Vegvisir Blockchain Formalized

Robbert van Renesse

Department of Computer Science, Cornell University

Vegvisir Block Graph

The system is comprised of a set of devices $\mathcal{D} = \{d_0, ...\}$. Some devices are correct and follow the specification below, but others may be *Byzantine* and depart from the specification. We denote by $\mathcal{D}^o \subseteq \mathcal{D}$ the set of correct devices.

A *block* $b \in \mathcal{B}$ is an immutable object with the following fields:

- $b.owner \in \mathcal{D}$: the creator of the block;
- $b.txs$: a set of transactions;
- $b.parents \in 2^{\mathcal{H}}$: a set of hashes of parent blocks.
- $b.signature$: an unforgeable digital signature from the owner of this block.

A hash function $hash : \mathcal{B} \rightarrow \mathcal{H}$ maps a block to a unique identifier in $\mathcal{H}$. It has the usual properties of cryptographic hash functions:

- it is straightforward to compute $hash(b)$ for any $b \in \mathcal{B}$;
- given a $h \in \mathcal{H}$, it is computationally infeasible to find an b such that $h = hash(b)$ (i.e., *hash* is hard to invert);
- given $h = hash(b)$, it is computationally infeasible to find an b' such that $h = hash(b')$ (i.e., *hash* is *collision-free*);

Let $B \subseteq \mathcal{B}$ be a set of blocks. We call B *complete* iff

- $\forall b \in B : \forall h \in b.parents : \exists b' \in B : hash(b') = h$

Note that a complete set of blocks and their parent relationships induce a *block graph*. We postulate that by the cryptographic properties of the hash function, a finite block graph cannot have cycles in it and therefore forms a Directed Acyclic Graph (DAG). We also observe that the union of two complete sets is a complete set (but not the intersection). We say that $b \leq b'$ if there is a path from b' to b. We also say that b' *depends on* b.

We call a device d *well-behaved* in B iff

- $\forall b, b' \in B : b.owner = b'.owner = d \implies b \leq b' \vee b' \leq b$

A *Vegvisir Block Graph* (VBG) B has the following properties:

1. B is a complete set of blocks;
2. There is a unique block $g \in B$ such that for all $b \in B : g \leq b$. We call g the *genesis block* of B;
3. $\forall d \in \mathcal{D}^o$: d is well-behaved in B.

We denote by $\mathcal{VBG}$, $\mathcal{VBG} \subset \mathcal{B}$, the set of of all VBGs.

We call two VBGs with the same genesis block *related*. Note that the union $B_1 \cup B_2$ of two related VBGs B_1 and B_2 is also a VBG and related to both B_1 and B_2.

Given a block b in a VBG, we define $ancestry(b)$ to be the set of blocks consisting of b and its ancestors all the way to the genesis block, that is:

$$ancestry(b) \triangleq \{b' \in B \mid b' \leq b\}$$

Ancestry is the transitive closure of the parent relationship. Note that $ancestry(b)$ is a VBG that is related to B, and $ancestry(g) = \{g\}$. Also,

$$\forall b' \in ancestry(b) : ancestry(b') \subseteq ancestry(b)$$

Note that $hash(b)$ uniquely identifies $ancestry(b)$.

We define a function $height : 2^{\mathcal{VBG}} \to \mathcal{D} \to \mathbb{N} \cup \{\bot\}$ as follows:

- if d is well-behaved in VBG B, then $height(B, d) = |\{b \in B \mid b.owner = d\}|$, that is, the number of blocks that d owns in B.
- if d is not well-behaved in B, then $height(B, d)$ is $\bot$.

We will call $height(B)$ the *vector timestamp* of B. Also, for each well-behaved device, we will call the most recent block added by d in B the *leader block* of d.

Authorization

We are now going to constrain which devices are allowed to add blocks to a VBG B by constraining the set of parents a block signed by a particular device is allowed to have. We define two special transactions `delegate(d)` and `revoke`(d). We say that a device d is authorized in block b iff both the following conditions hold:

- $\exists b' : b' \leq b \wedge$ `delegate`$(d) \in b'.txs \wedge (b'.owner \neq d \vee b'.parents = \emptyset)$
- $\forall b' : b' \leq b \implies$ `revoke`$(d) \notin b.txs$

(Note that only a genesis block can, and should, delegate to its owner.) We call a block *legal* if its owner is authorized by the block. A VBG B then has the additional constraint that all blocks owned by correct devices be legal and only depend on legal blocks.

A block b that is not legal is proof-of-misbehavior of $b.owner$, as are blocks that depend on b. We modify the *height* function so that $height(B, d) = \perp$ if B contains a block owned by d that is not legal or depends on a block that is not legal.

Reconciliation

Each correct device $d \in \mathcal{D}^o$ maintains a set of blocks $d.blocks$. The Vegvisir system maintains the following invariants:

- For each correct device $d \in \mathcal{D}^o$, $d.blocks$ is a VBG;
- For each correct device $d \in \mathcal{D}^o$, $d.blocks$ can only grow over time;
- For any two correct devices d_1 and d_2, $d_1.blocks$ and $d_2.blocks$ are related (i.e., they have the same genesis block).

The object of reconciliation is for some group of correct devices $D \subseteq \mathcal{D}^o$ and each device $d \in D$, to replace $d.blocks$ with $\bigcup_{d' \in D} d'.blocks$.

Usually the cardinality of D will be two. A simple reconciliation protocol, when there are two devices d and d' could be as follows:

1. d sends vector timestamp $H = height(d.blocks)$ to d'. Optionally d can include the maximum number of blocks that d is willing to receive in return.
2. d' uses H to determine $d'.blocks \backslash d.blocks$ (the blocks that d' has but d does not). If this is the empty set, the remainder of this protocol is skipped.
3. d' streams the blocks to d. d' sends the blocks in an order so that, upon arrival of a block, d can add the block to $d.blocks$ while maintaining the invariant that $d.blocks$ be complete. (TODO: we should prove that such an order always exists.)
4. d adds each block to $d.blocks$, after checking that adding the block does not violate the invariant that $d.blocks$ is a VBG.
5. After receiving the last block, d determines the set D^o of devices that are well-behaved in $d.blocks$. d then creates a new block b (with possibly an empty set of transactions if none are pending) with the last blocks of the devices in D^o as its parents. d adds b to $d.blocks$ and sends b to d'.
6. After receiving b, d' checks to make sure that adding b to $d'.blocks$ maintains the VBG invariant and, if so, adds the block to $d'.blocks$.

This protocol can be executed in the opposite direction concurrently. It can also safely be terminated at any time, for example if a network connection breaks. TODO: we have to prove that this protocol satisfies the invariants listed above and results in the VBGs of all devices converging to the same set.

Witnesses

Let b be a block. A correct device d is said to *witness* a block b if $b \in d.blocks$. Given a set of correct devices D, the set of blocks that are witnessed by all devices in D are characterized by $\min_d height(d.blocks)$.

The concept of witness is important in determining which blocks are durable in Vegvisir. Therefore, devices collect signed vector timestamps from the other devices. Note that in Step 1 of the reconciliation protocol described above, the devices already send timestamps to one another. We now require that these be signed. In addition, the signed vector timestamps may be gossiped between devices, that is, the sets of signed vector timestamps should be reconciled as well.

Each device keeps track of the set of vector timestamps received from its peers. For any two vector timestamps issued by a correct device, one will dominate the other, so peers only need to keep the maximum. If a device receives timestamps from the same peer that do not dominate one another, that is a proof-of-misbehavior, and the device can simply mark the peer as faulty. The set of vector timestamps form a square matrix with a row and column for each device.