Gitchain: Decentralized Git Hosting

Yurii Rashkovskii yrashk@gmail.com
Oleg Andreev oleganza@gmail.com

June 3, 2014

Abstract

This paper presents an approach for a decentralized storage of Git repositories as well as non-conflicting, fair name allocation and secure maintenance of push permissions to those repositories.

1 Introduction

Over past couple of years, increased usage of Git and similar tools has brought services like GitHub, Bitbucket and Gitorious to life. However, once a centralized service is experiencing a malfunction, network problem or a denial of service attack, wide reliance on it is severely impacting development workflow as a huge percentage of open source projects effectively go offline for the period of outage. Such services are also at risk of censorship and access breaches that can lead to repositories content alterations.

There has been at least one known attempt to create a decentralized git system, GitTorrent [3], however, it did not gain any significant traction and has been effectively abandoned.

However, recent developments such as Bitcoin [1] and Namecoin [2], combined with older ones such as different variations of DHT and public key cryptography warrant another attempt to remove a single point of failure present in today's Git infrastructure.

2 Goals

Before designing such a system, we must examine the reasons that have led to its inception and define a minimalistic but uncompromisable set of goals that such system must be able to attain.

1. No central authority of any kind
2. Secure, fair name allocation
3. Tamper-proof history of modifications
4. Redundant object storage

3 Notation and Definitions

To ease the comprehension of this paper, in this section we have compiled major notation elements used throughout this paper.

Notation	Description
$\frown$	Binary[1] concatenation operator
$\mathcal{B}(x)$	Binary encoding of x
$\overset{\mathcal{H}}{\mathcal{B}}(x)$	Binary encoding of x for the purpose of hashing
$\mathcal{H}(x)$	Hash of $\overset{\mathcal{H}}{\mathcal{B}}(x)$, defined as double SHA256
$\overset{b}{\mathcal{H}}(x)$	Hash of x, defined as per Bitcoin specification (double SHA256)
$\overset{g}{\mathcal{H}}(x)$	Hash of x, defined as per Git specification (SHA160)
$\mathbb{G}$	Set of all Git objects
$\mathbb{T}$	Set of all Gitchain transactions
$\overset{b}{\mathbb{T}}$	Set of all corresponding Bitcoin transactions
$\mathbb{E}$	Set of all Gitchain transaction envelopes
$\mathbb{S}$	Set of all states of the system
$\mathbb{K}$	Set of all key pairs
$\mathbb{K}'$	Set of all public keys
$\mathbb{K}''$	Set of all private keys
$\mathcal{S}_c(x,k)$	ECDSA signature of $\overset{\mathcal{H}}{\mathcal{B}}(x)$ over curve c, where $k \in \mathbb{K}''$
$\mathcal{S}(x,k)$	$\mathcal{S}_{secp256k1}(x,k)$

4 General Architecture

Gitchain is built around a couple of concepts, and is largely inspired by Bitcoin and Namecoin. It uses Bitcoin blockchain as a mechanism of consensus by leveraging Bitcoin's blocks and mining to secure its own transactions. Expensive, highly specialized computer farms is the most reliable way to achieve consensus. If we were to use non-specialized resources, it would be harder to gauge whether the majority of them are indeed used for proof-of-work computations. By observing that enormous amount of work happens in a very specific, easy-to-observe part of the economy, we can estimate how expensive it is to produce an alternative, equally difficult message. In case of Bitcoin mining farms, such an alternative would require a very expensive and complex production chain, requring either outcompeting other firms that use chip foundries or building single use datacenters in the most cost-effective locations on the planet.

Gitchain defines its own transaction independently from Bitcoin, however, every transaction on the Gitchain network is uniquely represented by a corresponding Bitcoin transaction that is used as an inclusion marker in Bitcoin blocks.

In addition to the blockchain, Gitchain uses DHT to store Git objects in a distributed manner and to broadcast transactions [4].

[1]Hereafter by binary we mean a sequence of bytes

5 Public Key Cryptography

Gitchain is using elliptic public key cryptography to implement signing and verification, using secp256k1 curve. Even though SafeCurves [6] lists secp256k1 as a potentially unsafe curve, we have to use it in order to match the one used in Bitcoin. There are no known attacks at this angle in Bitcoin to this day.

6 Transactions

Every Gitchain transaction $\mathbb{T}_k$ is a state modifier, such that $\mathbb{T}_k(\mathbb{S}_{k-1}) = \mathbb{S}_k$. A transaction carries a specialized message that describes the nature of this modification. There are multiple types of transactions, described further in this document.

Each transaction $\mathbb{T}_k$, authored by a holder of $\mathbb{K}''_m$ is enveloped with a tuple $\mathbb{E}_k$:

$$\mathbb{E}_k := \left\langle \underset{BitcoinTx}{\overset{b}{\mathcal{H}}(\overset{b}{\mathbb{T}}_k)}, \underset{Signature}{S(\mathbb{T}_k, \mathbb{K}''_m)}, \underset{PublicKey}{\mathbb{K}'_m} \right\rangle$$

We define envelope binary representation for the purpose of hashing as

$$\overset{\mathcal{H}}{\mathcal{B}}(\mathbb{E}_k) := \mathcal{H}(\mathbb{T}_k) ^\frown \mathbb{K}'_m$$

Each transaction has a corresponding Bitcoin transaction $\overset{b}{\mathbb{T}}_k$ that uses OP_RETURN facility to carry a magic header and a reference to the transaction using its envelope hash $\mathcal{H}(\mathbb{E}_k)$. Typically, primary output of this transaction is equal to its input.

Since it is bitcoin miners that provide block mining for Gitchain (see Mining) and there is no way to exclude invalid or conflicting transactions from those blocks, the convention is that invalid transactions should be ignored and in case of conflicting transactions, it is always the first one listed in a mined block that is chosen, all other conflicting transactions are ignored.

7 Mining

Gitchain has no mining of its own. Instead, it leverages computationally expensive mining provided by Bitcoin miners at the expense of end users having to pay Bitcoin transaction fees.

As a result, Gitchain is getting the most expensive proof of work to maintain the consensus on the state of the system and avoids the necessity to establish its own mining network, with all the risks and downsides of such a setup.

8 Name Allocation

Before any repository can be addressed (and therefore pulled from or pushed to), it needs to have a unique name, with following properties:

1. A name has a unique textual (defined as `hpath` in RFC1738 [5]) representation (to be used as a part of HTTP URL)

2. Name allocation happens on a first-come basis
3. Name can be deallocated (for repository removal or renaming)
4. Fair and reasonable name prefix allocation should be possible (for example, to claim your own base user identifier, for example `johndoe` in `johndoe/foobar`)

8.1 Name Reservation Transaction (NRT)

Similarly to Namecoin, before a name can be allocated it has to be securely reserved. The following transaction need to mature (6 confirmations) before the name can be allocated. However, unless followed by a Name Allocation Transaction within 12 blocks, it is considered abandoned.

$$\left\langle \underset{Hash}{\mathcal{H}(Name \frown Random)} \right\rangle$$

The motivation behind this is the same as in Namecoin, this is to prevent others from stealing your name before you had a chance to get it included and confirmed.

8.2 Name Allocation Transaction (NAT)

After the corresponding NRT has matured, a name allocation can happen:

$$\langle Name, Random \rangle$$

Name allocation is considered mature after 6 confirmations. It is important to note that if the name wasn't used to push a repository to within next 4320 blocks, it is considered deallocated on the 4320th block following the block in which this NAT was mined.

8.3 Name Deallocation Transaction (NDT)

After the NAT has matured, one can deallocate the corresponding name.

$$\langle Name \rangle$$

9 Repository Transactions

9.1 Reference Update Transaction (RUT)

$$\left\langle Repository, Ref, \underset{ObjectHash}{\overset{g}{\mathcal{H}}(\mathbb{G}_k)} \right\rangle$$

10 Permission Management

Upon name allocation, the holder of the private key used to allocate the name is granted full permissions to the repository.

11 Object Storage

Object storage is where Gitchain goes outside of the boundaries of transactions.

References

[1] Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008. `https://bitcoin.org/bitcoin.pdf`

[2] *Namecoin Design.* `https://github.com/namecoin/namecoin/blob/namecoinq-release/DESIGN-namecoin.md`

[3] Jonas Fonseca, *GitTorrent: a P2P-based Storage Backend for git*, 2006. `http://jonas.nitro.dk/tmp/foo/gittorrent.html/main.html`

[4] Sameh El-Ansary, Luc Onana Alima, Per Brand, Seif Haridi *Efficient Broadcast in Structured P2P Networks.* `http://www.sics.se/~seif/Publications/paper3.pdf`

[5] *Uniform Resource Locators (URL).* `http://www.ietf.org/rfc/rfc1738.txt`

[6] Daniel J. Bernstein, Tanja Lange, *SafeCurves: choosing safe curves for elliptic-curve cryptography* `http://safecurves.cr.yp.to/`

[7] Fabio Pietrosanti, *Not every elliptic curve is the same: trough on ECC security* `http://infosecurity.ch/20100926/not-every-elliptic-curve-is-the-same-trough-on-ecc-security/`

[8] Daniel J. Bernstein, Tanja Lange, *Security dangers of the NIST curves* `http://www.hyperelliptic.org/tanja/vortraege/20130531.pdf`

[9] Qingji Zheng, Shouhuai Xu *Secure and Efficient Proof of Storage with Deduplication* `http://eprint.iacr.org/2011/529.pdf`

[10] Giuseppe Ateniese, Seny Kamara, Jonathan Katz *Proofs of Storage from Homomorphic Identification Protocols* `http://www.cs.jhu.edu/~ateniese/papers/pos.pdf`