ECE 443/518 – Computer Cyber Security
Lecture 15 Consensus

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

October 19, 2022

Outline

Reading Assignment

- This lecture: Consensus
- Next lecture: Cryptocurrency

Outline

Consensus

- Consensus: how can multiple parties reach agreement?
  - E.g. to ensure there is a single branch for data management.
  - Assume some parties and communications could be faulty.
  - A fundamental problem of distributed computing.
  - A security problem as arbitrary faulty behavior is allowed.
    - One must consider possible attacks by participating parties.
- An example: each party presents a value of 0 or 1, and together they want to agree on the majority.
  - What faulty behavior can you think of?

The Byzantine Generals Problem

- A recast of the previous example by Lamport et al. 1982.
- Not related to any historical events.
- But in a more realistic setting for people to reason with assumptions and possible attacks.
- People now use 'Byzantine Fault Tolerance (BFT)' to refer to the consensus problem assuming arbitrary faulty behavior.
- Note that there is no confidentiality issue.

The Byzantine Generals Problem (Cont.)

- There is a group of Byzantine generals.
  - Each commands a division of Byzantine army.
  - Together they encircle an enemy city.
- The generals observe the enemy and then decide if they should attack or not.
  - Each general first makes a decision of attacking or not.
  - Then together they vote and follow the majority.
  - They could further agree on what to do if there is a tie, say not to attack.
- The generals cannot leave their divisions – they can only communicate pair-wise using messengers.
- We only care whether the consensus is reached or not – we don't care if they actually attack or not.

Traitors

- However, some of the generals are traitors.
  - Traitors do whatever they want.
  - Traitors may collude.
- The objective of the traitors is to break consensus.
  - E.g. trick some loyal generals to believe that majority plan to attack, while trick other loyal generals to believe that majority plan not to attack
  - In other words, we don't care if traitors trick all loyal generals to attack or not to attack, but care about the case when at least one loyal generals decide to attack and at least one loyal generals decide not to attack.
- Protocol design: a protocol all loyal generals follow.
  - So that they will reach a common decision.
  - Assume there are at least 2 loyal generals, how many traitors could there be at most?

Messengers

- Let's assume that all messengers are authentic.
  - They will identify the generals sending the messages and will not modify the messages.
  - We could always replace a general sending out treacherous messengers with a traitor sending out authentic messengers.
  - This requirement may be relaxed later.
- Let's further assume all messengers are available.
  - Traitors prefer not to block communications as they want to trick some loyal generals to attack.
  - Reasonable for practical networking as long as adversaries do not control large portion of the network.

Example: with Prior Knowledge

- Alice and Bob are loyal generals, and Oscar is a traitor.
- Traitor may break consensus for naive protocol.
  - Alice: attack
  - Bob: do not attack
  - Oscar tells Alice: attack
  - Oscar tells Bob: do not attack
  - Alice will attack and Bob won't if they just follow majority.
- Could Alice and Bob talk to reveal Oscar as a traitor?
  - Alice tells Bob: Oscar said "attack"
  - Bob tells Alice: Oscar said "do not attack"
- But Alice has no prior knowledge that Oscar is a traitor.

Example: the Actual Case

- Alice has to decide who are traitors only from messages.
- Messages received by Alice
  - Bob: do not attack, Oscar said “do not attack”
  - Oscar: attack, Bob said “attack”
- Alice can’t make a decision.
  - Either Bob or Oscar is traitor (or both).
  - Should not attack if Bob is not traitor.
  - Should attack if Oscar is not traitor.
- Will it help if we ask everyone to forward whatever other people said for multiple rounds?
  - E.g. Alice said “Bob said “Oscar said “Alice said ...”””.

Some Results

- If multiple rounds are allowed, for $3m + 1$ generals, there is a protocol to cope with at most m traitors.
- It can be proved no protocol can cope with more traitors.
  - E.g. 1 in 3 as our Alice/Bob/Oscar example.
- With the assumption that no one can prove what other people actually said.
  - We know nonrepudiation can be achieved by digital signatures assuming computationally bounded adversaries.
  - So this assumption can be removed and we can get stronger results with digital signatures.

The Protocol using Digital Signature

- All loyal generals sign their messages using their own public/private key pairs.
- Traitors do whatever they want.
  - For their best interests, they should still sign their messages using their own public/private key pairs.
  - But they could collude by knowing each other's private key and then forging each other's signature.
  - Though they cannot forge loyal generals' signatures.
- The protocol runs $m + 1$ rounds to cope with at most m traitors among any number of generals.
  - Every (royal) general first broadcasts his/her own decision.
  - For the remaining m rounds, every (royal) general broadcast what he/she received from the previous round.
  - Assume royal generals set timeouts for rounds to prevent traitors to sabotage the protocol by being silent.
- There is no need for messengers to be authentic now.

Example: Using Digital Signatures

- 2 generals A and B, 2 traitors C and D.
  - Since C and D cannot forge A and B's signature, A and B will know each other's decision.
- Can C/D collude to trick A/B to think differently of C?
  - Possible for 2 rounds but not possible for 3 rounds.
- First round: C tells both A and B "attack".
  - Note that each quoted message is signed by who says it.
- Second round
  - To A, B says "C said "attack"", D says "C said "attack"".
  - To B, A says "C said "attack"", D says "C said "do not attack"".
  - A has no reason to see C as a traitor, but B will see C as a traitor since B sees two different values signed by C.
- Third round
  - B tells A "D said "C said "do not attack""".
  - Now A will also see C as a traitor.

Practical Limitations

- By using digital signatures, we are able to reach consensus in $m+1$ rounds with at most m traitors.
- How many traitors are there if we would like to have consensus across a cybersapce?
  - There is no trusted third party.
  - Therefore digital signatures as identities cannot be connected to any physical identities.
  - Adversaries can generate as many identities as they would like – m could be arbitrarily large.
- In addition, when m is fairly large, the cost of $m+1$ rounds of communication would be prohibitive.
- Can we actually reach consensus?
  - What if we assume there is enough incentive for people to collaborate?

Outline

History

- Digital cash (Chaum 1982)
  - Cash instead of credit card: the need to protect payer's privacy.
  - Also allow payer to identify payee or to report lost cash.
- Hashcash (Back 1997)
  - Proof-of-work postage to limit email spam.
- B-money (Dai 1998)
  - Currency instead of cash: value of money depends on the effort to solve a previously unsolved computational problem.
  - Use ledger to record transactions.
  - Use distributed and trusted servers instead of banks.
- Bit gold (Szabo 1998)
  - Realize that gold (and other precious metals) is money without trusted third parties in the majority of human history.
  - Suggest to use BFT protocols to cope with dishonest servers.
- Bitcoin (Satoshi Nakamoto 2008) and beyond
  - But who is Satoshi Nakamoto?

The Ledger

- It is difficult to use binary strings to represent money directly.
  - They are easily copiable – one has to know which strings are issued for money but not used.
  - Also one would like money to be transferable.
- The ledger
  - Money is associated with account.
  - Transactions between accounts are recorded.
  - Account balances can be computed from relevant transactions.

Accounts

- Use random numbers instead of a sequence id.
  - To determine the next available number from a sequence is a consensus problem and thus requires BFT – too expensive.
- Proof of ownership
  - But anyone can claim ownership of a number if we allow arbitrary random numbers.
  - Use public-key cryptography: one claims an account number if that number corresponds to his/her public key.
  - The account remains anonymous as long as the public key is not associated with any physical entity.
- Implications
  - Not all account numbers are valid.
  - Initial account balance should be no more than 0.
  - You own the account only if you know the private key and nobody else knows it – your private key worth as much as your account.

Transactions

- Transactions are proposed by payers, each includes
  - Payee account.
  - Amount of transfer, possibly with transaction fee.
  - Additional data.
  - Signed by payer.
- Payee accepts a valid transaction automatically.
- Transaction validation.
  - Verify signature using payer's account number.
  - Amount of transfer should not be negative.
  - Amount of transfer should be no more than payer's balance.

Ledger Data Management

- At least provide integrity and nonrepudiation.
  - Use Merkle hash tree
- The blockchain
  - Block: a Merkle hash tree containing all recent transactions to be executed.
  - Chain: another level of Merkle hash tree that links all blocks and thus all historical transactions as the ledger.
- No adversary can spend your money given
  - They don't know you private key to sign transactions.
  - All transactions are valid in the blockchain.
  - The integrity of the blockchain can be validated.
- Is this secure enough?

Double Spending and Branches

- Double spending
  - Oscar has 100 on his/her account for the most recent block X.
  - Oscar pays Alice 100 by creating a block Y based on X.
  - Oscar pays Bob 100 by creating a block Y' based on X.
  - Alice (Bob) happily accepts Y (Y') without knowing Y' (Y).
  - But when Alice need to pay Bob, they know they are cheated by Oscar since the history diverges.
- There need to be a consensus on which block is the next.
  - Branches are not allowed in this application – the Merkle hash tree should be a chain.
  - If it is Y, then Bob should reject Oscar's request.
  - If it is Y', then Alice should reject Oscar's request.
- Need BFT protocols.
  - That can work with any number of adversaries efficiently.

Summary

- Consensus and Byzantine Fault Tolerance (BFT).
- Cryptocurrency depends on proper BFT algorithms to function without trusted third parties.