BitShares ID Technical Details

Invictus Innovations aims to make crypto-currencies like Bitcoin and BitShares a well as secure communication systems like BitMessage easy to use and understand.   To do this, it is critical to hide implementation details such as public and private keys from users who should never have to directly interact with a Bitcoin Address.   We have set out to design a system that allows users to pair a human-readable name with a public key in a decentralized, globally unique manner.

Before getting into the implementation details it is important to establish proper design goals. The first consideration is how many users should the system be able to support in a decentralized manner and what does it mean to be decentralized.

If we only look at e-mail and aim to be a decentralized solution usable by everyone on the internet, then we need an idea of how many accounts this would entail.   To begin with I would like to cite some statistics from current email services:

Google: 425 Million Active Users
Microsoft: 360 Million Active Users

If you throw in Yahoo and Apple and eliminate the redundancies, there is easily demand for 1 Billion users. So the target for our design is 1 billion users while keeping the system requirements light enough to remain decentralized.

To be decentralized, the resource requirements of the network must be such that the average bandwidth when fully scaled is less than a streaming radio station (128 kbit/sec) and the maximum disk usage when fully scaled is less than 100 GB.   By the time this system is widely adopted 100 GB should be less than 5% of the average drive and 128 kbit is already less than 5% of the average internet connection. For our purposes we consider this level of resource usage to be low-enough that the average user would not notice its presence on their home computer.

To be decentralized and establish globally unique names in a secure, ‘trust-free’ manner, is best achieved with something akin to a blockchain.   We have the following requirements on the network:

1. storage for names is not free, nor is network traffic so we must avoid spam.
2. discourage creation of unused names.
3. recover unused names or names for which the key has been lost
4. revoke names that have been compromised
5. Light clients should be able to independently verify results produced by untrusted servers.

Based upon these rules, names must expire and we have adopted a 1 year window. A name must be renewed at least once per year or it will become available again.

Based upon this constraint, we can calculate the maximum size of a transaction if 1 billion users are going to renew their names once per year and the storage requirement is capped at 100 GB then the average transaction must be under 100 Bytes. This would equate to an average bandwidth of 3.25 kilobytes/sec.

Now this number is revealing because the average Bitcoin transaction is 500 bytes. Using a system like NameCoin would require multiple transactions per-registration as well as all of the transactions associated with using NameCoin as a currency. Consequently, I estimate that NameCoin could support at most 100 million users. Unfortunately, NameCoin also requires the full block history and would thus have 100 GB / year growth in the block chain to run a full node. Consequently, NameCoin does not meet the design criteria.

We can also rule out any block chain with a 'currency' component to it because the currency adds extra transactions and requires signatures. A signature alone consumes 66 bytes leaving only 34 bytes left for the rest of the transaction.

Introducing the BitShares ID Block Chain
I will start by introducing the block header, this header defines how blocks link together once every 5 minutes with a valid proof of work.

Type	**Field**	**Description**
uint16_t	nonce	used for proof of work
uint32_t	utc_sec	timestamp in seconds from 1970
uint224	previous	sha224 hash previous block
uint64_t	name_hash	hash of the user's ID
char[33]	public_key	compressed ecc public key
varint	reputation_points	total renewals of this name_hash
uint32_t	age	first block number this name was registered in
uint160	trxs_hash	hash of all transactions included in this block, similar to the merkel root in the bitcoin block chain.
optional<char[66]>	change_sig	signature used to cancel or transfer name_hash to a new public key.

Some things you will notice about the block header, it actually registers a name to a public key and facilitates changing the public key for a name. You can only change the public key associated with a name in the block headers so we can maintain the creation of light-clients that can verify that a transaction is included in the block chain and that it hasn't been revoked given only the block that contains the transaction and the headers for 1 year worth of transactions. Because a light client cannot inspect every transaction, it cannot be sure that a node providing a valid name registration is actually providing the most recent and up-to-date transaction. However, by certifying that names can only change ownership or be canceled in the block header we assure the following properties:

1. names are expensive to transfer (you must solve a block) and thus squatters will have a harder time transferring ownership.
2. if your private key is compromised, you can make sure even light clients know it.

Name Transactions

They are identical to the name headers!   The only difference between a name transaction and a block header is the difficulty required to produce them which is the greater of the minimum proof of work or the block target difficulty / 10000.   Effectively, name registration and renewal transactions are merge-mined with the blocks and at scale, 1 out of every 10,000 name registrations will be sufficient to solve the block.

To ensure that all nodes actually do perform merged mining, every name earns a certain number of reputation points. One point each time it is renewed.   If its renewal solves a block, it gets one point for each transaction included in the block. These reputation points combined with the age of a name can allow web services to filter out throw-away names and fight Sybil attacks. A forum could use it to filter/prioritize posts or even limit who can create an account. As a result reputation points have some value to accumulate on an identity because they distinguish identities with a lot of investment from throw away ones.

Reputation points also enhance the security of your name. It is very difficult to simultaneously forge the name, age, and reputation points in a parallel chain. So for secure communication, you would probably tell someone both your name and registration block number as well as your current reputation points.   This would be an unforgeable, yet easy to understand ID. It is kind of like giving your name, date of birth to disambiguate people that share your same name.

A block header is nearly 100 bytes, but individual transactions are only about 70 bytes because they may omit the 'previous' field which must be identical to the block headers previous field. This can be validated by verifying the proof-of-work on each name registration transaction by converting it into a header.

Proof-of-Work

Because the proof-of-work is validated for every transaction and at the target scale there will be 3 transactions per-second on average and the proof-of-work will be validated multiple times in the life of the transaction, it is critical that the proof-of-work can be quickly validated without saturating an individuals CPU. It is also desirable to restrict / limit ASIC attacks, so the proof-of-work will be both memory hard and computationally hard to the extent that an individual hash takes less than 10 ms. This should restrict GPU and ASIC attacks by making them more expensive than any value that can be derived by rapid mining.

Nonce

Another factor of the proof-of-work is that the nonce is a mere 16 bits and the time field a mere 32 bits, of which the valid range is only 14 bits leaving at most 30 bits of search space. All other fields have very specific values if you want to solve a block to maximize your reputation points. This means that to 'hash blocks faster' one must include new transactions in the trxs_hash field. Because every transaction included in the block must also have a valid proof of work, you cannot get more 'nonce search space' simply by generating a dummy transaction.

As a result of this limited nonce space, we believe that there is a maximum amount of CPU time per unique name that can be brought to bear in an attempt to solve valid blocks leaving the only place for ASIC or GPU mining to be in 'mass registration' of non-block transactions. This mass registration, while 'spam' will be costly and only serve to secure the network.

Name Hash

The actual string representation of names are not stored in the block chain, instead only the hash of the name. This means that it is possible to test and check, but not search the name database. The use of a name hash gives us the following benefits:

1. Fixed Size Field
2. Eliminate similar looking names by generating intentional collisions.
3. A degree of privacy for names not likely found in a rainbow table.
4. Language independent.

With 1 billion people, there is only a 50% chance of a single hash collision on a 64 bit number. The consequence of a collision is identical to a name already being in use. The user must select a different account name.

Because these names will be used to uniquely identify people, it is important to prevent certain scams that are achieved via look-a-like names. As a result for languages that use the english character set our client reduces the 36 character alpha-numeric set plus '-', '_', and '.' are normalized into 13 distinct characters and merges runs of identical strings. This means that registering MOON and M00N, M88N, MBBN and NOON will all result in the same hash because in certain fonts 8,B,0,and 0 can appear similar. While this will increase the collision rate, we believe it will enhance the security of users against subtle bait and switch attacks as well as typos. In effect, when you register your name you also get anything that looks remotely similar to your name.

The actual rules applied to the name hash are flexible and could be different based upon your locality or standard. At the block-chain level it all boils down to a 64 bit number that must be unique.

Security Theory

The security of this network is based upon the total number of users, and how valuable reputation points become. This network does not face the same threats as a crypto-currency like bitcoin because of the following characteristics:

1. There is no value that can be stolen by a 51% attack.
2. All value in the network is held in the user's private key
3. Government's have no incentive to shut down something that has no potential as a currency because nothing is easily transferrable.
4. The proof-of-work should be difficult enough to prevent private criminals from attempting to isolate a user and present a fake ID.
5. The security behind an ID is cumulative. If you know both the name, age, and reputation points of your contact it becomes significantly harder to forge an individual name.
6. The biggest risk is sending encrypted data to the wrong party and for 99.9% of the users this risk is insignificant compared to even the best systems we have today.
7. Documents signed by a 'fake id' are easily repudiated once the real owner is known.
8. Combine the proof-of-work with trusted checkpoints and there is little that can be done to undo a name beyond a trusted checkpoint. Ideally, these checkpoints would be daily and confirmed with your social network.
9. When a fork does occur, transactions from the minority fork are not portable and cannot be moved to the new fork.
10. When a fork occurs the client can automatically flag all contacts that no longer match and users can take measures to get a new user name.
11. Names on the minority fork were valid for the users on that fork at the time. This should not compromise any security of exchanges or name lookups that occurred using the minority fork.
12. For sensitive documents, it is important to only deal with names that have enough age to survive any fork all the way back to the last checkpoint.

Each individual name will require a minimum of about 1 hour of CPU time to create a transaction. Which means that the minimum security of the network is 1 CPU hr / year per user. We project that a user would want reputation points and to enhance the security of their name and therefore the default client will mine continuously at an average of 1 CPU hr /month per user.

The maximum security of the network depends upon competition to have your name in the network, but has an upper limit of 1G/hash per unique name assuming they are aiming to find a block.