Distributed Settlers of Catan

Hassan Alsibyani, Tim Mickel, Willy Vasquez, Xiaoyue Zhang
Massachusetts Institute of Technology

May 15, 2014

Abstract

Settlers of Catan is a popular multiplayer board game that involves dice rolls, drawing cards from shuffled decks, secrecy, and agreements. This paper will first outline a fair protocol for playing Settlers of Catan with no trusted third party or shared resources, and then discuss a possible Python implementation. The protocol assumes that players are connected to each other via the internet, and that the connection is reliable. The protocol minimizes the effect of collusion between players, is able detect cheating and cheaters, and guarantees the privacy of players' hands.

1 Introduction

There are many reasons for making a game distributed. Playing a distributed game is the only option when physically playing the game is not possible, and when there does not exist a trusted third party to oversee and direct gameplay. Making a game distributed can be useful even when a trusted third party does exist, especially in the context of internet gambling. A distributed protocol is completely transparent and its guarantees can be proven or disproven, but it is difficult to know how a third party functions and whether or not it could be trusted. In fact, gambling servers are often shut down by the FBI due to fraud and illegal activities [1], so not only would distributed gaming be more trustworthy, it would be more reliable. A central gambling game server is also a much more coveted prize for hackers than a single instance of a distributed game. A breach in security of the former can affect many more players, and result in much more losses. There is far less incentive to hack an instance of a distributed game because not all information is hosted on a single instance. Furthermore, gambling servers often charge a large fee, known as the rake, for using their services.

Distributed gambling allows for players to receive all of what players put in, without needing to pay rake to a virtual casino.

1.1 Settlers of Catan

There has been some research done on distributed gaming, starting with Mental Poker, where Shamir, Rivest, and Adelman describe a protocol for dealing random hidden hands to just two players [2]. Since then, others have generalized the protocol to handle more players [3] and made performance improvements [4]. There is no literature for making specifically mental Settlers of Catan.

Settlers of Catan is a resource management board game typically played by 2-4 players. The rules for the game can be found here [5]: `http://www.catan.com/service/game-rules`. In summary, the initial board needs to be generated randomly, players have secret hands of resource cards, players gain resources according to dice rolls, and players may trade resources with each other. Players may also draw development cards from a deck and reveal them at their will. A player may be a robber for a turn and take a random card from another player's secret hand. Protocols for each component of the game are given in section 4.

Since our protocol is designed with internet gambling in mind, we need to minimize the effects of colluding, ensure that players who do not follow the protocols do not have an advantage over players that do, identify cheaters, and guarantee that information about resources and cards that should be secret is kept secret; section 2 discusses our guarantees in detail.

Our threat model is also based on that of internet gambling. An adversary may try to cheat and eavesdrop, section 3 describes the adversarial model.

We decided that simply identifying cheaters rather than preventing them is enough because our protocols ensure that they cannot ruin gameplay. section 5 discusses how cheating is detected or handled.

section 6 discusses some challenges we faced when creating protocols for playing the game. Two major challenges were the multiplayer aspect of Settlers, and performance.

As a proof of concept, we have created a python implementation to showcase how our protocol works, section 7 describes our implementation, and discusses the security and shortcomings of each component.

2 Guarantees

Our goal is to keep gameplay as close to the physical game as possible while maintaining important security parameters. Our protocols will satisfy three main requirements:

- **Validity:** Any strict subset of players should not have inappropriate control over game play.
- **Cheating Detection:** The ability to detect rule violation, or cheating, and the cheater by the end of the game.
- **Confidentiality:** Any player should not have unauthorized knowledge of the state of the game.

We cannot preclude all collusion amongst players. In traditional Settlers of Catan, collusion occurs. For example, there is no way to stop player A from showing player B her hand. However, there are elements of the physical game that do not allow for collusion, such as shuffling cards and rolling dice. Our algorithm will ensure that no single player has inappropriate control over those elements of the game. In fact, it will ensure that as long as there is a single fair player, dice rolls and card shuffling will be fair and seemingly random.

We decided that simply identifying cheaters rather than preventing them is enough as long as the repercussions of cheating are severe enough. We will use a log mechanism that views all transactions that can later be used to catch a majority of the instances of cheating at the very end of the game. Every player will be able to verify on their own that every single play in the game is valid, e.g., if someone bought a road with a wood and a brick, they actually had wood and brick resources in their hands and discarded them afterwards.

If one paid attention to the dice rolls in a traditional game of Settlers of Catan game, one would know a majority of the resources of every player. However, trades, development cards, and the robber add secret or random components to the state of the game. The privacy requirement does not necessarily guarantee that hands are kept secret, it just guarantees that no information not deducible from the public components of gameplay is leaked while exchanging messages.

Another guarantee that we make is that there is a reliable lossless connection between each player and no player leaves prematurely. Everyone remains connected to the game up to completion.

With this set of guarantees, it is impossible for an adversary from our threat model to gain an advantage in the game.

3 Adversarial Model

Our threat model is based on threats we expect to encounter in internet gambling. The goal of an adversary is either to win, to not lose, or to cause another player to win/lose. Our protocols are designed to be resistant to the following threat model:

- An adversary who does not follow the protocol in an undetectable way. For example, the adversary may be trying to affect a dice roll to be a particular number, so they do not use a random number when all players collectively roll the die, as our protocol requires. We need to ensure that such an adversary does not gain an advantage in the game.
- A passive eavesdropper adversary. Such an adversary is capable of ciphertext-only attacks. We assume that the channel between players is lossless but not secure.

4 Protocol

In this section, we will discuss protocols for each component of the game. We will also discuss why they satisfy our guarantees given our threat model.

Shuffling by an individual player will be used in many of the protocols. The algorithm we use to perform a shuffle is the Knuth shuffling algorithm [6]. The Knuth shuffling algorithm assigns a distinct number between 1 and n to each of the n cards. Then it generates a permutation by choosing numbers uniformly at random from the set of remaining numbers, creating a permutation that defines a shuffle.

4.1 Game Setup

Many of the protocols depend on an ordering of the n players. We will use a *canonical ordering*. When we say, player A sends player B a message, we actually mean that player A broadcasts to all players a message addressed to B. This will be useful for detecting cheating at the end of the game. For communicating between players, a signature scheme should be used as well. This would prevent malicious players from acting as another player or disputing their own previously sent messages. Thus, public key exchange needs to occur before the game.

4.2 Board Initialization

The Settlers of Catan board requires randomly tessalating 19 board tiles in a hexagon and then randomly distributing 18 chips on the tiles, with one chip on each tile except for the desert tile. Once an order for the cards and chips are determined, a unique board can be created by placing the tiles counter-clockwise in the tessalation.

1. Represent the board tiles as distinct numbers between 1 and 19.
2. Each player will generate their own AES key, encrypt what they "receive" using AES, and shuffle the encrypted list elements. Then the player passes the list to the next player in the canonical order created in the beginning of the game. The first player "receives" the unencrypted numbers 1-19.
3. Each player would broadcast their AES keys and random permutation, and every player would decrypt the deck in the reverse order it was encrypted in.
4. The final order of the deck would be the one used for creating the board.

Note that a single honest player would render all colluding groups ineffective, and the validity guarantee is satisfied. The other guarantees are irrelevant at this stage.

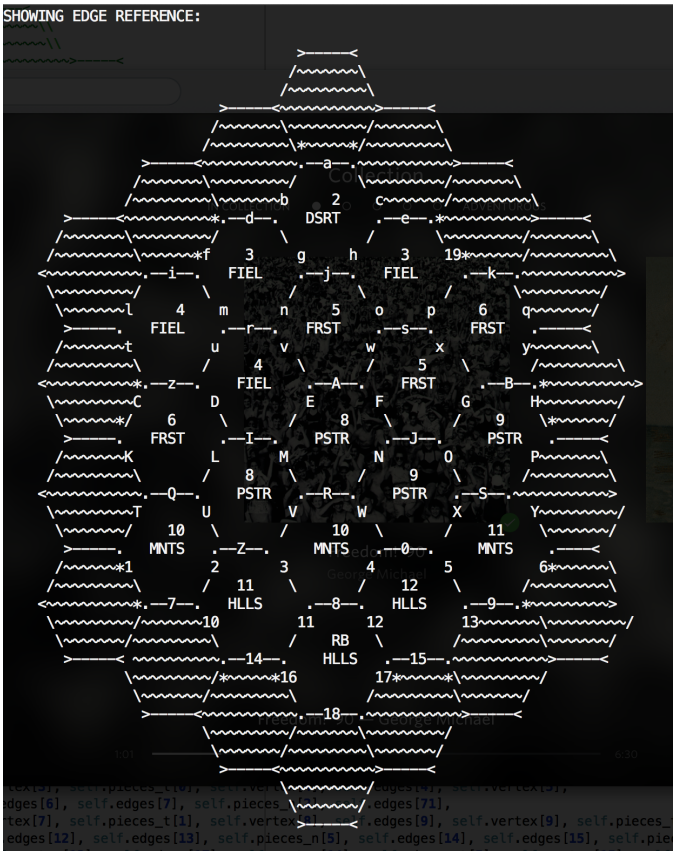

Figure 1: The view of the generated board after the above process. As you can see each hexagonal tile has an associated number and resource.

4.3 Dice Roll

Each turn in Settlers of Catan requires the roll of two dices. The rolls should be random. We divide the dice rolls into three phases.

1. Each player would choose a random 64-bit number R_i and then broadcasts SHA256(R_i).
2. Each player would now broadcast R_i.
3. Each player should verify that the other players' hash commitments matches their random R_i's. Afterwards, Each player would calculate $D = \text{SHA256}(\text{turnNumber } |R_1|R_2|\ldots|R_n)$. The dice roll would be $((D \mod 6) + 1)$.

The breaking of the dice roll in three phases allows for commitment. Each player would be commited to the choice of the random number and would allow for change to allow an advantage after seeing other players' random numbers. The hash function, SHA256 in this case, should produce pseudo-random numbers. This means that even if $n-1$ players were colluding to produce a specific dice roll; one independent truly random player is enough to maintain the randomness of the dice roll. Thus, the validity guarantee is satisfied.

4.4 Development Cards

Players may buy developement cards on their turn. The buyer takes a face down developement card from a shuffled deck of cards. A developement card can be kept secret for however long the owner chooses before being revealed and played.

1. Begin with a deck of cards represented as distinct strings.
2. In the predetermined order, every player encrypts each card using AES and shuffles the deck.
3. When a player buys a development card, all other players reveal their keys for that card. The player could then decrypt the card to read what it is.
4. To reveal a card, the player reveals his key as well, so everyone can completely decrypt the card.

The shuffling mechanism is very similar to that described in board initialization. Again, as long as there is one honest player, the cards are shuffled, and validity is satisfied. AES allows confidentiality to be satisfied. Only encrypted strings are passed in messages up until the card is finally revealed.

4.5 Gaining Resources and Managing Hands

Sometimes, a player is eligible to gain resources from the bank. Everyone is able to see what resources he gained. However, it should be the case that which card would be in his hands is still unknown.

1. If a player k is eligible to take a resource card R_i, then R_i would be broadcasted.
2. Player k then adds the card to his hand.

The player does not need to encrypt the hand or the card he just gained. Encryption is only needed to mislead other players in the game when they are trying to take a card from a specific player. Until this action is needed, there is no advantage of generating keys and encrypting a player's hand.

4.6 Trading with Players and Bank Management

During player A's turn, player A is allowed to negotiate and perform public trades with other players.

1. Suppose player A decided to trade with player B, then each player would broadcast the trade details.
2. Players A and B would post their private keys for the traded cards and everyone would verify the trade.
3. Players A and B get to keep the traded cards.

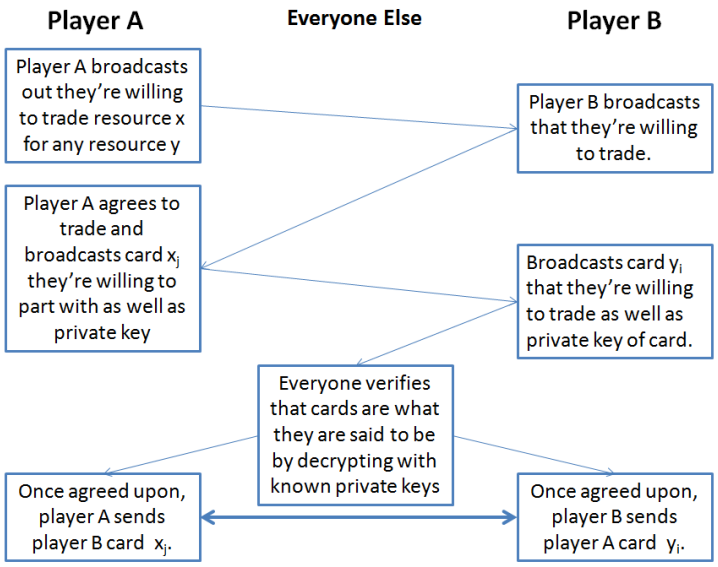

Figure 2: How trading is performed within our protocol between players A and B

This scheme also works if one of the players is the bank. The bank would just be a passive trader. In addition, this would also work if the Player is paying the bank in order to build houses and settlements.

4.7 The Robber

If a player A gained control of the robber, then that player can use it to steal a card at random from a player B of his choice.

1. Player A would need to choose a Player B to steal from.
2. Player B would re-encrypt and shuffle his hand with fresh keys.
3. Player A would choose a card from player B's encrypted hand.
4. After he makes his choice, Player B is required to give the key for that card to Player A for decryption.
5. Player A then gets to keep the stolen card.

This is the only time where a player needs to encrypts his hand to distort the stealer choice.

5 Cheating and Cheater Detection

In the end of the game, a log with all the game moves along with players' current hands is published for all players to review. Every player would need to publish the private key for all of their current hands. After that, each player will have the opportunity to review each trade of the game and insure that they are accurate in regards to events witnessed during gameplay.

The log file is verified manually by each player before a victor is announced in order to prevent a cheater from winning. This manually checking insures that we have an agreed upon winner from all parties by majority votes.

6 Challenges

A major challenge that we faced was making the protocols work for more than two players. In the two player case, one can assume mutual distrust.

However, with added players, the problem of collusion arises. Another challenge that the multiplayer component adds is in the message passing. Messages in all the protocols are broadcasted to everyone unless otherwise specified, to ensure creating a consistent log of the game at the end. However, Alice may send Bob one message and everyone else another, when she claims she broadcasts the message to everyone, and it would be difficult to tell if Alice is cheating or Bob is cheating.

Another big challenge we faced is in performance. We realize that our protocol would lead to very slow implementations of distributed Settlers of Catan, because the frequency we generate keys and because of the initial sequential shuffle and encrypt process. We tried to minimize key generation and shuffling the best we could, but we did not have the time to come up with methods to drastically improve performance.

Lastly, we found defining our goals, assumptions, and the adversarial model to be challenging. There needed to be a balance between achievable, realistic, and interesting. We ended up needing to make much stronger assumptions than we started out with.

7 Implementation

We have completed an initial implementation [7] of several of the discussed protocols, coming together in an almost-playable distributed Settlers of Catan game. The implementation has been completed using Python; cryptographic functions including AES and SHA256 are provided by the mature Python cryptography library PyCrypto [8].

7.1 Game setup

At the beginning of a game, the player specifies which port to run a server on their local address. Players begin the game by providing the game a list of other players and the ports at which their running servers are. For example, if there are three players, each user will provide the game with the hostnames and ports of each of the other two players. The client attempts to connect to each other player; once a successful connection has been negotiated with each other player, they begin exchanging messages. The player chooses a unique ID for itself and broadcasts that to the other users. The unique ID is used to identify messages and also to determine initial turn order for board negotiation. Messages in our implementation are JSON-encoded dictionaries with commands and other useful information for executing a turn or otherwise negotiating an event. The players would use an RSA

signature scheme to communicate with each others. This is to insure that no player is impersonating another player, and also hold a player accountable to their own messages.

7.2 Board negotiation

Board negotiation is the first event to happen in the implementation. The player with the lowest unique ID generates a full board (including all the board pieces and corresponding tokens), shuffles them using Python's standard list shuffler, encrypts all the pieces, and broadcasts the board to all of the players. This is repeated turnwise until everyone has encrypted and shuffled the board in order. Encryption occurs using AES in CFB mode; the IV chosen is random. As explained in our outline for this protocol, if at least one player shuffles and encrypts the board honestly, at the end, the unencrypted board should be fairly shuffled. The order of encryptions and shuffles also does not matter because of this. When each player has had their turn, they each broadcast the keys and everyone agrees on the board.

7.3 Turn order and dice roll, building

Similarly, our die roll protocol is fully implemented in our demo. The players together roll fair dice through message passing; they choose a random 64-bit value, commit to it to all of the other players, then reveal these values. Using our protocol, they calculate the die roll. In our implementation, die rolling is fairly slow because we perform a large MOD calculations (hash mod 6), but it takes only a few seconds which is reasonable for the interactive game. Using dice rolls, the players take turns (in UID order) to determine the player who should go first in the game.

Once turn order has been established, players are given the option to build their houses and roads. A nontrivial part of our implementation is representing the board and ensuring that players are placing pieces in legal locations; a player broadcasts their intended move to the entire group of players and if any of them determines that the placement was illegal, a notification is raised.

7.4 A turn, resources

Our implementation also has a fully-working resource bank and resources are distributed to players upon dice roll. The player who rolls a number 7 is also permitted to move the robber, which has an impact on how resources are distributed. Each player tracks each other player's resource allocations

for verification at the end of the game. The resource bank is implemented as a large list of numbers, for example, with 1-200 representing wood, 201-400 representing sheep, and so on, giving each resource card a unique number.

7.5 Future work and comments

Due to time constraints, our implementation is not fully playable at this point; we have yet to implement our proposed trading protocols, the development cards, and the end-of-game cheater detection as described above. However, we believe all of these features are completely possible to implement within the framework that we have built for the game. As it is, it can be very fun to watch the game negotiate cryptographically with other players, which was the goal and intention of our implementation.

8 Conclusion

We have demonstrated a distributed version of Settlers of Catan that is able to guarantee validity, cheating detection, and confidentiality. Our total protocol implements the essential parts of Settlers of Catan such as board generation, dice rolling, player trading, and development cards. We have also created a Python implementation as a proof of concept of our protocol that is not yet fully playable but given more time could be.

References

[1] Nagesh, G. 2011. FBI Shuts down Poker Sites in Major Online Gambling Crackdown. The Hill. http://thehill.com/policy/technology/156429-fbi-shuts-down-online-poker-sites

[2] Shamir, A., Rivest, R.L., Adleman, L.M. 1981. Mental Poker. The Mathematical Gardner:37-43.

[3] Zhao, W., Varadharajan, V., Mu, Y. 2003. A Secure Mental Poker Protocol Over The Internet. Australasian Information Security Workshop (AISW2003).

[4] Golle, P. 2005. Dealing Cards in Poker Games. Proc. of ITCC 2005 E-Gaming Track.

[5] Settlers of Catan rules: http://www.catan.com/service/game-rules

[6] Knuth, Donald E. (1969). Seminumerical algorithms. The Art of Computer Programming 2. Reading, MA: Addison–Wesley. pp. 124–125.

[7] Crypto-settlers implementation, Github. https://github.com/tmickel/crypto-settlers

[8] PyCrypto - the Python Cryptography Toolkit. https://www.dlitz.net/software/pycrypto/