Intro to Crypto Sci-Club Notes

April 25, 2019

Abstract

Notes preparing for sci-club crypto-blockchain presentation.

List of possible topics

Start with history, review of core tech concepts. Finish with applications.

- First large-scale use of blockchain was in WWII Enigma coding machines. Maybe explain how these work?
- Concept of mixing/entropy. Concept of mixing at different time-scales (slow, fast).
- Public key crypto mathematics theory.
- Public key crypto examples: PGP/GPG, x.509 certs, SSL/https, onion routing described at general level
- Crypto at a detailed level: ratchet, double ratchet, forward secrecy, what is it for, how does this work?
- Nonce; using nonces for secret boxes; all secret boxes must have a different nonce else there is no forward secrecy
- Merkle tree (hash tree): it doesn't have to be an actual chain; it can be a tree of chains, leading up to a signed root. Thus the dat:// protocol and btrfs, zfs
  - Bitcoin has merkle trees inside of it; where? why?
- Git as a blockchain, and why it was needed - to prevent source code fraud (real, not imagined - intentional covert, secret insertion of corruption/bugs into Linux kernel by unknown actors (possibly NSA??))
  - Git is a blockchain of commits.
  - Each commit is a Merkle tree of file-objects (blobs) that captures the state of that filesystem tree at that particular moment.
- File sharing, sleepycat/napster,

  - Gnutella, BearShare, LimeWire, Shareaza, all use tiger tree hash

- torrent architectures (bittorrent tracker is example of DHT)
- Other weirdo crypto ideas: death-lottery.
- Forking and fork resolution. A fork prevents maintanance of state (e.g. maintance of bank balances) because in one fork, a withdrawl is made, and in another, there isn't (multiple accounting books).
- Forking is easy to solve, if there is a centralized authority (one set of books) – e.g. a bank. The bank is authoritative as to your bank account balance.
- Centralization has disdvantages: (a) snooping (b) high cost prohibits microtransactions (pennies)
- bitcoin predecessors: erights.org, how to do banking in a secure container by delegation. So, many small banks (purses) are possible. Trick is to keep each purse secure, honest. Possibly by cryptographical proving that the computations were actually performed.
- Distributed mutual exclusion protocols. https://www.cs.uic.edu/~ajayk/Chapter9.pdf Need: liveness (guaranteed forward progress), fairness (everyone gets a shot), safety (there is only one lockholder).
- Distributed mutual exclusion algorithms e.g. Lamports algorithm, "work", but only cooperatively, not in a hostile network.
- Byzantine generals i.e. hostile network
- Tangles aka "Observer Remove Conflict-Free Replicated Data type" (OR-Set CRDT)
  - Like block chain, but DAG with root (genesis node); growing tip/tips,
  - As defense of cryptocurrency against attackers. https://www.iota.org/research/meet-the-tangle
  - Cooperative tangles: OR-Set CRDT with vector clock: arXiv:1210.3368
  - nice diagrams: https://github.com/cn-uofbasel/ssbdrv/blob/master/doc/tangle.md
- Bitcoin solves: (1) the forking problem (2) the Byzantine generals problem.
- Note that there's still "centralization" in bitcoin, just not central authority. There is only one growing tip in bitcoin. There are NOT multiple, distributed purses. Bitcoin "wallets" are NOT purses! They are not actually wallets, because they do not actually store the value – the value is on the bitcoin blockchain, and tehre is exactly zero value in the wallet itself. The wallets do store the secret keys, though.
- HashCash for spam prevention. Proof-of-work - computationally expensive operations.

- bitcoin as generic database
- bitcoin as a ledger: addition subtraction – thus, a transaction database.
- But also: multiplication, division, if-statements, loops, – Turing-complete; generic programming that is verifiable, unfalsifiable, non-repudiable – etherium. These properties make it strong enough for legal contracts!
  - Etherium improvements e.g. rigorous mathematical proofs, from the (who were they?? I/O Hong Kong people?)
- Problems with global blockchains:
  - Global blockchains are heavy. Typically require all participants to store the entire chain -gigabytes worth and more.
  - The problem with proof-of-work - wasteful of CPU power.
  - Moral of the story: global blockchains represent a *single* centralized server. They represent a centralized resource in a distributed fashion.
  - Originally, ostensibly, blockchain was to solve the banking problems of double-spending of coin. But the solution is sloppy: it places *all* coins from the whole wide world, onto one single blockchain. This is overkill. All that was really needed was a single global representation for a *single* coin (to see if that particular coin was spent). If one cares only about a single coin, there's no real point in looking at all other coins in the world - they are of no paricular interest. And yet, bitcoin, etc. *forces* all coins to live in one single centralized repo.
  - Scaling: 7 billion people, 256/512 bytes of ID per person: 3.5 terabytes of info to keep in sync globally. That's ID's only. This does not count/record any pair-wise or N-wise interactions between peers.
  - namecoin expires (destroys) data after N blocks, charges fee to keep data alive.
- Distributed hash tables - Why. Examples IPFS, Tahoe-LaFS, and predecessors (Freenet). Also bittorent tracker. Pros:
  - decentralization; load-balancing
  - fault tolerance; data integrity
  - scalability; load-balancing
- Critique of DHT
  - The problem of spam in IPFS, etc. and potential DDOS. FreeNet forgets unaccessed data.
  - Sybil attacks: one entity (person) creates many puppet identities (sockpuppets). Identity validation. Defense against sybil remains open research question.

  - How does sybil attack against DHT actually work? Tech details?
  - Again - this is a symptom of representating a singleton: i.e. a single global copy, instead of a locally-balancing, locally-verifiable distributed system.

- Anti-singleton designs. The point is: chat is inherently P2P. Social media is inherently P2P. Financial transactions are inheretly P2P. Legal contracts are inherently P2P. So why are we using centralized designs to solve them? Short answer: its hard to avoid entralization...
- Distributed P2P social media: https://www.scuttlebutt.nz/stories/design-challenge-avoid-centralization-and-singletons
- Whisper protocols??? (SWRLDS)
- Double Ratchet algorithm
- scuttlebutt verification handshake #ssb is networking protocol
  - How does ssb compar to I2P (invisible internet protocol)? https://geti2p.net/en/
  - How does ssb compare to Noise protocol? https://noiseprotocol.org/
- GnuNet as networking protocol
- secureshare built on GnuNet has little itty-bitty blockchains everywhere
- Noise protocol - mutual authentication, identity hiding, forward secrecy, zero round-trip encryption
  - Users include WhatsApp WireGuard, Lightning, and I2P.
- homomorphic encryption
- Blockhain apps: uncorruptable, uneraseable database. When is this desirable? For what reason? Why isn't git enough? (scope of git; generic databse issues. More to life than SQL)
- Decentralized vs. Federated. What moxie said about dominating carries when federated.
- Log-structured merge tree (LSM)
  - Basic structure for distributed databases
  - Allows off-line operation - read-write-append (secure scuttlebutt)
- Apps cornerstone: unique identity; authentication; needed for voting, UBI, but also for plain-old social interaction.
- Distributed ledgers
  - ERP (enterprise resource planning) as decentralized distributed accounting ledger build on 1990's technology that fails to scale down to small business or consumer level. (too complex)

  - IOTA https://www.iota.org/ still has a strong corporate focus
  - No micro-transactions (pennies per trade)
  - no personal accounting (McDonalds emails me my credit-card receipt, instead of placing it in my personal general ledger. Ditto for Amazon, my bank, my insurance company, my doctor, etc. No privacy, because gmail can read my receipts. No automation: I have to hand-transcribe into gnucash)

- UBI apps (value-flow apps, community-value-exchange apps, mutual credit)
  - https://github.com/valueflows/vf-apps
  - fractalide
  - Input-Output Hong Kong .org https://iohk.io/research/papers/#an-ontology-for-smart-contracts
  - https://rchain.coop/platform
  - https://statebox.org/
  - https://www.stellar.org/how-it-works/stellar-basics/#how-it-works
  - Critique: UBI is income w/o voting rights. Compare to stock ownership, where you get dividend income AND voting rights!

- Liquid democracy (the need for a voting infrastructure beyond representative democracy, and beyond direct referendum)

- Apps: shard storage of mapping/geographic info; a global geographic database. With control over who sees what information (utilities, property developers want to hide geographic info). Needed for construction, property ownership, utility easements.

- "Personal control of information" (?? what does this mean???)
  - Openbook as opensouce face-book with control over information. (huh?)
  - solid.mit.edu (sucks, so far...)

- Still got political issues: EULA, DRM, anti-hacking, patents

- Anonimity: TOR, I2P (Invisible Internet Project)

- Matrix, Wire, Onion.chat, GNU Ring (Signal vs. Telegram and WhatsApp)

- Holochain for transactional guarantees

- Radicle for replicated state machines