Lecture 7 - Anonymity, Pseudonymity, Unlinkability

Rylan Schaeffer and Vincent Yang

May 12, 2016

Note: This lecture is based on Princeton University's BTC-Tech: Bitcoin and Cryptocurrency Technologies Spring 2015 course.

1 Motivation

- Since blockchain is public, we want privacy for users

2 Definitions

- Pseudonymity: real name not required, but some moniker is e.g. Reddit
- Unlinkability: a user's multiple interactions with a system cannot be tied or associated together
- Anonymity: no moniker required e.g. 4Chan

  Anonymity = Pseudonymity + Unlinkability

3 Pseudonymity

- Bitcoin (and currencies previously described) are pseuodnymous, as public keys are used
- if public key is compromised once, an adversary can link all transactions made with that key to you

  Relatively easy to do. An exchange will track who you are, since they need $. Someone you order something from will ship to your real life address.
- Sidechannels (indirect leaks of information) can reveal your identity

  Timing attacks, network usage attacks
- once public key linked to you, all past, present, future transactions will be traceable

4 Unlinkability

- Goals:

  Linking a user's multiple addresses together should be difficult

  Linking multiple transactions by a user should be difficult

  Linking a sender of a payment to the recipient should be difficult (not for single transaction, but chain of transactions)
- Achieving the last is hard, as usually your transaction is for a certain amount and the amount (relatively) uniquely identifies the transaction

  Instead, seek to maximize anonymity set of your transactions (the set of transactions that an adversary cannot distinguish from yours)

5 Transaction Graph Analysis

- Shared spending is evidence of joint control
- Can transitively link cluster of transactions as belonging to one entity
- Wallets usually have similar implementations, that can be exploited

  When a "change address" for a transaction is needed, the wallet generates a new address. This means that if a transaction's output has a never-before-seen-address, adversary can infer which output is to who

  Error prone, as users can override default behavior
- Can also identify users by "tagging" - interacting with them including pools, exchanges, wallet services, gambling sites
- Other techniques:

  directly transact with user

  asking service provider e.g. exchange

  carelessness (users will post bitcoin addresses on forums with other information related to their identity)
- Fistful of Bitcoins - transaction graph

6 Network-layer Deanonymization

- Block-chain is application layer, peer-to-peer network is network layer
- If multiple nodes on network collude, when a user broadcasts a transaction, the nodes can pinpoint the origin of that transaction, pin down the IP address
- "the first node to inform you of a transaction is probably the source of it." - Dan Kaminsky, 2011 Black Hat talk

7 Chaum's Blind Signatures

- David Chaum's Blind Signatures, 1983. Digital Signatures. Idea: Bank issues notes with serial numbers. Sellers check with bank to confirm that notes are not being double spent before confirming transactions.
- Chaum's contribution: when new note issued, recipient chooses serial number. Keeps number hidden from bank, and bank signs ("blind signature")
- RSA blind signature:

  $r \leftarrow \$\{0,1\}^k$ such that gcd(N,r)=1 i.e. r is relatively prime to N

  User creates $m' \leftarrow mr^e(modN)$. Note that m' leaks no information about m as r is a random value and thus $r^e(modN)$ is a random value

  Authority signs $s' \leftarrow (m')^d(modN)$

  Author compute signature of original message $s \leftarrow s'r^{-1}(modN)$, where r^{-1} is modular inverse of r

  This works because $s = s'r^{-1} = (m')^d r^{-1} = m^d r^{ed} r^{-1} = m^d r r^{-1} = m^d(modN)$

8 Mixing

1. Everyone sends transaction to service, service scrambles inputs, forwards to outputs
2. Similar to banking, in that service can keep track of inputs and outputs
3. can use series of mixers. If one is honest, hides your identity. Greater degree of security.
4. standardize transaction amount to better hide
5. Fees need to be all-or-nothing. Mixes expect to get paid. If they take a fraction out of a transaction, it identifies who participated. Also, cannot be sent to another mix, as the output is now smaller than the initial input. Alternative solution is probabilistic all-or-none consumption. But then need to convince users of correct functioning

9 Decentralized Mixing

- Would like decentralized mixer. Philosophically aligned with cryptocurrencies, doesn't have bootstrapping issue, better anonymity, don't have to pay mixing fee.
- Coinjoin: Find peers who want to mix. Eschange input/output addresses. Construct transaction. Collect signatures. Broadcast transaction.

  Vulnerable to denial of service attack. Anyone participating knows input-output pairs.

10 Zerocoin and Zerocash

- Zerocoin: Convert bitcoin to zerocoin, then back again. Burning original bitcoin breaks link between input and output. Each zerocoin is token that you sacrificed a bitcoin. Need proof that 1 bitcoin was sacrificed per 1 zero coin.

  Minting a zerocoin: Generate serial number S and random number r. Compute H(S, r). Publish commitment to block chain.

  Using a zerocoin: Lots of many "minted" coins in block chain. Claim "I know r" such that H(S, r) is in hash set $h_1, h_2, ..h_n$. Miners use ZKP to verify your ability to open commitment without actually opening it. Miners check that S has never previously been used. Now have new basecoin. When spent, serial number S becomes public.
- Zerocash: Builds on zerocoin. All transactions can be done in ZKP manner. Ledge publicly records existence of transactions, along with proofs to allow miners to verify correctly functioning of system. Neither addresses nor values are revealed.

  Requires "public parameters" (gigabyte long public keys). If compromised, adversary can create new zerocoins without being caught.

  Zerocash uses zk-SNARK (zero-knowledge Succinct Non-Interactive Arguments of Knowledge)

11 Zero Knowledge Proofs

- Zero-Knowledge Proofs: Provide evidence that you know something specific, without revealing that specific thing.
- First proposed in 1980s by Goldwasser, Micali and Rackoff. Ordinarily, Prover and Verifier. Switched to "What if Verifier is malicious?" How much information will leak? Goal: prove 'This statement is true' and nothing else
- Everything Provable is Provable in Zero-Knowledge. Paper by Goldwasser, Goldreich, Micali, a few others, master student Phillip Rogaway. All of NP admits zero-knowledge proofs.

- 3 Properties

  Completeness: if statement is true, honest verifier will be convinced by honest prover

  Soundness: if the statement is false, no dishonest prover can convince honest verifier except with some small probability

  Zero-Knowledge: if the statement is true, no cheating verifier learns anything other than this fact.

- Graph 3-coloring. Verifier draws graph. Prover provides solution by coloring graph. Covers each node with a hat. Verifier can randomly query. Repeat as many times as necessary, changing the coloring each time. As long as solution exists, Prover will always be able to answer Verifier query.

  Instead of hats, use commitments.

- Zerocoin relies on Strong RSA assumption and double-discrete logarithm proofs

- Pinocchio Coin at Microsoft Research uses elliptic curves and bilinear pairings

12 Zerocoin ZKP

- Previously-logged commitments $C_0, C_1, ..., C_{n-1}$

- Owner of coin generates (S, r), uses to compute C_n and adds to commitments. Prove knowledge of a $C_i \in C_0, C_1, ..., C_n$ and that $C_i = g^s h^r$.

- http://zerocoin.org/media/pdf/ZerocoinOakland.pdf

- $http://www.wisdom.weizmann.ac.il/\ naor/PAPERS/SUDOKU_DEMO/$