Project 2 - Secure Game

Description

This assignment will focus on the implementation of a robust protocol for handling a distributed game. The game under study will be Bingo, which is a game of chance. Implementation will consist of a server (caller) and multiple clients (players) communicating over a network.

Bingo: the game

Bingo is a game that can take a variable number of players. We will not deal with the monetary dimension of it. Each player receives a card with a random set of M unique numbers (from 1 to N, but not all, since $M < N$) and those numbers (deck) are then randomly selected by a caller (the game host, the entity that coordinates the evolution of the game) until a player completes a row in their card with the numbers selected so far. That player is the winner. It is possible to have several winners, though.

In the secure version, we consider that each player can create its card, by shuffling the numbers available, and then commit the card to the playing area.

For simplicity sake, we will consider that a player will win the game if his card becomes completely filled with the numbers selected by the caller, and not just a single row.

Second, the deck numbers are not picked one by one, randomly, until one player wins. Instead, the caller will send all N numbers in the deck randomly shuffled, and each player receives the exact same shuffled deck. Then, each player is able to find the winners by filling all the cards with the deck numbers, following their order, and determining which player fills the card with the least amount of numbers. Thus, at the end, all players can agree on the winner(s).

No entity has complete control over the game. Players can detect if the caller cheated (e.g. did not provide a deck with set of unique N numbers to be shuffled, accepted an invalid card, etc.) or if any player provided a wrong message.

Playing area

We consider the existence of a playing area, to which all users connect and acts as a secure playing field. It doesn't interfere with the game it self, but makes information available to all, implements authentication and authorization.

When the game starts, the playing area shall not accept more players.

The playing area has a keypair which it uses to sign messages exchanged or forwarded.

The playing area will also record all messages exchanged, and actions executed. The log is composed by entries with the following format:

```
sequence, timestamp, hash(prev_entry), text, signature
```

Players and caller can request this log and audit the events.

Player and Caller Identity and Registration

Upon connection to the playing area, both players and caller must be authenticated and authorized. Any Portuguese citizen can play, as proved by the use of their citizen card. The playing area can only have one caller, and the playing area will have a list of citizens that can act as callers.

Each user will be identified by a sequence number, a public key and a nickname. The caller will be number 0, and the players will have sequential numbers starting from 1.

Each user's public key belongs to an asymmetric key pair generated to itself just for playing. This key is provided during the login process and is considered to have a short duration.

A user's authentication process must ensure a commitment of the user public key and nickname. This means that attackers cannot tamper the public key registered by a user. Digital signatures and password-based message authentiction codes (depending on the authentiction process) are common solution to implement that commitment.

The caller should sign the data from all users `seq, nick, publickey`

As a simplification of this process, if students do not wish to implement support for the Citizen Card, a custom PKI may be used. This option will not award full points.

Card Generation

Each player generates their own card[1]. You can consider the card to be a set of $N/4$ positions, filled with $N/4$ random unique numbers between 0 and N.

The card of each player is committed to the caller and to the other players, protected by means of a digital signature. This way, no player can change their card once the game starts. Everybody should validate the players cards, and players that provided an invalid card (repeated numbers, signature not valid, invalid size, etc…) should be immediately disqualified by the caller.

Random shuffling of the deck numbers

The game provides an interesting point where there are two conflicting actions: creation of the deck and creation of the cards. If the deck is made available before the cards are known, players can cheat. If the cards are made available before the deck is created, the caller can cheat (no winner, or select a specific winner).

Therefore, we will use a strategy where all players will blindly shuffle the deck.

The process starts by the caller shuffling N numbers and encrypting each number with a symmetric key. The resulting deck is signed and posted to the playing area. Following the order of their registration, each player will take the deck, encrypt each number, reshuffle the numbers, sign the deck and post the result to the playing area. It is important that each signature includes the player number.

This way, the individual numbers will be shuffled and encrypted by all players, and then made available to all users. No player has access to the numbers and can cheat, and all have the opportunity to shuffle the deck, which can be validated by the signatures provided.

After all decks are provided, the caller will sign it again, making this deck the one to be considered for playing.

Finally, all players and the caller provide their symmetric keys to all users.

Players can use the keys, in reverse order, to decrypt the deck and check if each decrypted version (properly unshuffled, which means checking number by number) corresponds to the one that was committed by the correct player. Upon all decryption steps, all players must reach the exact same shuffled version of the plaintext deck, which they must confirm that is signed by the caller.

Any dishonest player detected at this last stage must be disqualified by the caller, and the disqualifications must be broadcast to all the players.

At the end of this last stage, all players can compute the winners, and communicate their outcome, signed. Again, all outcomes not conforming with the result reached by the caller are disqualified and that is broadcast to all the players.

Digital signatures on exchanged data

All the messages produced by players must be signed by them, before being transmitted. The public keys used to verify players' signatures must be conveyed to the playing area during the registration of a player for a game. Thus, when a game starts, all the players, and the caller, know the public keys of all the players.

The caller should publish the game outcome as a signed message. The public key to validate the signatures should be well-known to the players as it is available in the playing area. This way, players can abandon a game after receiving an invalid message (with a wrong signature) received from the caller.

Trust in the caller

The caller is trusted not to cheat. The caller acts mostly as a user, only with different access control rules. Upon the revelation of the encryption keys, the caller has the power to detect cheaters and disqualify them. A caller has the power to abort a game upon detecting a wrong player signature that compromises the continuation of the game.

List of keys that must be used

The following different keys must be used:

- A random, asymmetric key pair for the playing area. Used to sign messages it originates.
- A random, asymmetric key pair per player, generated just before their registration. The private key is used to sign the player’s messages. The public key is made available in the player/caller profile.
- The Citizen Card authentication key pair. Its private component should be used to sign a player’s registration message, which should include the certificate of the key pair mentioned above.
- A random, symmetric key per player, generated before encrypting the deck and stored until being publicly disclosed.

Implementation details

Encryption algorithms

Since each number in the deck must be individually encrypted, and shuffled afterwards, it is not suitable to use stream ciphers. Use instead block ciphers. Furthermore, do not use padding, as a single block of the cipher is more than enough to storing a small integer number. For simplicity sake, all the entities that encrypt and decrypt the deck can use the exact same algorithm.

Signature algorithms

The Citizen Card can only perform RSA signatures. However, the other signatures performed by the caller or the players can use RSA, DSA (Digital Signature Algorithm) or ECDSA (Elliptic Curve Digital Signature Algorithms). In this last case, and for simplicity sake, you can use a single algorithm for everybody.

Storage

Because all players are present when the game plays, the playing area doesn't need to store all data. If it forwards all actions to other participants, it only needs to store player registration data and the action log.

A basic protocol needs to be implemented to support retrieval of this information (`get_user_list`, `get_audit_log`)

Cheating tests

Both the caller and the player applications should have options for cheating (or otherwise misbehaving). Misbehaving should be probabilistic (e.g. a 10% chance of delivering an invalid signature). Cheaters must output a message each time they cheat.

Hints

In the following files you can find the base structure for a communication application with users, a playing area and the messages exchanged. This code doesn't implement any security mechanism but may provide some help with the serialization and communication aspects.

Please check: messages.py, parea.py and user.py

Project delivery

Delivery should consist of a git repository with at least three folders and a file:

- `player`: contains the player code, including instructions to run it.
- `caller`: contains the caller code, including instructions to run it.
- `playing area`: containing the code of the playing area, including instructions to run it.
- `README.md`: contains the project description and the authors;

Projects will be graded according to the implementation, the implementation of the code that implements the authentication and the protection of the related information, and the documentation produced.

This project is expected to be authored by the students enrolled in the course. The use of existing code snippets, applications, or any other external functional element without proper acknowledgement is strictly forbidden. Themes and python/php/javascript libraries can be used, as long as the vulnerabilities are created by the students. If any content lacking proper acknowledgment is found in other sources, the current rules regarding plagiarism will be followed.

References

- Bingo (English version
- Bingo (American version

---

1. This is not an advantage for players, since they do not control the order by which the deck numbers are selected to find the winners. ↩

2022

PREVIOUS
Project 1 - eHealth Corp

Last updated on 15 Nov 2022