Cryptographic Dark Chess

Jimmy Ou Yang

Supervisor: Claude Crépeau

April 12, 2024

1 Introduction

Dark chess is an incomplete-information variant of chess. Unlike regular chess, the players of dark chess do not see the entire board nor do they see all of the opponent's pieces. Dark chess is typically played with a third party with full information of the game, usually a referee or an online server, who actively oversees the game. One might then ask: can two people play a game of dark chess without an active third party?

2 Dark Chess

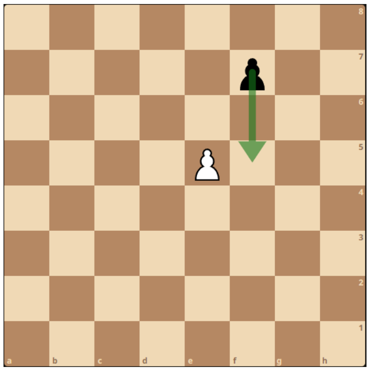
Dark chess plays very similarly to regular chess, with some differences to what each player sees, what moves are allowed, and the goal of the game.

- Each player can only see their own pieces and the squares that they can move to. Squares that cannot be seen are "dark".
- A pawn only has partial vision of the square directly in front of it. The player can see that the square is empty or occupied, but not the piece occupying it.
- A pawn has "conditional" vision of the squares diagonally in front of it. If there is an opponent piece on the square, then the player will see that piece. Strictly speaking, if the square is empty, then the player should see the square as dark. However, the player can deduce that the square is empty in this case. For this reason, the player can "see" the square and this type of vision can be treated as complete.
- Capturing a pawn *en passant* is allowed. During a turn where *en passant* is possible, the capturing player can see his pawn's landing square (front diagonal) and the opponent's threatened pawn (adjacent). The player loses this vision on the next

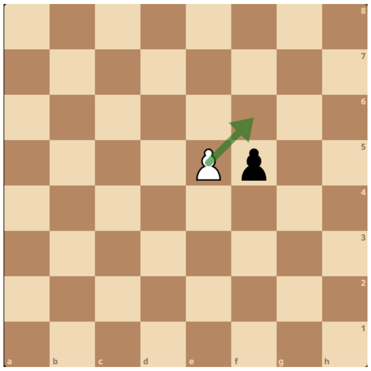
turn whether he captures the pawn or not.

- There are no checks or checkmate, as it is not necessarily the case that a player even knows that his king is in check. A king is then free to stay in or move into a checked position. It may also castle through would-be checked squares.
- A player wins by capturing the opponent's king.

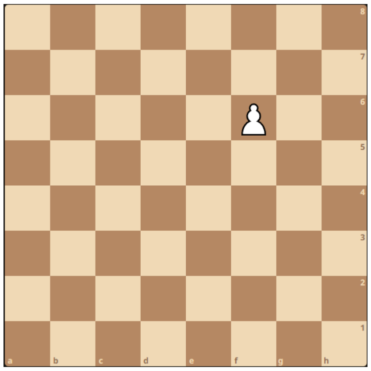
En passant



White to play



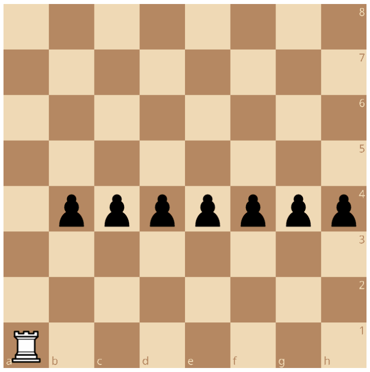
Black to play



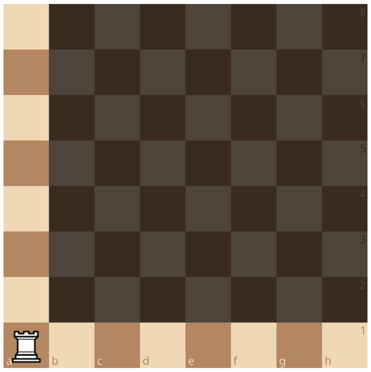
Pawn is captured

3 The Challenge

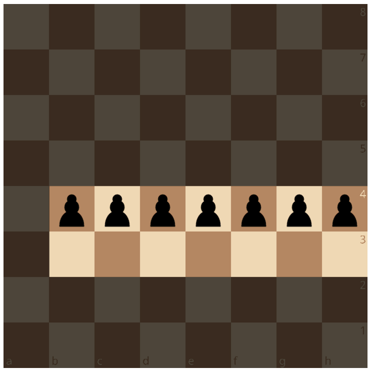
The difficulty behind building a protocol for dark chess comes from the fact that the players do not actually know what they should know, and they cannot let the opponent know what they know (or do not know). Take for example poker and Battleship, two other games with imperfect information that have received the attention of cryptographers [GKWZ, Cr 86, Cr 87]. In Battleship, both players start with no knowledge of the opponent's fleet. When a player fires a shot, he learns whether his shot hit or missed. The receiving player learns where the first player shot, and he knows that his opponent knows if the shot hit. In poker, when a player draws a card, he knows that only he knows the drawn card. When a player opens (reveals) a card, he knows that everyone now knows his card. In both of these games, each player knows what information the other players possess (eg Player 1 knows that Player 2 knows X, Player 2 knows that Player 1 knows that he knows X, and Player 1 knows that Player 2 is aware that he knows). Now, consider the following dark chess position, where it is white's turn to move:



Full board

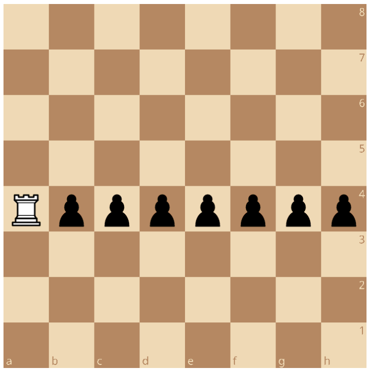


White's view

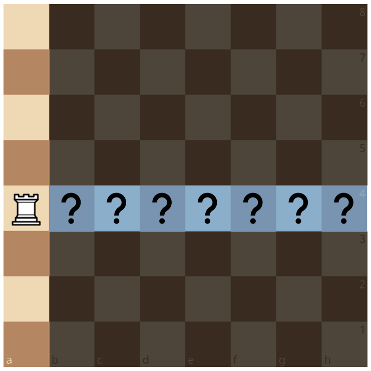


Black's view

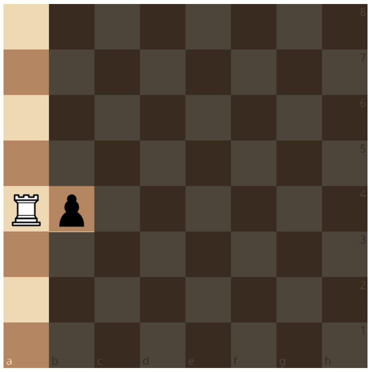
White plays ♖a4.



New full board



White's potential view



White's actual view

After his move, white does not immediately know which squares of the 4th rank he should be able to see. Similarly, black has no information on white's position and also does not know what white should see. In a case without a third party, white needs to somehow learn of the black pieces from black himself without revealing what he learned. Additionally, black wants to ensure that white does not get more information than he should. As we will find out, these restrictions are quite non-trivial.

4 Cryptography Background

Oblivious Transfers

First introduced by Rabin [Rab81], an oblivious transfer (OT) is a two-party protocol that allows a receiver to receive some piece of information from a sender, without the sender knowing what information has been sent. There are many variations of OT protocols, but we will focus on k-out-of-n oblivious transfers. In particular, we make use of the All-or-Nothing Disclosure of Secrets (ANDOS) protocol introduced by Brassard, Crépeau and Robert in [BCR86] which allows a receiver to choose and obtain k of the n secrets from a sender.

Adversaries

Adversaries in crypto protocols can have different levels of power and exhibit various behaviors. In terms of power, we only consider adversaries that have bounded computational power. Any computation that they do must be computable by a probabilitic polynomial time algorithm (PPT), that is, an algorithm that runs in polynomial time and that has access to a source of randomness (ie it can flip coins). The adversary can be corrupt in different ways, which decides how it behaves. For our purposes, we will consider semi-honest (or honest-but-curious) adversaries and fully dishonest (or malicious) adversaries. In the semi-honest setting, the parties do not deviate from the protocol but will try to learn more from the information they have. In the malicious setting, the parties can deviate from the protocol.

5 A First Protocol

This first rudimentary version of the protocol will have the following assumptions:

- Players P_W and P_B want to play a game of dark chess.
- The players are honest but curious. They play by the rules of the game and do not deviate from the protocol, but they will try to learn more information from what they are given.
- The players are computationally bounded.
- The players are smart. They understand all rules and nuances of the game.

5.1 Board Representation

Let $RANK = FILE = \{1, 2, .., 7, 8\}$ and assume a correspondance between the set $\{a, b, ..., g, h\}$ and $FILE$.
Let $PIECE = \{Empty, Pawn, Knight, Bishop, Rook, Queen, King, Unknown\}$.
Let $COLOR = \{W, B\}$.
Let P_c where $c \in COLOR$ be the player playing color c.
Each P_c keeps track of 64 squares.

Definition 1. A square is represented as

$$S_{c,i,j} = \langle c', p \rangle$$

where

$c \in COLOR$ is the color of the player viewing the square,

$i \in FILE$ is the file (column) coordinate,

$j \in RANK$ is the rank (row) coordinate,

$c' \in COLOR$ is the color of the piece,

$p \in PIECE$ is the type of the piece.

When $p = \mathit{Empty}$ or $p = Unknown$, c' does not have a meaning, but for the sake of convenience and consistency, we will set it to $\overline{c}$. ($\overline{W} = B$ and vice-versa).

A square can be encoded in 4 bits. 1 bit for the color c, and 3 bits for the piece type p.

For example, if P_W sees a black knight on the c4 square, then $S_{W,3,4} = \langle B, Knight \rangle$.

Suppose $S_{c,i,j} = \langle c', p \rangle$.
Let

$$\mathit{col}(S_{c,i,j}) = c'$$

and let

$$\pi(S_{c,i,j}) = p$$

5.2 Moving

Since we assume that the players play by the rules and follow the protocol, making a move will simply consist of updating the board state.

To move the piece on (i_s, j_s) to (i_t, j_t), P_c sets $S_{c,i_t,j_t} = S_{c,i_s,j_s}$ and $S_{c,i_s,j_s} = \langle \overline{c}, Unknown \rangle$. Of course, for P_c to move the piece on (i_s, j_s), it must be that $\mathit{col}(S_{c,i_s,j_s}) = c$. This condition holding implies that $\pi(S_{c,i_s,j_s})$ is not *Empty* or $Unknown$ by the definition of $S_{c,i,j}$.

If the player chooses to castle, he treats the move as two separate moves and uses the method above for the king and the rook individually. If the player is taking a pawn *en passant*, he does the method above for the capturing pawn and sets the square of the captured pawn to $\langle \overline{c}, Unknown \rangle$.

When a player makes a capturing move, he publicly announces the square that he is capturing.

5.3 Resetting

After each change to the board, a player may gain or lose vision of some squares. We will update the entire board's vision in the next step, so we will reset/clear the current vision to avoid having dangling squares.

For each $S_{c,i,j} \in B_c$, if $col(S_{c,i,j}) \neq c$, then P_c sets $S_{c,i,j} = \langle \overline{c}, Unknown \rangle$

5.4 Querying

After resetting the board in the last step, the player needs to rebuild his board's vision by querying his opponent.

Querying takes part in three phases: general, pawn, and *en passant*. In the general phase, a player queries for squares that he has complete vision of. In the pawn phase, a player queries for squares that he only has partial vision of (squares in front of pawns). In the *en passant* phase, a player queries for the squares that could contain a pawn that can be captured *en passant*.

Although the pawn and *en passant* phases are not always necessary to rebuild a board's vision, it is imperative that all three phases still be carried out, as not doing so could leak information. Suppose P_W is rebuilding his board's vision by querying P_B. If the pawn phase was optional and P_W decides not to engage in it, then P_B would learn that P_W has vision of all squares in front of his pawns. Similarly, if the *en passant* phase was optional and P_W decides to skip it, then P_B learns that P_W has no pawns on the 5th rank. If P_B is the one skipping, then P_W learns that P_B did not play a double pawn push on his most recent move. The players are assumed to be semi-honest, so P_B would not be skipping the third phase to withhold information.

The board is finite in size, so each piece type has an upper bound to the number of squares that it has to query.

Definition 2. Let $p \in PIECE \backslash \{Empty, Unknown\}$. The maximum number of squares that p needs to query is $max(p)$. Explicitly,

$$\begin{array}{rrr} max(Knight) = 8 & max(Bishop) = 13 & max(Rook) = 14 \\ max(King) = 8 & max(Queen) = 27 & max(Pawn) = 27 \end{array}$$

The pawn itself has an upper bound of 5, but due to its potential promotion to a Queen, which has the greatest upper bound, its bound needs to be 27 as well.

Both players know what their opponent's pieces are because they know which of their opponent's pieces have been captured. Thus, announcing the piece that you are querying for and then querying the maximum number of squares for that piece leaks no information on its position. In the case of a promoted pawn, the querying player still announces it as pawn as not to unnecessarily reveal its new type.

Definition 3. A query set for a piece $S_{c,i,j}$ is a set Q of ordered lists L of pairs that the piece can potentially see. Each pair is comprised of a file and a rank coordinate. A query set depends on the piece type $\pi(S_{c,i,j})$, the piece color $col(S_{c,i,j})$ and the square's coordinates (i, j). It also depends on the query phase.

General phase:

- Case $\pi(S_{c,i,j}) = Rook$

$$\begin{aligned} Q(S_{c,i,j}) = \{&[(i-n, j) \mid n \in \mathbb{N}, n < i], \\ &[(i+n, j) \mid n \in \mathbb{N}, n \leq 8-i], \\ &[(i, j-n) \mid n \in \mathbb{N}, n < j], \\ &[(i, j+n) \mid n \in \mathbb{N}, n \leq 8-j]\} \end{aligned}$$

- Case $\pi(S_{c,i,j}) = Bishop$

$$\begin{aligned} Q(S_{c,i,j}) = \{&[(i+n, j+n) \mid n \in \mathbb{N}, n \leq 8 - \max(i, j)], \\ &[(i+n, j-n) \mid n \in \mathbb{N}, n \leq \min(8-i, j-1)], \\ &[(i-n, j+n) \mid n \in \mathbb{N}, n < \min(i-1, 8-j)], \\ &[(i-n, j-n) \mid n \in \mathbb{N}, n \leq \min(i, j) - 1]\} \end{aligned}$$

- Case $\pi(S_{c,i,j}) = Queen$

$$\begin{aligned}Q(S_{c,i,j}) =&\{[(i+n, j+n) \mid n \in \mathbb{N}, n \leq 8 - \max(i,j)],\\ &[(i+n, j-n) \mid n \in \mathbb{N}, n \leq \min(8-i, j-1)],\\ &[(i-n, j+n) \mid n \in \mathbb{N}, n < \min(i-1, 8-j)],\\ &[(i-n, j-n) \mid n \in \mathbb{N}, n \leq \min(i,j) - 1]\\ &[(i-n, j) \mid n \in \mathbb{N}, n < i],\\ &[(i+n, j) \mid n \in \mathbb{N}, n \leq 8 - i],\\ &[(i, j-n) \mid n \in \mathbb{N}, n < j],\\ &[(i, j+n) \mid n \in \mathbb{N}, n \leq 8 - j]\}\end{aligned}$$

- Case $\pi(S_{c,i,j}) = Knight$. Let $U = (\{-1,1\} \times \{-2,2\}) \cup (\{-2,2\} \times \{-1,1\})$

$$Q(S_{c,i,j}) =\{[(i+x, j+y)] \mid (x,y) \in U, 1 \leq i+x, j+y \leq 8\}$$

- Case $\pi(S_{c,i,j}) = King$. Let $U = (\{-1,0,1\} \times \{-1,0,1\}) \backslash \{(0,0)\}$

$$Q(S_{c,i,j}) =\{[(i+x, j+y)] \mid (x,y) \in U, 1 \leq i+x, j+y \leq 8\}$$

- Case $\pi(S_{c,i,j}) = Pawn$ and $col(S_{c,i,j}) = W$

$$Q(S_{c,i,j}) =\{[(i+x, j+1)] \mid x \in \{-1,1\}, 1 \leq i+x \leq 8\}$$

- Case $\pi(S_{c,i,j}) = Pawn$ and $col(S_{c,i,j}) = B$

$$Q(S_{c,i,j}) =\{[(i+x, j-1)] \mid x \in \{-1,1\}, 1 \leq i+x \leq 8\}$$

Pawn phase:

- Case $\pi(S_{c,i,j}) = Pawn$ and $col(S_{c,i,j}) = W$

$$Q(S_{c,i,j}) = \begin{cases}\{[(i, j+1)]\} & j > 2\\ \{[(i, j+1), (i, j+2)]\} & j = 2\end{cases}$$

- Case $\pi(S_{c,i,j}) = Pawn$ and $col(S_{c,i,j}) = B$

$$Q(S_{c,i,j}) = \begin{cases}\{[(i, j-1)]\} & j < 7\\ \{[(i, j-1), (i, j-2)]\} & j = 7\end{cases}$$

En passant phase:

- Case $\pi(S_{c,i,j}) = Pawn$, $col(S_{c,i,j}) = W$ and $j = 5$

$$Q(S_{c,i,j}) = \{[(i+x, j) \mid x \in \{-1, 1\}, 1 \leq i + x \leq 8]\}$$

- Case $\pi(S_{c,i,j}) = Pawn$, $col(S_{c,i,j}) = B$ and $j = 4$

$$Q(S_{c,i,j}) = \{[(i+x, j) \mid x \in \{-1, 1\}, 1 \leq i + x \leq 8]\}$$

The actual querying will be done using the ANDOS protocol from [BCR86]. Without going all the details, the protocol goes as follows:

Let's call the querying player the Bob, and the queried player the Alice.
Let $x_1, x_2, ..., x_{63}, x_{64}$ be the queried player's 64 4-bit secrets.
For a square at (i, j), the corresponding secret is x_k, $k = 8(j-1) + i$.
Let b_{ij} be x_i's j^{th} bit.
Assume Bob is P_c and Alice is $P_{c'}$.

1. Alice prepares her 64 secrets depending on the phase.

2. Alice initializes the ANDOS protocol as described in [BCR86]. She randomly selects two large primes p and q, computes $m = pq$, and randomly selects a quadratic non-residue y modulo m.

3. For each secret bit b_{ij}, she computes $z_{ij} = x_{ij}^2 y^{b_{ij}} \mod m$, where x_{ij} is a random element of $\mathbb{Z}_m^*$.

4. She sends m, y, and all the z_{ij}'s to Bob.

5. Alice convinces Bob of the validity of the initialization using the protocols of [GHY85] and [GMR85].

For each $S_{c,i,j}$, where $col(S_{c,i,j}) = c$, Bob gets $Q = Q(S_{c,i,j})$ according to the phase and announces $\pi(S_{c,i,j})$. Then for each list $L \in Q$, and for each $(i', j') \in L$ (in order),

6. Bob prepares a random permutation σ of $\{1, 2, ..., 64\}$.

7. For each $1 \leq k \leq 64$ and $1 \leq j \leq 4$, Bob computes $q_{ij} = z_{ij} r_{kj}^2 y^{a_{kj}} \mod m$, where r_{kj} is a random field element, a_{kj} is a random bit, and $i = \sigma^{-1}(k)$. He takes all of the q_{ij}'s and puts them in a $\sigma-$packet P_σ.

8. Bob gives P_σ to Alice and convinces her of its validity.

9. Bob sends $k = \sigma(8(j'-1) + i')$.

10. For each $1 \leq j \leq 4$, and the k that she received, Alice tells Bob whether q_{kj} is quadratic residue or non-residue.

11. Bob infers the contents of the square at (i', j').

It is important that Bob redo steps 6 to 8 for every query that he asks Alice. Not doing so would reveal whether his queries are distinct or not, which will need to be hidden in the upcoming steps.

In a semi-honest setting, the convincing part of steps 5 and 8 is not necessary, since it is assumed that the players will not act maliciously.

General Phase

Alice's 64 secrets are $x_1, x_2, ..., x_{64}$, where $k = 8(j-1) + i$ and

$$x_k = \begin{cases} S_{c',i,j} & col(S_{c',i,j}) = c' \\ \langle c', Empty \rangle & otherwise \end{cases}$$

Now, let $S_{c,i,j}$ be the piece for which Bob is querying for and let (i', j') be the coordinates of the square that he is interested in. He receives the corresponding secret x_k.
If $col(S_{c,i',j'}) \neq c$, he sets $S_{c,i',j'} = x_k$.
Then, if $\pi(S_{c,i',j'}) = Empty$, he continues querying by repeating steps 6-11 as usual.
If $\pi(S_{c,i',j'}) \neq Empty$, he breaks out of the current sequence and moves on to the next list in Q. If Bob has finished all of the lists in Q, and the number of queries that he actually asked Alice m is less than $max(\pi(S_{c,i,j}))$, then he fixes (i, j) and he queries for it $max(\pi(S_{c,i,j})) - m$ times. If steps 4-5 were not done for each query, then Alice would notice that Bob is repeating queries, which indicates that the piece's vision is complete and the remaining queries are meant as padding.

Pawn Phase

The secret x_k is

$$x_k = \begin{cases} \langle c', Unknown \rangle & col(S_{c',i,j}) = c' \\ \langle c', Empty \rangle & otherwise \end{cases}$$

The rest is identical to the general phase.

En Passant Phase

If Alice did not play a double pawn push as her most recent move, she sets $x_k = \langle c', Empty \rangle$ for $1 \leq k \leq 64$. If she did play such a move, she sets $x_k = \langle c', Pawn \rangle$ where $k = 8(j-1)+i$ and (i,j) are the coordinates of the moved pawn. She sets all the other secrets to $\langle c', Empty \rangle$.

Let $S_{c,i,j}$ be the piece for which Bob is querying for and let (i', j') be the coordinates of the square that he is interested in. He receives the corresponding secret x_k.
If $x_k = \langle c', Pawn \rangle$, he sets $S_{c,i',j'} = x_k$. Otherwise, he does nothing and continues to query down the list.

Note on Promotion

When a player promotes a pawn, he updates $S_{c,i,j}$ of the square in question to the right piece type. However, he needs to remember that it is a promoted piece. When he goes to query the squares in this piece's vision, he announces it as a pawn, but treats it as the actual piece type.

5.5 Turns

The procedure for each turn goes as follows

1. P_W MOVE
2. P_W RESET
3. P_W QUERY
4. P_B RESET
5. P_B QUERY
6. P_B MOVE
7. P_B RESET
8. P_B QUERY
9. P_W RESET
10. P_W QUERY

5.6 Security and Issues

Under the assumption that the quadratic residue problem (QRP) is hard [GM84], the ANDOS protocol is computationally secure [BCR86]. Announcing publicly the piece that you are querying for does not reveal any new information to the opponent. The players also query sufficiently many squares not to reveal the whereabouts of any piece. This first protocol is then secure for players who do not have access to infinite computational power.

The first area of improvement would be decreasing the number of queries needed to rebuild a board's vision. Currently, the players must over query as not to leak any information on the whereabouts of their pieces. There are also many redundant queries, as the players may ask to receive the same squares multiple times at no additional gain. In the first protocol, establishing the vision of just 3 pawns requires 81 queries. In theory, the number of OT rounds needed to fully rebuild vision is at most 88 (64 in general phase, 8 in en passant phase, and 16 in pawn phase). This number can be further lowered depending on what pieces are left. Since the number of pieces of each color is commonly known, and since the players have permanent vision of their own pieces, then it is not necessary to query for the squares of your own pieces.

Now, the most glaring issue of this protocol is the heavy reliance on the assumption that the parties are semi-honest. The queried players trusts that the querying player does not ask about squares that he should not see. Likewise, the latter trusts that the former is not withholding or sending any wrongful information. For a dark chess protocol to be viable in a fully dishonest setting, there would need to be a way for the players to convince their opponent that their actions are legit. Of course, there is also the constraint that they be able to do so without revealing anything that the opponent should not possess.

6 Improved Protocol Sketch

With the overall structure of the protocol out of the way, the focus now is to weaken the assumption of honest-but-curious adversaries to having malicious players.

6.1 Zero-Knowledge

In 1985, Goldwasser, Micali and Rackoff [GMR85] introduced the idea of interactive proofs, a way for a prover to convince a verifier of some statement by interacting with them. They also showed that for some statements, the proof can be done in zero-knowledge. That is, other than now being convinced of the statement, a polynomially bounded verifier does not gain any new knowledge that he did not already have. Later, Goldreich, Micali and Wigderson [GMW86] showed that all statements in NP can be proved in zero-knowledge. In a chess protocol, the types of statements that would be made are of the sort ”I made a legal chess move” and ”My query is valid with regards to my board”, which are all easily verifiable. It is then certainly possible for the players to prove them in zero-knowledge.

6.2 Proof vs Argument

When talking about zero-knowledge *proofs*, there is the notion that a proof be statistically sound, that is, not even an unbounded prover can convince a verifier of a false statement. Since our protocol is already limited to beings that are less than all-powerful, it is not necessary to maintain such level of security. For our purposes, the weaker notion of *arguments* suffices. Introduced by Brassard, Chaum, and Crépeau [BCC88], argument systems only require that computationally bounded provers be unable to prove unsound statements. Going forward, I will use the term ”proof” out of convenience, but note that the notion of arguments satisfies our needs.

6.3 Commitments and Chaining

Not only does every action itself need to be proved, they must be valid with regard to the current game state, which is the result of all the actions that come before it. The progression of a game can be seen as a chain, where every new action, along with its proof, attaches to the end of the existing sequence. If a game begins in a valid state (the chess starting position), and all additions to the chain are proven to be valid, then the entire chain will be a transcript of a valid game and the most recently added piece will reflect a legal board state.

One way to implement this chain is the use of zero-knowledge proofs in combination with commitments. A commitment scheme allows one party to commit to a value in a binding and concealing way. The committer also has to reveal the value later on. The binding property ensures that the committer cannot change the value once a commitment is made, and the concealing property ensures that the other parties cannot know what the value is before it is revealed [BCC88]. For each action that wants to be included into the chain, the player needs to provide a proof that shows 1. the action itself is valid, 2. the action is valid with respect the current game state (the last commitment), 3. a commitment to the result of the action (the new state). A player cannot change their past actions due to the binding commitments, and the opponent cannot learn anything about the actions because of the concealing property.

Commitments also solve another shortcoming in the semi-honest protocol. In the first version, the queried player must be trusted in giving accurate information about his pieces, and the querying player must be trusted in updating his board correctly after receiving an answer. A verifiable or committed oblivious transfer is an extension of the notion of an OT, where the sender is committed to his secrets and the receiver is committed to his choice and obtained result [Cr 90]. Crépeau showed that any OT protocol can be turned into a verifiable OT, so we could use a verifiable construction of the ANDOS protocol, the details of which will be left out. The secrets are already committed in each piece of the chain. What is left is for a player to commit to the square that he is querying, and then prove that he updated his board correctly in accordance to the committed result.

Proving Actions

Each action needs a proof and a commitment. More specfically, the needed proofs are

1. A move is valid
2. The board was reset correctly
3. A query is valid
4. The board was updated correctly with respect to a query result

The New Protocol

The new protocol will proceed in the same way as the first, with some slight additions.

Both players first initialize their local boards to the starting positions (with appropiate starting vision). Both players commit to their boards and immediately open their commitments to show that they set up the game properly. These commitments serve as the beginning of each player's chain. Then, on each turn, the players do the same steps as the first protocol, committing and proving each action as they go. Note that each individual query and each associated board update will need its own commitment and proof. Thus, the QUERY step is actually many smaller steps.

Acknowledgements

I would like to thank Claude Crépeau for his time and valuable insights. I would also like to acknowledge his numerous contributions to the field of cryptography. Many of the notions I used in this project were either his work or a product of it, so I literally could not have done it without him.

References

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, oct 1988.

[BCR86] Gilles Brassard, Claude Crépeau, and Jean-Marc Robert. All-or-nothing disclosure of secrets. *Advances in Cryptology — CRYPTO' 86*, page 234–238, 1986.

[Cr 86] Claude Crépeau. A secure poker protocol that minimizes the effect of player coalitions. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pages 73–86, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[Cr 87] Claude Crépeau. A zero-knowledge poker protocol that achieves confidentiality of the players' strategy or how to achieve an electronic poker face. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 239–247, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[Cr 90] Claude Crépeau. Verifiable disclosure of secrets and applications (abstract). In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology — EUROCRYPT '89*, pages 150–154, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

[GHY85] Zvi Galil, Stuart Haber, and Moti Yung. A private interactive test of a boolean predicate a minimum-knowledge public-key cryptosystems. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 360–371, 1985.

[GKWZ] Aayush Gupta, Nicolaas Kaashoek, Brice Wang, and Jason Zhao. Zero-knowledge battleship. `https://courses.csail.mit.edu/6.857/2020/projects/13-Gupta-Kaashoek-Wang-Zhao.pdf`.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GMR85] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, page 291–304, New York, NY, USA, 1985. Association for Computing Machinery.

[GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *27th*

Annual Symposium on Foundations of Computer Science (sfcs 1986), pages 174–187, 1986.

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer. In *Technical Report TR-81, Harvard Aiken Computation Laboratory*, 1981.