

Design Document for JScrypt

Jean Lucas Ferreira, Ocean Cheung, Harit Patel

December 7, 2015

Contents

1	Introduction	3
2	Anticipated and Unlikely Changes	3
2.1	Anticipated Changes	3
2.2	Unlikely Changes	3
2.3	Revision History	4
3	Module Hierarchy	4
4	Connection between requirements and design	5
5	Module Decomposition	5
5.1	Hardware Hiding Modules	5
5.2	Behaviour Hiding Modules	5
6	Uses Relation	6
7	Schedule	7
7.1	Pert Chart	7
7.2	Gantt Chart	7
8	MIS for JScript	7
8.1	Methods	7
9	MIS for eksBlowfish	9
9.1	eksblowfish	9
10	Source Code	9
11	Implementation	9

Design Document

1 Introduction

This document is dedicated to documenting the the module interface specification (MIS) of JScript. It is intended to represent the the systems architectural design and detailed design, and describe the levels of interaction and hierarchy of modules throughout the project. The creation of this project will facilitate developers and testers to create test cases, and find modules that can potentially violate certain design principles.

2 Anticipated and Unlikely Changes

The system is susceptible to many changes and have been categorized by their likelihood as anticipated changes and unlikely changes.

2.1 Anticipated Changes

Anticipated changes to JScript mainly include addition of more preferences/options that users may be able to integrate into their applications. Perhaps slight changes in encryption or decryption algorithms may be made to improve performance.

AC1: Addition or changes to current options available to the users.

AC2: Changes in encryption/decryption algorithms

2.2 Unlikely Changes

Constants used in the system will unlikely be changed. These values are common throughout various implementations of bCrypt. Changing some of these values may result with an inconsistent implementation, and will no longer be following the bCrypt standard. Examples of these constants are:

- Salt string length
- P-Arrays and S-boxes values
- Minimum, maximum, and default rounds value

Also, the implementation of Eksblowfish in the project will unlikely be changed in terms of data structures, function signatures, and return types.

UC1: Constants: Salt String length, P-Array and S-boxes values, minimum and maximum rounds value.

UC2: Implementation of EksBlowfish algorithm in terms of data structure, function signatures, and return types.

2.3 Revision History

Table 1: Revision History

Date	Revision #	Authors	Description
September 21	0	All members	A Problem Statement
September 28	0	All members	Proof of Concept Plan
October 5	0	All members	Software Requirements Specification
October 19	0	All members	Test Plan
October 26	0	All members	Proof of Concept Demonstration
November 2	0	All members	Design Document
November 27	1	All members	Software Requirements Specification
November 27	1	All members	Test Plan
November 27	1	All members	Design Document
December 6	1	All members	Software Requirements Specification
December 6	1	All members	Design Document

3 Module Hierarchy

The different design modules of JScript will be defined in this section. The modules listed below will be implemented into the JScript project.

Level 1	Level 2
Hardware-Hiding Module	N/A
Behaviour-Hiding Module	Key Expansion Module Feistel Cipher Module Feistel F Module
Software Decision Module	Generate Random Salt Module Hash Key Module Get Components Module Compare Key Module

M1: hashKey

M2: compareKey

M3: getComponents

M4: generateRandomSalt

M5: keyExpansion

4 Connection between requirements and design

Due to some of the non-function, requirements specified in the Software Requirements Specification document, certain design decisions were made:

1. To fulfill the ease of use requirement, the library will only allow the user access to two methods. By doing so, the user is only responsible for encrypting and storing the input string.
2. The API of the project will be provided in the git repository (through readme.md), explaining how to get started with the project, and what can be done with it. This will help satisfy the ease of use, and learning requirements of the project.
3. The user will be provided an option to input the number of rounds the encryption algorithm should perform on the input string. This will satisfy the speed and latency requirements as the user will be able to choose the speed at which the algorithm is completed.
4. The library will only be in use each time the library functions are called and does not make changes to the application in question. Additionally, the library will be made open-source to all developers. This will fulfill the reliability and availability requirements.

5 Module Decomposition

5.1 Hardware Hiding Modules

Not Applicable for this project

5.2 Behaviour Hiding Modules

Module Name: JScrypt

- **Secret:** How the hash string is constructed/de-constructed
- **Service:** Allows the program to be instantiated for creating a hash string, or for comparing a hash string

Module Name: eksBlowfish

- **Secret:** Encryption algorithm for a string
- **Service:** Allows a key and salt (of string formats) to be encrypted, and creates a representation of the salt and key as a large string made up of ASCII characters

6 Uses Relation

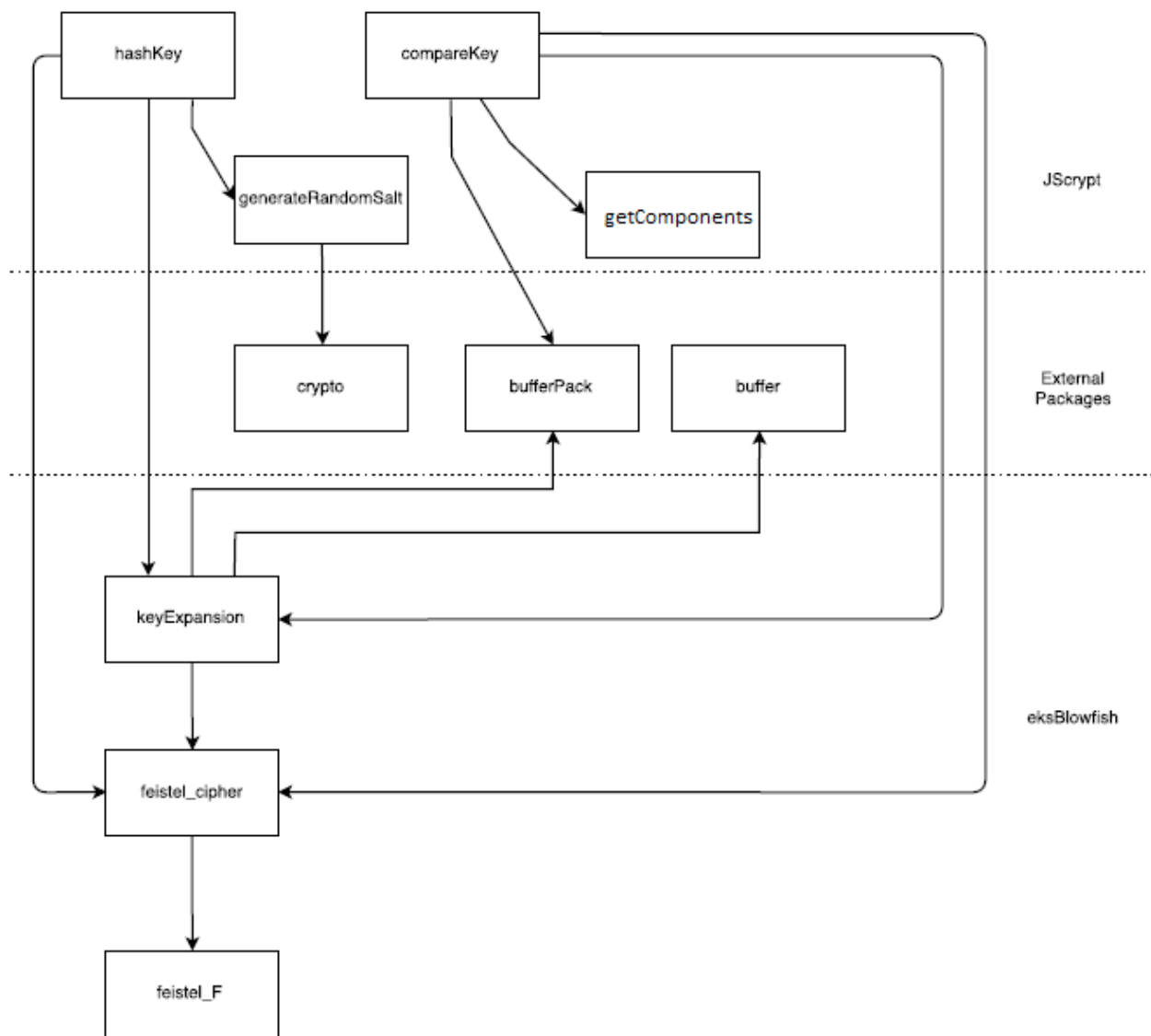


Figure 1: Uses Relation

7 Schedule

7.1 Pert Chart

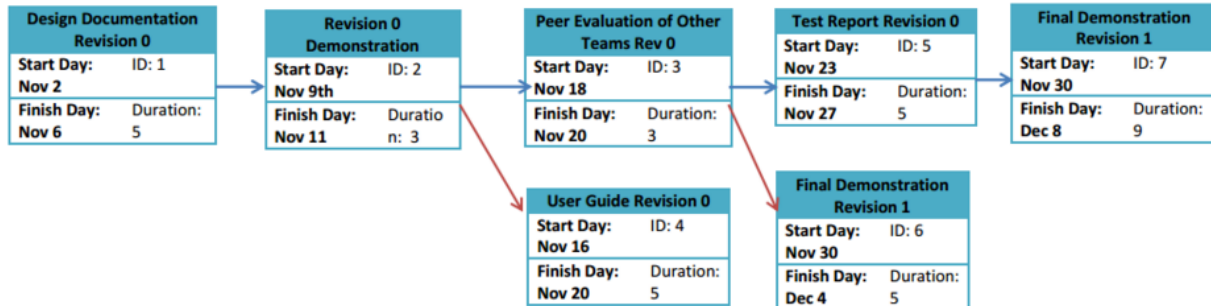


Figure 2: Pert Chart

7.2 Gantt Chart

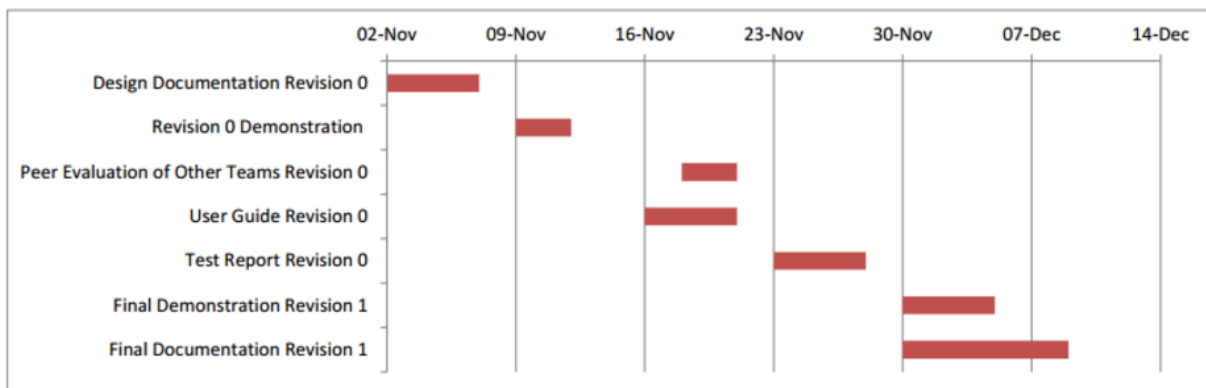


Figure 3: Gantt Chart

8 MIS for JScript

8.1 Methods

compareKey(cleanKey, hashKey)

Encrypts the string 'cleanKey' and compares to see if the cleanKey is the same as hashKey after the encryption process.

Parameters:

Name	Type	Description
cleanKey	string	plain text string that needs to be compared with hashKey
hashKey	string	hashed string generated by the program which consists of version, number of rounds, salt, and encrypted key.

Source: JScript.js, line 151

generateRandomSalt(rounds)

Generates a random padded string of length 24, which is used for hashing the key. This string includes padding which will later be removed before hashing the key.

Parameters:

Name	Type	Description
rounds	integer	Number of times the key is hashed.

Source: JScript.js, line 41

getComponents(hashKey)

A String can go through a series of rounds to become hashed and finding the number of rounds in the components gives us the knowledge required to compare a string of plaintext to a hashed string.

Parameters:

Name	Type	Description
hashKey	string	string generated by the encryption process with information containing the version, number of rounds, salt, and encrypted key.

Source: Jscript.js, line 209

hashKey(rounds, key)

Generates the hash string(refer to section 3.4) that will be stored in the applications database. This is the only function that a user of the project will be required to call in order to hash a string to encrypt information.

Name	Type	Description
rounds	integer	integer input from 6 to 31
key	string	string input of length 1 to 56 characters

Source: JScript.js, line 78

9 MIS for eksBlowfish

9.1 eksblowfish

eksBlowfish implementation

Properties:

Name	Type	Description
keyExpansion	function	Sets up the encryption environment with the given key by setting up the P_arrays and the S_boxes with hexadecimal values of the key
feistel_cipher	function	The heart of the encryption. Runs through a feistel network of $2^{(numberofrounds)}$ times. At each iteration, the P_arrays and S_boxes are updated according to the salt and the key.
feistel_F	function	Helper function of the Feistel_cipher

Source: eksBlowfish.js, line 15