6.824 Blockstack FAQ

Q: The whitepaper is light on details. How can I find out more about how Blockstack works?

A: The authors' earlier USENIX paper explains in more detail how to use Bitcoin as the basis of a decentralized naming system:

https://www.usenix.org/system/files/conference/atc16/atc16_paper-ali.pdf

Blockstack's May 2019 white paper, while also light on technical details, has a fuller explanation of what they are trying to achieve:

https://blockstack.org/whitepaper.pdf

Blockstack's web site has lots of documentation and tutorial material on how to use their system to build applications.

Q: Could Blockstack use technology similar to Certificate Transparency logs rather than Bitcoin? After all, Bitcoin is expensive to operate, and slow.

A: CT doesn't have a resolution scheme to decide who really owns a name if two people register it; it only publishes the fact that a conflict exists. Bitcoin/Blockstack choose one registration as the winner. CT can't really use the same ordering idea because CT has multiple logs, but the logs may be different; two records may appear in opposite orders in two different logs. It is Bitcoin's expensive mining that allows Bitcoin to cause all the replicas of the Bitcoin blockchain to reach agreement without much risk of malicious forks; CT has nothing like mining to force agreement.

Malicious people would likely submit lots of name registrations to CT-based personal naming system, and consume every useful name. Blockstack/Bitcoin defend against this by charging fees for names, but CT has no corresponding defense. (When CT is used for web certificates, people have to buy a valid web certificate from a CA before they can register is with CT, but it's not clear how to build something similar for personal names.)

My guess is that open (public) block-chains with no associated crypto-currency can't have strong enough fork-resolution to support Blockstack-like naming, because they can't have fees or mining (or even proof-of-stake, since there's no money at stake). But that's just a guess.

Q: Would decentralization actually change user experience significantly?

A: As things stand now it's likely to make the user experience worse. Blockstack has nice properties in principle with respect to privacy and user control, but it's not clear that most users care much about this. In the sense that they probably wouldn't be willing to sacrifice much in the way of features or convenience. And, at least so far, it's much harder for developers to develop applications with complex features in Blockstack than in traditional centralized web sites.

Q: Does the kind of decentralization delivered by Blockstack improve user privacy?

A: The main way in which Blockstack could improve security is that data is stored encrypted in the storage system. So that, even though the storage is probably run by the same outfits that store your data in the existing web (Amazon AWS, Microsoft Azure, Google Cloud), they can't read the data. And the encrypted stored data is resistant to

malicious people exploiting bugs to break into the storage service software.

But there are a bunch of major caveats. 1) You still have to run application software, which someone else writes: e.g. if you run Facebook's Blockstack-based application, you have to trust it not to snoop on you or do anything else bad. 2) The technology for sharing encrypted data is (so far) fairly awkward; it's a pain to write Blockstack-based applications that share data among users but encrypt it for privacy. Managing the encryption keys is the problem. 3) The track record for users being able to keep private keys safe is pretty bad; in other private-key systems, e.g. Bitcoin wallets, users routinely lose their keys, lose the devices containing keys, run malware that steals their keys, &c.

Q: What if every email address were associated with a public key?

A: It's a good idea, but no scheme for this has ever gotten much traction. A name for what you're looking for is PKI (Public Key Infrastructure). Blockstack's naming system is a PKI.

A deep problem is how to find someone else's public key in a secure way in a large open system like the Internet. You could imagine a global registry, but even if the registry is entirely trustworthy, there's no reliable way for them to check ownership of e-mail addressses or identities. So they cannot really know whether the person asking to register a public key for "rtm@mit.edu" is really rtm@mit.edu. So you cannot reliably find a public key for rtm@mit.edu. And of course if the central registry turns out to have corrupt employees, or is required to comply with laws, they may intentionally give you bad information. This has caused people to search for decentralized PKI solutions, like Blockstack, but these have even weaker properties with respect to guaranteeing that the person who registered the name rtm@mit.edu really is rtm@mit.edu.

Even in more limited situations -- e.g. MIT's Kerberos, which is a PKI -- it's very hard to achieve real security. If I don't already know someone's e-mail address, it's not clear how I can find that out reliably. Even if I know your e-mail address, it's not clear even MIT can be sure that the person who registered your e-mail address is really you. Suppose someone tells IS&T that they are you, that they have lost their password or private key, and that they need to re-gain access to their account. Will that person be able to take over your account? Quite possibly, and this kind of failure happens all the time.

There are a bunch of modern messaging systems that encrypt in much the way you have in mind (Keybase, Signal), but key discovery is still a weak point for them. Of the ones I'm familiar with, Keybase has the best story.

Q: What problems does Blockstack create for application developers?

A: My impressions from having used and built systems like Blockstack is that they are significantly harder to develop for than the traditional web site arrangement.

The storage interface is restricted to key/value. So there are no powerful queries such as you get from SQL databases. And queries take a relatively long time (each must cross the Internet).

Security has to be enforced with cryptography. For data that should only be read by the data's owner, that's not a problem. But for shared data, it's pretty awkward to arrange for the required encryption. You could encrypt the data once for each permitted reader; or encrypt the

data once with a unique key and encrypt that key for each permitted user. If you want to support big groups (e.g. everyone in EECS), that's another level of complexity, particularly if the person who controls the group membership is different from the person who owns the data. If you need to retract access (e.g. if someone leaves EECS) that's even more complex.

Many web sites need to use data that the user shouldn't be able to see. For example, when I use eBay, the software I run needs to be able to tell me if I'm currently winning the auction, which requires the software to know what other people have bid. But I can't be allowed to know other bids directly, or if I'm not winning. But if the eBay software runs on my computer (as it would with Blockstack), I can modify it to tell me anything the software knows. The real eBay solves this by running the software on their own servers, and storing user bids in their own database, but the point of Blockstack is to reject that arrangement.

Many web sites have data that's really the property of the web site itself, not any user. Blockstack has no obvious way to support that.

Many web sites maintain public data that's not owned by any user, and incorporates many users' data. Indices and "like" counts, for example, or the article rankings for the front pages of Reddit and Hacker News. Blockstack has a scheme for indices specifically, but it's centralized, and it's not clear how to build other shared constructs.

Q: What is the consistency hash about?

A: Blockstack can suffer forks in its own sequence of transactions, for example if different Blockstack peers run different software versions that have different rules for accepting transactions. If Blockstack weren't careful it might never notice such a fork. But these forks could hide malicious behavior -- if (perhaps by exploiting software version differences) an attacker cause some Blockstack peers to see a given transaction, but others to ignore it. Blockstack uses the Consistency Hashes to ensure that any such fork is noticed -- after a fork, each peer will only pay attention to subsequent transactions on the same fork, and will ignore transactions on other forks. This is tricky on top of Bitcoin -- Blockstack cannot simply put the hash of the previous transaction in each of its transactions, because a given Blockstack peer is likely not aware of all the recent transactions. Thus Blockstack uses a looser scheme involving a kind of skip-list of previous hashes, so that peers can decide how close they are to having seen identical histories.