Title: Development of a Tetris Game with Cryptographic Challenges

Summary:

The project's main objective is to create a Tetris-style game with an added layer of cryptographic challenge. The player is required to achieve a certain score within a given time frame to access an encrypted message. The development process encompasses multiple

Python files, each contributing to the game's functionality. The project has been executed in a step-by-step manner with continuous modifications to reach the desired outcome.

Game Overview:

The game starts with the user entering a message, which is then encrypted using a public key and displayed within the game window. The player's goal is to reach a predefined score within a specific time limit to unlock the decrypted message.

Key Components:

1(a). Start_game.py: This script serves as the entry point for the game, containing the MessageEncryptionApp class. It is responsible for initializing the game, providing a GUI for message input, and controlling the difficulty level settings.

1(b). Main.py: This is the main script that initializes the game, sets up the game window, and controls game flow. It also manages the user interface and game logic.

2. GameSettings.py: This file contains constants and settings used throughout the game, suchas screen resolution, colors, and tile size. It defines the game's key parameters and designelements.

3. TetrisMain.py: This file houses the core components of the Tetris game, including the Tetrominoes, their behavior, and game logic. It handles game dynamics and updates the game state.

4. TetrominoBlock.py: This script defines the Tetromino blocks, their rotation, movement, and collision detection. It includes the logic for rendering and animating these blocks.

5. Text Class: This class is responsible for rendering text on the game screen, such as the encrypted message, decrypted message, score, and time remaining. It also handles the cryptographic aspects of the game.

Packages used

The selection of appropriate packages is crucial in the development of a cryptographic Tetris game. Pygame was chosen as the primary game development library due to its ease of use, cross-platform compatibility, and extensive resources. pygame_gui complements Pygame by simplifying the creation of graphical user interfaces within the game. Additionally,

pygame.freetype and cryptography.fernet serve specific roles in rendering text and handling

cryptographic functionalities, respectively. These libraries work together harmoniously to provide a seamless and engaging gaming

experience.

1: Pygame:

Pygame is a fundamental library for developing 2D games in Python. It offers various modules and functionalities for game development, including graphics, sound, and event handling.

Pygame was chosen as the primary framework for the game due to the following reasons:1. Ease of Use: Pygame is known for its simplicity and beginner-friendly nature. It provides a straightforward and accessible API, making it an excellent choice for developers of all skill

levels.

Cross-Platform Compatibility: Pygame is cross-platform, supporting Windows, macOS, and Linux, ensuring that the game can run on a wide range of operating systems.

Abundance of Resources: Pygame has a thriving community, with numerous tutorials, forums, and resources available. This makes it easier to find solutions to common development challenges**.**

Extensive Documentation: Pygame offers comprehensive documentation, which simplifies the learning process and enables efficient development.

Game Development Features: Pygame provides essential features for game development, including handling user input, managing game events, and rendering graphics.

2: cryptography:

The `cryptography` library is utilized for implementing cryptographic functionalities within the game.

Key features include:

1. Public Key Encryption: `cryptography` supports public key encryption, enabling the encryption

of

user-inputted messages using a public key.

2. Padding and Hashing: The library provides padding and hashing algorithms necessary for

secure

encryption and decryption processes.

3. Backend Support: It off

Functionality Achieved:

- User inputs a message that is encrypted using a public key.

- The encrypted message is displayed within the game.

- The player controls Tetrominoes in a Tetris-like game environment.

- The player's score is tracked, and full lines are cleared for points.

- A timer counts down to create a sense of urgency.

- The player must reach a specified score within the time limit to decrypt the message.

- The user selects a difficulty level (1 to 3), affecting the required score (Level 1: 1200, Level 2:

1400, Level 3: 2000).

- Progression in the game occurs through milestones and reaching the required score to decrypt

the message.

cryptography:

The cryptography library is utilized for implementing cryptographic functionalities within the

game.

Key features include:

1. Public Key Encryption: cryptography supports public key encryption, enabling the encryption ofuser-inputted messages using a public key.

2. Padding and Hashing: The library provides padding and hashing algorithms necessary for secureencryption and decryption processes.

3. Backend Support: It offers a backend system that facilitates the use of specific cryptographic Implementations.

Tests Conducted

The test conducted for the cryptographic Tetris game primarily focused on ensuring the game's functionality, educational aspects, and entertainment value. Here's an explanation of the test

process and its results:

1. Functionality Test:

- Checked if the game runs smoothly without crashes or errors.

- Verified that the dynamic scoring system generates a required score that is always divisible by 100.

- Tested the countdown timer to ensure it starts at the expected duration (30 seconds) and triggers the "game over" state if time runs out.

- Verified that the cryptographic elements, including encryption and decryption, function correctly.

2. Educational Aspect Test:

- Evaluated the game's ability to introduce players to the concept of encryption through the encrypted message.

- Assessed how well the game educates players about achieving a specific score within a time limit to decrypt the message.

3. Entertainment Test:

- Evaluated the overall gaming experience to ensure it is engaging and fun.

- Assessed the level of challenge presented by the dynamic scoring and countdown timer.

- Checked if players found the game entertaining and if it encouraged them to replay for higher scores.

GamePlay

1. Game Initialization:

- The game is initialized in the GameStateManager class's constructor.

- Pygame is initialized with the window's dimensions and a clock for controlling frame rate.

2. Loading Game Assets:

- Game assets, including images for Tetrominos, are loaded in the load_images method.

- These images are stored in a list for later use.

3. Cryptographic Elements:

- The game uses the cryptography library to implement asymmetric encryption.

- A public-private key pair is generated, and a message is encrypted using the public key in

the Text class.

- The encrypted message is stored for later display when the player achieves a certain score.

4. Tetris Gameplay:

- Tetris gameplay is handled in the Tetris class.

- Tetrominos are represented as blocks, and the game field is managed as an array.

- Tetrominos are controlled by the player using keyboard inputs.

5. Scoring and Countdown Timer:

- The game keeps track of the player's score, the number of full lines cleared, and the points

earned for each line.

- A dynamic scoring system ensures that the required score is always divisible by 100.

- The countdown timer starts at 30 seconds and ticks down in the update method.

6. Check for Game Over:

- The game checks if the player has lost by running a function named is_game_over.

- If the top block of the current Tetromino reaches the initial position, it's game over, and the

player must restart the game.

7. Drawing the Game:

- The game field and Tetrominos are drawn in the draw method of the Tetris class.

- The game's grid is also drawn to provide visual guidance to the player.

8. Handling User Input:- User input is handled in the check_events method.

- Keyboard input is used to control the Tetrominos' movement and rotation.

9. Countdown Timer

Display:- The remaining time is displayed on the game screen using the timer_font in the Tetris class.

10. Game Over Logic:-If the player's score reaches the required score within the time limit, the encrypted_message is displayed to the player in the Text class.

11. Quitting the Game:-If the player decides to quit or the game ends due to a loss, the game is exited gracefully.

12. Pygame GUI Integration: The game incorporates Pygame GUI to create a "game over" window with two buttons: one

to restart the game and the other to quit. These buttons are enabled when the game ends, allowing the player to make a choice.

13. Handling Game Over Window Events: -The handle_events method in the App class is responsible for enabling buttons in the

"game over" window.

14. Restart and Quit Game Functions: Two functions, restart_game and quit_game, are called when the respective buttons in

the "game over" window are clicked.

- restart_game resets the game, and quit_game exits the game.

15. Dynamic Required Score:

- The required score is determined dynamically using a random number between 100 and

1000, ensuring that it is always divisible by 100.

16. Game Over Display: -When the game is over, the encrypted_message is displayed, showing the decrypted

message if the player meets the required score within the time limit.

17. Game Loop:- The game loop, present in the run method of the App class, continually checks for events,

updates the game state, and redraws the screen until the timer runs out or the player quits.18. User Interface Elements:

- User interface elements are drawn using Pygame's Font class in the Text class.

- The game keeps the player informed of the required score and their current score.This code successfully integrates Tetris gameplay with asymmetric cryptography concepts,creating an educational and entertaining experience for the player. The use of dynamic

scoring,countdown timers, and Pygame GUI elements enhances the game's complexity andengagement.