MIECT: Security 2020-21

Secure, multi-player online domino game

Presentation of project: January 25-29, 2021
Deadlines: January 24, 2021 (project delivery)

André Zúquete

Changelog

- v2.0 - The tile distribution protocol was modified to include tile pseudonimization.
- v1.1 - Typos fixed.
- v1.0 - Initial version.

1 Overview

The objective of this project is to develop a system enabling users to create and participate in online, domino games. The system is composed by a table manager and a bounded set of players. The system should be designed to support the following security features:

- **Anonymity and authentication of players**: All players must authenticate their actions to a given pseudonym. Points collected in one or more games stay associated to that pseudonym.
- **Identity assurance for accounting** The points collected by an anonymous player on behalf of a pseudonym can be transferred to a given identity using strong identification methods. Namely, players should use their Citizen Card to prove they are the owners of a given pseudonym since the beginning of its use. Pseudonyms resolved to identities upon point accounting should never again be used.
- **Confidential and random tile distribution**: the table manager creates the tile deck and initiates its hiding and scrambling in order to create the stock. The stock distribution should involve all players, and all of them should contribute to the randomness of the stock distribution and to the guarantee that no single entity is able to know, for

sure or with a high success probability, the hand of any other player by tracing the tiles they have picked from the stock.

- **Honesty assurance**: players must play only with the tiles they have picked (their hand), and cannot use any other. To enforce this, players must commit to their hand before the start of the game. The commitment must not reveal the tiles in the hand and must be properly signed.
- **Correct game evolution**: the game evolution should be orchestrated by the table manager. On the other hand, the table manager and the players can collaborate to catch cheaters. For instance, the table manager and the rest of the players can detect a tile played twice, or when a tile was played in a situation where it could not have been used. And players can detect if another player is using one of their tiles.

  In the first situation, either a wrong play can be denied a anteriori, and the game can proceed upon a correct play, or a wrong play can only be detected a posteriori, and the game needs to be aborted. In the second case, all players must show their initial hand by revealing the data that originated the bit commitment; the cheater should not be able to do so.

  Dominoes admit two variants regarding the stock: in draw games, it can be used by players without any playing alternative to find a suitable one. In this case, a cheater can attempt to guess a tile in the stock and use it instead of another one in its hand. This case may ultimately be discovered during the course of the game if another player effectively picks that piece from the stock later on, or at the end of the game, when each player must show its initial hand.
- **Correct accounting**: at the end of each game some accounting must be done relatively to the outcome of the game (e.g. points earned by players). Such accounting must be agreed by all players. Those points can then be transferred to given identities by proving the relationship between a pseudonym and an identity (by means of a Citizen Card).

2 Project Description

2.1 System Components

We can consider the existence of two main components: (i) multiple clients, through which players interact, and (ii) one server, which serves as table manager. A table manager can serve several games simultaneously, but for simplicity we will consider that it serves only one game at the time.

Ultimately, each player can develop their own client application, but this is not the goal of this project. Therefore, each group will develop a single

client, which can be run by several different players (in different hosts, if necessary).

2.1.1 Table manager (server)

This server will expose a connection endpoint through which clients can exchange structured requests/responses with it.

The table manager is the system component that forms a table of anonymous players, creates the tile deck, initiates the formation of the stock and its distribution (with the help of the players), controls as much as possible the sequence of the game, receives the complaints of the players, makes the game accounting, receives accounting confirmation from players and associates accountings to identities.

Before being able to play, players should present themselves to the table manager (show their pseudonym) and prove their pseudonym ownership. The pseudonym ownership should include a way to link the pseudonym to an identity that will ahead reclaim the points earned by the pseudonym (using the Citizen Card). This way, no anonymous player can sell their earnings to arbitrary people. Note, however, that the identity of a pseudonym owner cannot be extract solely from the (anonymous) player registration data; only when a player presents the identity related with its pseudonym such relation must be established without any doubt.

Each person should not be allowed to have more that one pseudonym participating in a single game, as it gives that person and advantage over the other players. However, this requirement is hard (or even impossible) to fulfil without a physical control of the players, and therefore it should be disregarded in the project. Students should assume that players do not know each other and do not work in collusion.

2.1.2 Player (client)

A player is an application that interacts with a user, enabling they to create and participate in games. This application needs to interact with the user's Citizen Card for producing digital signatures only when collecting the points earned by its pseudonym.

On the other hand, a player will always act on behalf of a pseudonym, which should be undoubtedly bound to the following actions:

- Commit to a given hand;
- Play a given tile;
- Pick a tile from the stock;

- Protesting against an illegal play; and
- Agree with a game accounting.

Throughout the interaction between a player and a table manager, or even pairs of players (see below), they both need to prove to themselves that they are authentic, which requires source and flow authentication. However, such authentication may explore mechanisms lighter than digital signatures, for instance, using message authentication codes based on session keys. When confidentiality is required together with authentication, students may consider the use of authenticated encryption strategies.

Suggestion: during the registration process clients may provide a (signed) value that enables them to establish different session keys with the table manager and all other players.

Note that the player and the table manager applications implement a client-server model to establish their communication (is the table manager that creates the rendez-vous communication endpoint for the players it manages), most of the time they perform a master-slave interaction model (the table manager sends a command to the player and the player performs an action and responds).

2.2 Security assumptions

All entities can cheat; no individual honesty is assured. But, if possible, the system should be immune to collusion. Namely, a player and the table manager, together, cannot derive with complete precision the hands of the other players. It is trivial to show that all but one colluded players can derive with some precision the game of the remaining player (depending on the stock), and there is no technology that can solve this problem other than to prevent physically the possibility of collusion.

2.3 Processes

There are several critical processes that must be supported. Students are free to add other processes as deemed required.

- Join/leave a table manager;
- Start a game;
- Create the stock, distribute tiles from the stock, keep the remaining stock;
- Commit to a hand;
- Play tiles from a hand;

- Check the tiles played by others;
- Pick a tile from the stock;
- Complain against cheaters;
- Accept the outcome of a game;
- Cash the points of a pseudonym for a given identity.

Several types of domino games are possible, but only one needs to be implemented. Choose a simple game, such as the double-six game, with a deck of 28 tiles.

Draw variants, where players are allowed to pick tiles from the stock during the game when otherwise not being able to play, must nevertheless be designed and implemented to aim for the highest grade. Non-draw variants, besides less graded, should nevertheless do not allow players to know the contents of the stock during the game (even if not used, its knowledge partially or totally reveals the game of the adversaries).

In terms of players per game, allow a variable number of players, with a minimum of 2. Also, allow games to start with a given number of tiles per hand, leaving the rest of them in the stock.

The player application should be designed to play alone, i.e., without user intervention. Tile selection on each play does not need to be clever, it just needs to be legal or illegal (if you want, you can rank tiles by value and start choosing from the topmost in the ranking). Illegal actions can be randomly activated, in order to test the robustness of the system. Graphical or otherwise appealing game interfaces will not be valued.

Note that automatic playing by a player application is fundamental for testing and debugging, as it allows to run games much faster than with human input. Without it, testing the project is a nightmare!

3 Cryptographic protocols

A bit commitment is a value that stands as a commitment made by someone relatively to some action performed (or data created) in the past that can be shown in the future to hold. In the meantime, the bit commitment does not allow others to tell which actual action (or data) were committed by its creator.

In this project, a bit commitment of a hand is a computation performed over the set of tiles that form a hand which can be revealed before the beginning of the game without revealing the tiles. At the end of the game, a player can demonstrate its honesty by showing the data (which must include the

hand) that was used to compute its bit commitment, thus enabling others to check whether or not it played with the right tiles.

Bit commitments can be computed in many ways. One of them consists in using a one-way hash function h and two random values. Assuming that the value to commit with is T (tiles), the bit commitment b can be computed as

$$b = h\left(R_1, R_2, T\right)$$

and published as (R_1, b). To prove the correctness of this bit commitment one has to publish (R_2, T).

The deck randomization and hidden selection of tiles per player is not trivial, but there are some solutions. One is the following, that runs in four five:

1. **PSeudonymization stage**: the deck of N tiles is created by the table manager as a set of N elements with the two values that form a domino tile. All tiles are randomly numbered with a index from $[0, N-1]$ (hereafter referred as tile index). An pseudonymized deck is then created by replacing the value of tile i vale by a pseudonym P_i.

   A pseudonym is a value the hides a real value (here, a tile value), but can be reversed on a needed basis. Here, we go further, and pseudonyms can be shown to be correct, i.e., their correspondence to a given tile value can be demonstrated.

   One way to create pseudonyms that can be demonstrated to be correct is the following:

   $$P_i = h\left(i, K_i, T_i\right)$$

   where K_i is a random, unique key for tile index i (T_i). Given a P_i for an i, it is infeasible to derive a key and a tile value that could produce it. Furthermore, knowing the key K_i that was used to calculate the pseudonym P_i, one cannot find T_j from P_j, for $j \neq i$, because $K_i \neq K_i$. Only the manager can provide the T_i that corresponds to a P_i, and prove the correspondence with K_i, because it can keep a pseudonym table.

   Concluding, at the end of this stage we have a set of N $\langle i, P_i \rangle$ tuples, or a pseudonymized deck. Only the manager knows what each piece of the deck means.

2. **Randomization stage**: the table manager circulates the pseudonymized deck by the players, and each player adds a layer of encryption of each tuple (it is fundamental to encrypt the index!). Furthermore, each player shuffles the encrypted tuples.

   The final result will constitute the stock. It will be a shuffled, encrypted set of pseudonymized tiles. Nobody knows where each tile is.

Nobody.

Each tuple must be encrypted with a different key, but tuples cannot be indexed. Therefore, for each key selected for a tuple by a player, it must keep internally a mapping

$$C \longrightarrow K$$

where C is the encrypted tuple value and K the key that was used to produce it.

Thus, if a tuple is encrypted 2 times (sequentially, by players A and B, in this order), it will produce the following transformation sequence

$$\langle i, T_i \rangle \xrightarrow{K_i} C_i \xrightarrow{K'_i} C'_i$$

and A will keep a mapping

$$C_i \longrightarrow K_i$$

whilst B will keep the mapping

$$C'_i \longrightarrow K'_i$$

It is possible that, for at least two tiles on the stock, the encryptions performed by a player produce equal values. This collision should be detected and avoided by generating a new key.

3. **Selection stage**: the stock circulates secretly among players. Each player receives the stock and randomly selects one out of two actions: pick up a (random) tile or back off.

   In the first case, it removes the (encrypted) tile from the stock.

   In the second case, two options can happen: maintain the stock as it is or swap part (or the whole) of the tiles already picked up by itself by other tiles from the stock.

   At the end, the stock is sent (directly) to another player, randomly selected. The probability of tile pick-up should be low (say, 5%), in order to increase the difficulty of tracing the effective distribution protocol.

   When all players have their hands complete (and the circulating stock reaches is given low level), any player can randomly activate the following final stages.

   Note that for this stage secrecy is critical, as no external observer can be able to follow the actions of each individual player. The same is also true for each player, they cannot know what each other player did in the past, or will do in the future.

4. **Commitment stage**: each player, sequentially publishes a signed bit commitment of its hand (still encrypted).

   All players should get the bit commitments of all other players, together with the stock. This is going to be used to check the validity of the game's starting point by everybody (including the table manager).

5. **Revelation stage**: the last player to encrypt the stock reveals to the other players the keys that decrypt all tiles except the ones in the stock. The previous player uses those keys and does the same, and so do the others, until all the keys that were used to encrypt the distributed tuples are known.

   At the end of this stage, the table manager knows which tuples were chosen by the players, and which still remain in the stock. However, it does not know which tuples exist in the hand of each player, nor the tuple corresponding to each stock member.

6. **Tile de-anonymization preparation stage**: players will randomly (and secretly) fill an array of random public keys, for which they have a corresponding private key. Such keys are indexed with the tuple indexes that each player received. Thus, if a player picked tuple i, it will fill the slot i with a public key K_i^+. At the end, all slots corresponding to all selected tuples will be filled, each with a unique public key.

   Each player receives the public key array and selects one out of two actions: add a public key (for one of its tuples that still does not have one) or back off.

   In the first case, it creates a new key pair, saves the private key with the tuple, an sets the public key in the array using the tuple index.

   At the end, the array is sent (directly) to another player, randomly selected. The probability of a public key addition should be low (say, 5%), in order to increase the difficulty of tracing who added which key.

   When all players have added all their public keys, any player can randomly activate the following final stage.

7. **Tile de-anonymization stage**: the table manager uses each public key K_i^+ to encrypt T_i and K_i, and circulates all the results (in one or more messages) by all players. The owner of each K_i^+ will then be able to, privately, have access to its tile T_i, and check if the table manager did not cheat in providing that information.

   At the end of this stage, no single player knows the tiles picked by each other player, or even by all of them. It can have access to this last information, though, if it could collude with the table manager.

8. **Stock use**; For using each stock piece during the game, a player has to get the other keys from the other players and, upon a multiple decryption with all the keys, in the right order, must ask the table manager for a tile de-anonymization. This means that, for each piece picked from the stock, the table manager will know which tile was effectively picked, but cannot influence that.

This protocol is not perfect, but its weaknesses are tolerable:

- The table manager knows which tiles were blindly selected by players in the beginning of the game, as well as the tiles that remain in the deck. But it could not influence the tile distribution in any way. The information it has, if shared with a player, gives to the benefited some advantage for choosing the best tile to play.
- The table manager knows which tile was picked from the stock by a player during the game. But it could not influence the tile selection from the stock. Again, if this information is shared with another player, gives to the benefited some advantage for choosing the best tile to play thereafter.

If we assume that the table manager does not leak information to any player, than it cannot corrupt a game.

The direct communication between players must be confidential, to prevent third parties from eavesdropping it. Such communication can pass through the table manager, for making it easier to implement the player-to-player communication, but the table manager must not understand the data exchanged. Therefore, players must establish a session among them, one per each pair of players, with a temporary session key for protecting the communication. Session keys can be used to ensure confidentiality or authenticity.

Note, however, that the authenticity of actions or data that must be checked by all players, and not only one player, such as the bit commitments or each tile played, must be ensured with digital signatures performed with a private key bound to a pseudonym.

In protocols that take a variable number of iterations to increase confusion, it is often crucial to maintain the size of the exchanged messages in order to prevent eavesdroppers from performing side-channel attacks, namely to learn how the protocol is evolving over time. Thus, all messages exchanged during the stock distribution protocol should be of equal size, regardless of the number of tiles already pick up so far by the players.

4 Suggestions

The communication between players and the table manager, or with other players, is easier to implement with TCP streams, one per player with the table manager. Direct communication between players can be routed through the table manager.

The table manager can handle user input to start a game, or use a launch option to define the minimum set of players required to initiate a game.

Since the use of the Citizen Card is only to cashing points from an anonymous user to a given identity, students can use the same player application, with the same Citizen Card, to handle several players in the same game, as long as, at the end of a game, only one player wins.

To simplify the implementation, you may:

- Assume the use of well-established, fixed cryptographic algorithms. In other words, for each cryptographic transformation you do not need to describe it (i.e., what you have used) in the data exchanged (encrypted messages, receipts, etc.). **A bonus of 2 points** may be given if the complete system is able to use alternative algorithms.
- It can be assumed that each server has a non-certified, asymmetric key pair (Diffie-Hellman, RSA, Elliptic Curve, etc.) with a well-know public component.

Do not encrypt communications that are not secret. Note that most of the time you need integrity control, not secrecy. Unnecessary secrecy will be penalized.

It is strongly suggested to structure all exchanged messages as JSON objects or Google Protocol Buffers. JSON is a very user-friendly textual format and there are many libraries for building and analysing JSON objects. Binary content can be added to JSON objects by converting them to a textual format, such as Base-64. Google's Protocol Buffers[1] has the advantage of creating functions to perform the marshaling and unmarshaling of data into and from exchanged messages.

5 Functionalities to implement

The following functionalities, and their grading, are to be implemented:

- (2 points) Protection (encryption, authentication, etc.) of the messages exchanged;

[1] `https://en.wikipedia.org/wiki/Protocol_Buffers`

- (1 points) Set up of sessions between players and a table manager;
- (1 points) Set up of sessions between players;
- (2 points) Deck secure distribution protocol;
- (2 points) Validation of the tiles played during a game by each player;
- (1 points) Protest against cheating;
- (1 points) Possibility of cheating;
- (1 points) Game accounting;
- (2 points) Claim of points for an identity provided by a Citizen Card;
- (3 points) Picking of stock tiles during a draw game;

Grading will also take into consideration the elegance of both the design and actual implementation.

Up to 2 (two) bonus points will be awarded if the solution correctly implements interesting security features not referred above. But, please, implement the features required first!

A report should be produced addressing:

- the studies performed, the alternatives considered and the decisions taken;
- the functionalities implemented; and
- all known problems and deficiencies.

Grading will be focused in the actual solutions, with a strong focus in the text presented in the report (4 points), and not only on the code produced! It is strongly recommended that this report clearly describes the solution proposed for each functionality. Do not forget to describe the protocols used and the structure of each different message.

Using materials, code snippets, or any other content from external sources without proper reference (e.g. Wikipedia, colleagues, StackOverflow), will imply that the entire project will not be considered for grading or will be strongly penalized. External components or text where there is a proper reference will not be considered for grading, but will still allow the remaining project to be graded.

The detection of a fraud in the development of a project (e.g. steal code from a colleague, get help from an external person to write the code, or any other action taken for having the project developed by other than the responsible students) will lead to a grade of 0 (zero) and a communication of the event to the University academic services.

6 Project phases

The security features should be fully specified prior to start its implementation.

We recommend the following steps for a successful project development:

- Develop a complete, non-secure client and server applications. This step can start immediately.
- Produce a draft report of the security specification.
- Add secure sessions between players and a game manager and between players.
- Add security to the deck distribution protocol.
- Add the validation of the played tiles.
- Add the accounting and its validation.
- Use the Citizen Card to reclaim earned point by a pseudonym.
- Add cheating support.
- Add the protest against cheating.
- Add stock picking.

Since groups are formed by several elements, you can (and should) develop parts of the project in parallel. For instance, one student can develop the stock creation and distribution, another student the game playing, another the game playing validation, and another the accounting. Choose the one that better suits you, but maximize parallelism.

7 Delivery Instructions

You should deliver all code produced and a report before the deadline. That is, 23.59 of the delivery date, January 24, 2021. Penalties will apply to late deliveries (1 point per day, evaluated on a minute basis).

In order to deliver the project you should use a project in the CodeUA[2] platform. Please send an email to the course professor (André Zúquete) with the email of the group members to request a project creation. Do not create a project by your own!

Each CodeUA project should have a `git` or `svn` repository. The repository can be used for members of the same group to synchronize work. After

[2] `https://code.ua.pt`

the deadline, and unless otherwise requested by students, the content of the repository will be considered for grading.