Cryptography and IT-Security - Coursework 2

Due 11am, 3rd April 2017

Anja Strobel (923886)
Martin Damov (877929)

Part 1 Implement Merkle's Puzzle

Part 2 Demonstrate Merkle's Puzzles Working

Part 3 Programming Style and Documentation

Swansea University College of Science

Question 1 Puzzle Generation

Question 1: Puzzle Generation

Methods for the generation of puzzles can be found in the *PuzzleGenerator.java* class. The generation is implemented in a public function which the user calls. This function is called *generatePuzzle()*. All relevant information is displayed and returned to the user.
Further documentation and comments can be found in the implementation.

Question 2: Puzzle Storage

The storage of puzzles is also dealt with in the *PuzzleGenerator.java* class, namely in the private *writeToFile()* method. Here the puzzles are written to a .txt file. The path to this file is passed in the class constructor.

Question 3: Puzzle Cracking

Puzzle cracking is implemented in the *PuzzleCrack.java* class. Here a file containing puzzles is opened and a random puzzle is cracked using a brute force approach. Each generated key is tested before another key is generated, therefore only the minimal amount of keys is created. If a decryption is successful all information is displayed and saved in a variable. A further method *getKey()* allows to retrieve the information.

Question 4: Key Lookup

The key lookup functionality is part of the *PuzzleGenerator.java* class. This means that the creator of the puzzle only has to use one class for all functionalities. The lookup information is stored in a list of bytes and retrieved by iterating over the list.

Part 2 Demonstrate Merkle's Puzzles Working

Question 5: Simulation

A simulation of the created classes can be found in *MerklesPuzzle.java*. Here the exchange between two parties is simulated by printing their actions to the screen. An additional class *DESEncryptionModule.java* was generated in order to provide message encryption/decryption functionalities. Running the main method displays the following:

```
ALICE: I am generating puzzles using the PuzzleGenerator.
--PuzzleGenerator: Puzzles generated sucessfully, encrypted and saved to specified
file.
ALICE: I am sending my puzzles to BOB.
BOB: I have recived a file containig puzzles from Alice.
BOB: I am using PuzzleCrack to crack a random puzzle.
--PuzzleCrack: The puzzle number is 423.
--PuzzleCrack: The secret key is Rfs4gPg0GqQ=. Keep this key secret. Do not
show it to anyone!
BOB: I am sending the puzzle number to Alice.
ALICE: I have recieved a message from Bob containing 423.
ALICE: I am using the lookup function of the PuzzleGenerator to get the secret
key.
--PuzzleGenerator: Key lookup sucessful. The secret key is Rfs4gPg0GqQ=.
ALICE: I am using the DESEncryptionModule to send a secret message to Bob.
ALICE: I will encrypt the following message and send it to Bob:
HelloBob
BOB: I have recieved this encrypted message from Alice.
yKqJuI7hcwA=
BOB: I am decrypting the message using the DESEncryptionModule and the key from
the puzzle
BOB: The message reads:
HelloBob
```

Part 3 Programming Style and Documentation

Question 6: Programming Style

The class structure was set up in a way that gives each party only one class to implement for solving Merkle's Puzzle and an additional class for DES encryption. Furthermore a simulation class was generated to show users a successful implementation.
It was chosen to simply print all relevant information to the screen, including errors and exceptions. This would have to be changed if the code was to be distributed as a library.
A JavaDoc was added, giving detailed description of each method. Further comments were added in the code to give a quick overview of implementation details.