

Researcher and writer specializing in social impact projects, co-founder of Astro Ledger, publisher at UBI Research, Ronin scholar, and graduate of Tufts University.

LinkedIn, Twitter

Alireza Paslar
Research, Communications

Paslar does community management, research, and content creation for BrightID.

LinkedIn, Twitter, Github

Mahdi Heydari

Research, Mobile, Smart Contracts, Web Apps

Mahdi has over ten years of experience in programming. He also has years of research experience in economics and social criticism on banking and monetary systems.

LinkedIn, Twitter, Github

Mohsen Khan-mohammad-zadeh

Research, Smart Contracts, Web Apps

Mohsen has years of experience as a programmer. He has been researching blockchain and smart contracts for some time and is a trusted and fluent programmer in the blockchain ecosystem.

LinkedIn, Twitter, Github

Advisory Team

Philip Silva

Strategy, Mission

Philips helped create ZeroPoverty and the non-profit HedgeForHumanity to advance the ideas of universal sharing, crypto-UBI, and paying social dividends for all of humanity.

LinkedIn, Twitter, Github

Griff Green

DAO, Commons

Community manager for TheDAO, co-founder of the White Hat Group, Giveth, and the Commons Stack, as well as advising many other core Ethereum community projects.

LinkedIn, Twitter, Github

Luke Duncan

DAO, Commons

Luke Duncan is an advocate for open source technologies and decentralized platforms. He co-founded 1Hive and is working to advance DAO usability and adoption on the Aragon One team.

LinkedIn, Twitter, Github

Auryn Macmillan

DAO, Commons

Auryn is a community builder and user researcher with a passion for open technologies. Founder of DAOhub, BD;SM at Colony, former pro basketball player, MSc Psych & Research Methods.

LinkedIn, Twitter, Github

Further Reading and Links

BrightID Links

- Mobile Reference App
- Website
- DAOs

Socials

- Github
- Twitter
- Telegram
- Keybase
- Riot

- Discord

Research

1. Anti-Sybil Systems

Videos

- BrightID in 2 minutes (intro video)
- Aracon 2019 Demo / Presentation
- Social Coding - 07/09/2019 - “Show and Tell”
- OpenUBI talk
- TEDx talk: “How much poverty should exist in the world”

Articles

- BrightID: A Personal Stamp of Uniqueness by Bowen Sanders of Giveth
- Decentralized Unique Identity via Graph-based Sybil Detection on a Peer-to-Peer Credit Network by Aleeza Howitt
- BrightID: Becoming a World Citizen by Alireza Paslar
- Aragon’s Impact on BrightID by Adam Stallard
- BrightID: Proof of Digital Uniqueness by Philip Silva, Adam Stallard, Rachel Gordon for MIT Solve

Podcast Episode

Understanding Unique Identity with the BrightID Team

Projects Using BrightID

- Dollar for Everyone (community spec)
- BurnSignal
- BlankDAO
- Mannabase (planned)
- SwiftDemand (planned)

- Value Instrument (planned)
- Aragon (planned)

Thanks

Daniel Jeffries for laying fertile ground with Cicada and Why Everyone Missed the Most Mind-Blowing Feature of Cryptocurrency and creating the virtual center of the decentralized universe. Everyone who followed us on decstack, especially Alex Howlett for the endless feedback sessions. Arthur Lunn for intros to Giveth and Aragon. Everyone at Giveth. Yalor Arnold for planting a seed. Everyone at Aragon. Yalda Mousavinia for making BrightID seem real. All volunteers and team members past and present. Everyone we met at Aracon, DGOV Council, and OpenUBI Jan 2019. Robert Gilman and Bright Future Now for the courage. Grace Rachmany for writing the book on DAOs and pointing us to the stars. Ross Campbell for making DAOs legal. Han Hegeman for web hosting and design. Anyone willing to talk unique identity, including Joe462, Austin Fatheree, Michael Ten and others in /r/CryptoUBI, Andrew Whitham, Santhan Naidoo, Hadar Rottenberg, Kenneth Thomas, Jose Gonçalves, Chris Gregorio, Carsten Munk, Matt Czarnek, Anna Blume, Jeff Emmett, Dani Bellavita, Pol Lanski, Vojtěch Šimetka, Lorelei Loie, Nick Emmons, Peter Grassberger, Jordi Baylina, Maria Gomez, John Light, Bingen Eguzkitza, Brett Sun, Jorge Izquierdo, Luis Cuende, Jouni Helminen, Gorka Ludlow, Ed, Drummond Reed, Alex Zimmermann, Jan Berchtold, Aaron Foster, Alfred Guo, Yoni Assia, Gilad Barner, Darrell Duane, Doug Kent, Johannes Zerbst, Philippe Honigman, Craig S. Page, Aiden Pearce, James Waugh, Dmitry Christie, Christopher Seifert, Titusz Pan, Ramona Phelps, Ben Kaufman, Christian Hildebrand, Thomas Zeinzinger, Martin Batiste, Eric Maublanc, Hugo Trentesaux, Olivier Jansens, Martin Köppelmann, David Terry, Josh Fairhead, Luuk Weber, Slava Balasanov, Johan Nygren, Tina Roh, Daniel Schmidt, Lawrence Lanoff, Alejandro Machado, Raphaël Mazet, Ilya Kachalin, Didi, Raph Carrier, Itamar Caspi, Hendrik Richter, Rick Stefanowski, Jordan Mack, Jerry Michalski, Rouven Heck, Lucas Geiger, Matt Prewitt, Glen Weyl, Abishek Punia, Tim Draper, Tomer Kagan, Jeff Dance, Craig Hansen, Trevyn Meyer, Justin Tuttle, Santi Siri, Kyle Graden, Rich McAteer, Niran Babalola, Petr Porobov, Seth Goldfarb, Vipin Bharathan, Bertrand Juglas, Nave Rachman, Clement Lesaege, et. al. Everyone who tested BrightID. Seed group members: Kay Gertler, Bowen Sanders, Kris Decoodt, Josie, Pete-ster, Rachel Gordon, Jen Hansen, Heather Stallard, Chuck Peters, et. al. Hedge for Humanity (Brandon, Eric, Jon, Ken) and Code the Change Stanford (Drew, et. al).