CMPT 982 – Information Privacy

Module 6: Extra Topics in Privacy

Blockchain

- Problem: We want a distributed transaction ledger so that no single entity has control over the ledger
- Solution: Use cryptographic techniques so that participants can verify the ledger
- We will look at a set of techniques called a blockchain
- Global transaction ledger:
  - Globally distributed to all participants (=> open)
  - Can be sent by anyone, and therefore requires some verifiability

Global Transaction Ledger

Sender	Receiver	Bitcoin amount
Alice	Bob	0.31
Carol	Bob	1.21
Alice	Bob	0.4
Bob	Alice	0.532
Carol	Bob	0.01
Bob	Carol	0.01
...	...	...

Block 1

Block 2

Block 4

- Everyone needs to agree on the same ledger
- For bitcoin, each block is about 10 minutes of transactions
  - Empty blocks are OK
- What if Bob says, "Actually, block 1 is empty – I was never paid"?

Global Transaction Ledger

- Three types of threats against the ledger’s integrity:
  - **Add**: People refuse to add a real transaction to the ledger
  - **Modify**: Someone modifies a transaction in the ledger
    - This is easiest to fix: Simply have the participants sign all transactions, and include the signature in the ledger
  - **Delete**: Someone removes a transaction from the ledger
    - To prevent those, we use a *proof of work* to safeguard blocks
- Two types of participants in a blockchain system:
  - Users: Signs their transactions and announces them
  - Miners: Helps add the transactions to the ledger by generating the proof of work (technically, miners can also be users)

Proof of Work

- A proof of work is required for each block added to the ledger
- It proves that a certain amount of computational power was spent by the miner who added this block of transactions
- Challenge: Given C and hash h(), how can we find P such that

h(C||P) *ends with 32 zero bits?*

- If h() is a cryptographic hash, then the only (best) way is brute force: keep trying new P
  - 2^{31} tries is expected
- We can also change "32" (or give a range of allowed bits) to finetune the difficulty of the challenge

Proof of Work

- Challenge: Given C and hash h(), how can we find P such that

$h(C||P)$ *ends with 32 zero bits?*

- C is a hash of the latest transaction block
- P is the proof of work to be attached to the current transaction block
- If a miner is able to publish P that satisfies the above, he has “proven” that he has spent significant computational power on the next transaction block

Proof of Work

$$B_2 = \langle T_2, h(B_1), P_2 \rangle$$
$$B_3 = \langle T_3, h(B_2), P_3 \rangle$$
$$B_4 = \langle T_4, h(B_3), P_4 \rangle$$

$h(T_i||h(B_{i-1})||P_i)$ *ends with 32 zero bits*

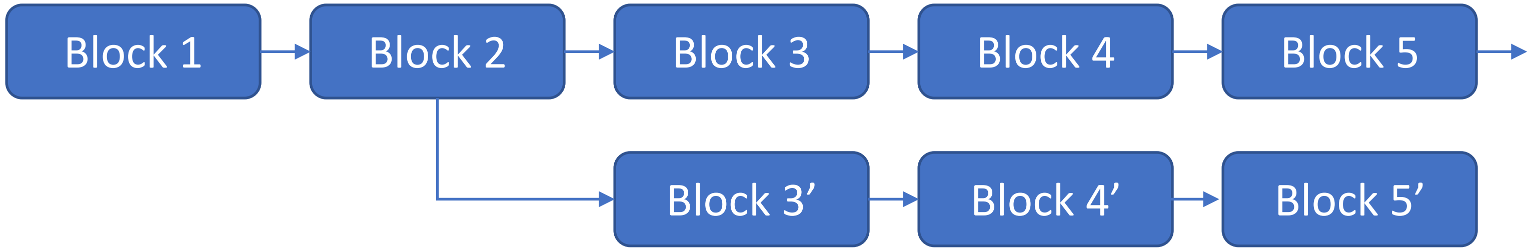
B_i = Block i

T_i = Transactions in Block i

- If a block is changed, all future blocks must be changed:
  - Each block contains a hash of the previous block, so the hash must be changed
  - This means the proof of work must be changed too
  - The more time since a transaction has been recorded, the more secure it is

How does Proof of Work prevent the delete problem?

- Suppose 30 minutes later, Bob publishes a fake ledger removing his transactions, to claim that he received no money for his services



- All participants accept the longest path: In order to convince everyone that his fake ledger is real, Bob must find at least 3 proofs of work
- However, unless he has >50% of the total computational power of all miners, the other miners will find more than 3 proofs of work in the time being (Blocks 6, 7, 8…) – and he will never catch up
  - This also implies that someone with >50% computational power of everyone could break the blockchain (“51% attack”)

Mining

- Why should miners provide a proof of work?
  - Each proof of work earns the miner a certain number of bitcoins
  - Currently: 6.25 bitcoins, halves every few years – to ensure that there is a maximum number of bitcoins in total (21 mil)
  - The difficulty of the problem is automatically adjusted so that it is expected that one block should take 10 minutes
- Miner algorithm:
  - Generate random guess of proof
  - Test if proof works with current block (hash it)
  - If yes, publish; if not, generate new random guess of proof

Step by Step

1. Alice sends Bob some Bitcoins. Alice signs this transaction with the private key associated with her account and Bob broadcasts the transaction and signature.
2. Miners verify the signature and include it in their transaction ledger.
3. Miners attempt to hash the new transaction ledger appended with a proof of work to arrive at a specific output (e.g. hash output must have 32 zeroes at the end). (Other transactions may also be included when they come in.)
4. When miners (any miner) finds a valid hash, they broadcast the proof of work as a new Bitcoin block, the system rewards them with Bitcoins, and the new block of transactions is downloaded by all miners to update the transaction ledger.

How to prevent the addition problem?

- Why should miners include your transaction?
  - In fact, it is easier for miners to ignore transactions, so they can generate proof of work repeatedly on the same object
- To solve this problem, a transaction can also include a “transaction fee” that is automatically obtained by any miner that includes it
- The transaction fee is effectively independent of transaction amount
- At most about 7000 transactions can be included in one block (size limits), so transactions with no fees or lower fees might be ignored

Transaction problems

- 7000 transactions per block is about 10 transactions a second globally
  - This is many magnitudes lower than current financial systems
  - But more transactions per block = larger communication cost
- Fees are relatively high for small transactions; impractical for coffee, etc.
- Transaction delay: it is best to wait for several blocks (30+ minutes) before considering a transaction confirmed
  - Even non-maliciously, it is possible other miners will find different proofs of work and follow their own ledgers until a clear winner has been selected
- ASIC mining means there is strong motivation not to change the protocol

Privacy problems

- A global transaction ledger is the least private protocol possible
  - All senders, receivers, and transactions amounts can be tracked by anyone
  - Although it does not necessarily reveal real names, transactions are nevertheless linked to the same account
  - Starting and maintaining multiple accounts is not easy and still traceable
- Can send bitcoins to a “mix” who will send them to your other account – but this is risky and costly
- No forward “secrecy” – a lost private key is compromised forever
  - The corresponding public key is a person’s identity

Zerocoin

- We will leverage the global transaction ledger to solve these problems with zero-knowledge proofs
- Intuition: If Alice can prove that she has money, then people should accept her spend transaction for that amount of money
- When this proof is done with zero-knowledge, she can hide how she obtained that amount of money
- Several components:
  - **Minting**, creating zerocoins
  - **Spending**, using ZKP to prove she has money
  - **Verifying**, verifying the proof is correct

Minting

- Each zerocoin c will have a different serial number S; let us say each zerocoin is worth 1 bitcoin
- c is a cryptographic commitment of S:

$$c = g^S h^r \bmod p$$

- S and r are kept secret, g, h and p are public parameters
  - S will be revealed later during spending
- A bitcoin transaction is recorded in the global transaction ledger: "Alice spends 1 bitcoin to create c"
- After it is processed, Alice has 1 less bitcoin but can use the zerocoin spend function to spend the zerocoin

Spending

- Reveal the serial number S and provide two ZKPs:
  1. I know a certain *c* in the set of all zerocoins C
  2. This c satisfies

$$c = g^S h^r \bmod p$$

- Proving a certain *c* in a set of all zerocoins C uses an **accumulator**, which works as follows:
  - For a set of prime numbers $C = \{c_1, c_2, \ldots, c_n\}$, the accumulator is $A = u^{c1c2\ldots cn} \bmod N$; this is publicly known
  - The "witness" that c_1 belongs to C is $w(c_1) = u^{c2\ldots cn} \bmod N$; similarly for any other element
  - To verify, check that $w(c_1)^{c1} = A \pmod N$
  - This way, it is very easy to add or remove from the accumulator

Verifying

- A verifier needs to check that:

1. All proofs are correct;
2. The serial number revealed has never been used before.

- Since proof verification of ZKP is slow, we can have different nodes verify different iterations of the proof
- Clients must store a list of all serial numbers revealed to be able to verify this
- Refer to [Miers 2013] for full ZKP

Further improvements

- Two main weaknesses of Zerocoin:
  - ZKP validation is inefficient (not comparable to Bitcoin)
  - Privacy weakness: transaction amounts and targets are still revealed (only source is hidden)
- Zerocash fixes both of these issues with better ZKP techniques

Presentation

- A good-quality presentation should have:
  - A clear and simple style
  - High priority on important and interesting findings
  - Clever use of diagrams, figures, and analogies to explain complicated concepts
  - Practiced speech
- It is easy to express effort in presentations
- Don’t forget the main objective – explaining things

Writing a Paper

- A good-quality presentation should have:
  - A clear and simple style
  - High priority on important and interesting findings
  - Clever use of diagrams, figures, and analogies to explain complicated concepts
  - Practiced speech
- It is easy to express effort in presentations
- Don’t forget the main objective – explaining things

Exam

- 40 Multiple Choice questions, 2 hours
- Similar style to quizzes
- Open-book is allowed
    - However, using the Internet except to access the Zoom meeting and the quiz are not allowed