Software Requirement Specifications for JScrypt

Jean Lucas Ferreira, Ocean Cheung, Harit Patel

December 7, 2015

Contents

Software Requirements Specification

1 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
November 27	1	All members	Software Requirements Specification
December 6	1	All members	Software Requirements Specification

Project Drivers

2 The Purpose of the Project

The purpose of this project, is to re-implement bCrypt, and allow web developers to safely store sensitive user information in a database. Once implemented correctly, the stored information should be seemingly impossible to decipher through brute force, independently of the power of the computer, and will also keep up with Moores Law, naturally by its design.

3 The Client, the Customer and other Stakeholders

3.1 The Client

Developers wishing to encrypt sensitive information about the users of their web application.

3.2 The Customer

The users of web applications that implemented the JScrypt library. These applications could be storing information such as passwords, credit card numbers, social insurance number, et cetera.

3.3 Other Stakeholders

- Contributors: Developers interested in the development process of the project, who are willing to share their knowledge in order to develop a better library.

- Testers: These may be developers, or regular clients who will test the projects functionality and usability.
- Security and Encryption Specialist: Building an encryption algorithm can be very complex, and may be prone to loopholes for hackers if specific procedures are not taken. Therefore a specialist would be a great investment for the project in order to verify its security.

4 Users of the Product

4.1 The Hands-On Users of the Product

Developers with minimal experience with NodeJS can follow the documentation and use the library to encrypt and decrypt passwords.

4.2 Priorities Assigned to Users

- The primary users of this library are web developers that wish to protect sensitive information by encrypting and decrypting sensitive information.
- The secondary users include users of application that utilize the library, as well as contributors and testers that may add on to the library.

4.3 User Participation

- User installs and imports the library into their application.
- User provides sensitive information to be encrypted
- User may compare text with encrypted password to check for authorization

4.4 Maintenance Users and Service Technicians

Maintained by the development team, testers and contributors.

Project Constraints

5 Mandated Constraints

5.1 Solution Constraints

- **Description:** Project must be completed by November 9, 2015
  Rationale: Date for which Revision 0 demonstration is due.

Fit Criterion: N/A

- **Description:** The library must operate in the same manner, regardless of operating system, and web browser.
  Rationale: DDevelopers wishing to use the library shouldnt have to change their developing environment. Rather all they require is NodeJS to be installed on their system. Similarly, the users of the library shouldnt have to use specific web browsers in order to have their information safely stored.
  Fit Criterion: The testers will test the library under several web browsers and operating systems to assure portability.

5.2 Implementation Environment of the Current System

The project will be entirely written in JavaScript, within the NodeJS runtime environment.

5.3 Partner of Collaborative Applications

N/A.

5.4 Off-the-Shelf Software

There exists many implementations of bCryot, in multiple languages, which can be used as reference.

5.5 Anticipated Workplace Environment

- On any computer system (Windows, Linux, Mac), or mobile device.
- Any web browser with javascript enabled (Chrome, Mozilla Firefox, Safari, IE, Opera).

5.6 Schedule Constraints

- Test Plan Revision 0: October 5, 2015
- Proof of Concept Demonstration: October 26, 2015
- Design Document Revision 0: November 2, 2015
- Revision 0 Demonstration: November 9, 2015

5.7 Budget Constraints

There is no budget constraint for this project, and JScrypt will be open source and completely free once completed.

6 Naming Conventions and Definitions

6.1 Definition of all Terms

- **NodeJS:** An open-source runtime environment for developing server-side applications in JavaScript.
- **Key:** A key is a string of characters to be encrypted, they are sensitive information given by the user.
- **Blowfish:** An algorithm that uses block ciphers to encrypt/decrypt keys.
- **Eksblowfish:** Expensive Key Schedule blowfish, the main algorithm of bCrypt, which is an extension of blowfish that allows the addition of a salt, and cost parameters.
- **Encryption:** Encoding passwords into an incomprehensible string of ASCII characters.
- **Decryption:** Decoding an Encrypted password to retrieve the original provided password.
- **Plaintext:** A key given to the encryption algorithm that needs to be encrypted.
- **Ciphertext:** A representation of the plaintext, but is an illegible string.
- **Brute Force:** A trial and error method used by applications to decode encrypted data.
- **Open Source:** Software that can be used, changed, and shared by anyone.
- **String:** A sequence of characters.
- **Database:** A collection of data that is organized to allow it to be easily accessed, managed and updated.
- **Runtime Environment:** A configuration of hardware and software that is used to run certain applications.
- **Library:** A collection of functions and software packages that can be imported into certain applications.
- **NPM:** Node Package Manager, a sytem that allows installation of node libraries through a command line interface.

6.2 Data Dictionary for Any Include Models

<u>input</u>

- **Key** (String): The information required to be encrypted
- **Cost** (Integer): The work factor of the algorithm, the larger the number, the slower the algorithm will be, but it will be more difficult on brute-force attacks.
- **Salt** (String): A unique salt is given to each stored key, it is used in the encryption process.

<u>output</u>

- **EncryptedKey** (String): e encryptedKey is a concatenation of the cost, salt (base64 encoded) and ciphertext (base64 encoded) of the key provided by the user. The encryptedKey will be outputted to the application, which can then be stored in a database, as an example.

7 Relevant Facts and Assumptions

7.1 Facts

- Eksblowfish is a purposely slow algorithm.
- Powerful hardware does not decrease the security against brute force attack

7.2 Assumptions

- Developers using this library will have a basic understanding of web development with JavaScript and NodeJS.
- Web applications using this library will be published on a server that can support NodeJS applications.

Functional Requirements

8 The Scope of the Work

8.1 The Current Situation

JScrypt is a library for NodeJS designed to provide reliable, user-friendly, and robust encryption functionality. This will mainly be done by using an encryption algorithm called Eksblowfish. The library will be tremendously simple to use for developers as it will only require the user to provide a password to encrypt and then store an encrypted password provided by the library. Likewise, a developer can easily compare a non-encrypted password with an encrypted one by simply using a function provided by the library. Additionally, many of the current encryption algorithms are prone to brute-force attacks while the Eksblowfish algorithm is mostly impervious.

8.2 The Context of the Work

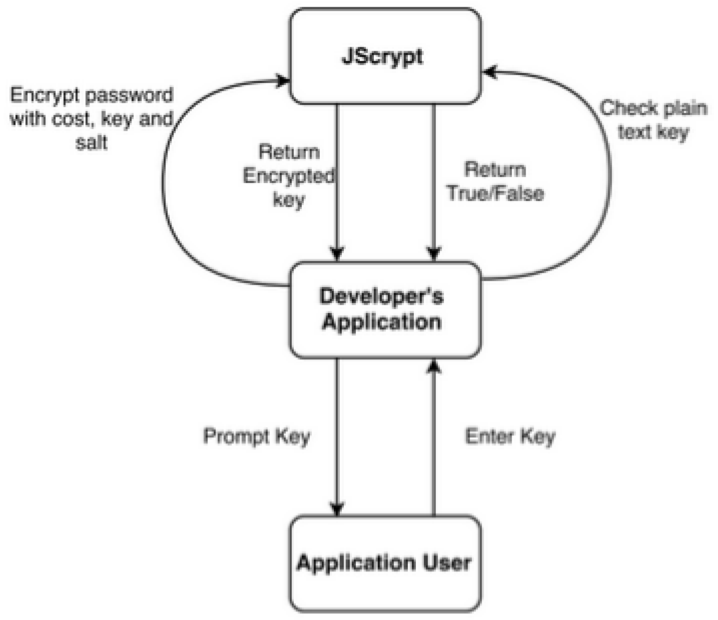

Figure 1: Context of the Work

8.3 Work Partitioning

Refer to Table 1

Table 2: Work Partitioning

Event	Input/Output	Summary
1. App prompts key	none	Application asks user to enter a password, credit card number, or any other sensitive information
2. User enters key	plainTextKey(input)	User sends key to the application
3. App verifies key	validKeyCheck(output)	check if key provided by the user meets all criterias required
4. App encrypts key	key(output), cost(output),salt(output), encryptedKey(input)	App provides library with key, cost and salt to encrypt the key. Encrypted key is returned
5. Store key	none	App stores key in a desired database
6. Compare Key	plainTextKey(input), authorized(output)	JScrypt determines whether the plainTextKey matches an encrypted key

9 Scope of the Product

9.1 Product Boundary

Refer to Figure 1

9.2 Product Use Case

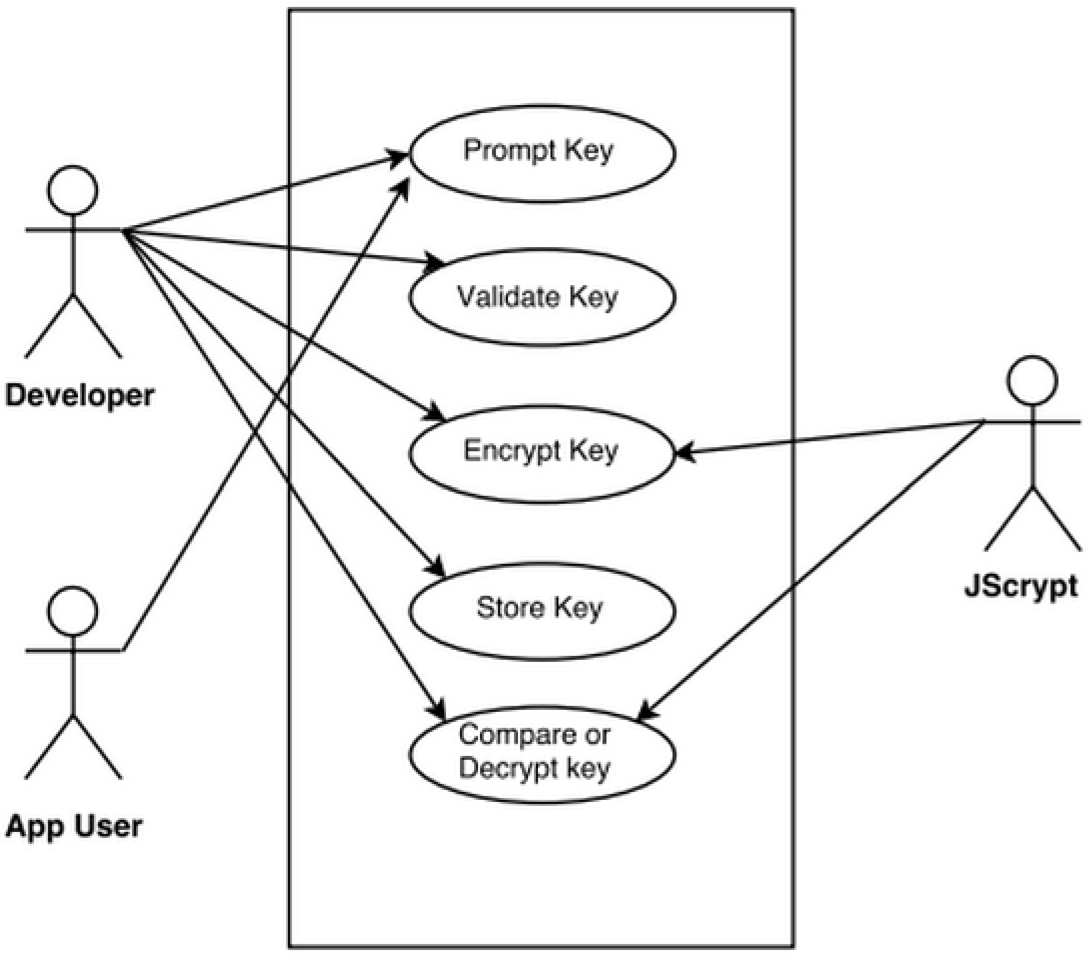

Figure 2: Uses Case Diagram

10 Fuctional and Data Requirements

10.1 Functional Requirements

Requirement #: 1
Requirement Type: 9

Event/use case: 2,4
Description: The product shall be able to accept a string and encrypt it.
Rationale: Information stored in plain text is vulnerable if the information in a database was breached. Encrypting the information adds an extra layer of security for protect sensitive details.
Fit Criterion: No unauthorized applications can decrypt the encrypted string.

Requirement #: 2
Requirement Type: 9
Event/use case: 2,4
Description: The product shall be able to accept a string and encrypt it.
Rationale: Information stored in plain text is vulnerable if the information in a database was breached. Encrypting the information adds an extra layer of security for protect sensitive details.
Fit Criterion: No unauthorized applications can decrypt the encrypted string.

Requirement #: 3
Requirement Type: 9
Event/use case: 6
Description: The product shall be able to verify an encrypted string and a plain text string for equality.
Rationale: The product needs to be able to confirm information between encrypted strings and plain text strings without decrypting the encrypted string.
Fit Criterion: No unauthorized applications can decrypt the encrypted string.

Requirement #: 4
Requirement Type: 9
Event/use case: 4
Description: The product shall create an encrypted string which cannot be brute forced easily.
Rationale: Powerful hardware is capable of brute forcing through some encryption methods such as MD5. The eksblowfish algorithm scales with Moores Law and is intentionally slow which deters users from brute forcing the JScrypts encrypted strings.
Fit Criterion: No unauthorized applications can decrypt the encrypted string easily through a brute force method.

Requirement #: 5
Requirement Type: 9
Event/use case: 3

Description: The key provided by the user must not exceed 448 bits, which is equivalent to 56 ASCII characters.
Rationale: By the way Eksblowfish and Blowfish is designed, the restriction on the key size ensures each bit of the subkeys is dependent on each bit of key.
Fit Criterion:A key size must be checked prior to encryption or decryption.

Non-Functional Requirements

11 Look and Feel Requirements

11.1 Appearance Requirements

N/A

11.2 Style Requirements

N/A

12 Usability and Humanity Requirements

12.1 Ease of Use Requirement

Library should be easily implemented in any application, by developers with basic experience in NodeJS and JavaScript. Access to this library should be made intuative through either GitHub or NPM (which will require the developer to have NodeJS installed on their system).

12.2 Personalization and Internationalization Requirements

There is no language barrier with using the library, but documentation will be written in english.

12.3 Learning Requirements

Any guidance required to use this library will be provided in the project documentation.

12.4 Understandability and Politeness Requirements

Users are only required to understand the implementation of the provided API functions. They are not required to understand the design.

12.5 Accessibility Requirements

N/A

13 Performance Requirements

13.1 Speed and Latency Requirements

- The speed of the algorithm is directly proportional to the cost parameter given by the user.
- It is meant to be purposely slow.

13.2 Safety-Critical Requirements

N/A

13.3 Precision or Accuracy Requirements

N/A

13.4 Reliability and Availability Requirements

- The library will be made open-source to all developers.
- The library will only be in use each time the library functions are called and does not make changes to the application in question.

13.5 Robustness or Fault-Tolerance Requirements

If the developers accidently sends the wrong information to be encrypted/decrypted by JScrypt, there is no way of catching this error, and it is solely up to the developer to assure the information is consistent.

13.6 Capacity Requirements

There is no capacity to number of users using the library and the number of times the library is used.

13.7 Scalability or Extensibility Requirements

N/A

13.8 Longevity Requirements

N/A

14 Operational and Environmental Requirements

14.1 Expected Physical Environment

Application running on NodeJS will be required to run the library

14.2 Requirements for Interfacing with Adjacent Systems

The library will be available for any operating system

14.3 Productization Requirements

N/A

14.4 Release Requirements

The library is open source, thus it is always under development and available for use.

15 Maintainability and Support Requirements

15.1 Maintenance Requirements

Since the project is open sourced to the public, suggestion for improvements will be taken through pull request.

15.2 Supportability Requirements

The library will not be constantly supported, nonetheless the source code will be available for examination and improvements by the public.

15.3 Adaptability Requirements

N/A

16 Security Requirements

16.1 Access Requirements

- Access to the librarys source code is open to the public.
- Developers will have access to make changes to a local version of the project. Any changes to the live version of the project must be submitted through a pull request which may or may not be accepted.

16.2 Integrity Requirements

N/A

16.3 Privacy Requirements

The library shall not reveal, store or send any sensitive information that is provided by the user.

16.4 Audit Requirements

N/A

16.5 Immunity Requirements

N/A

17 Cultural and Political Requirements

17.1 Cultural Requirements

- The library in no way implies any cultural offense.
- Due to the way the algorithm is implemented, only ASCII character keys are allowed.

17.2 Political Requirements

N/A

18 Legal Requirements

18.1 Compliance Requirements

JScrypt must follow the same algorithmic structure for encrypting and decrypting as Eks-blowfish in order to satisfy the security of the key encryption.

18.2 Standards Requirements

JScrypt must comply with all encryption standards

Project Issues

19 Open Issues

There are no issues to be considered at this stage of the development process.

20 Off the shelf solutions

There exists many implementations of JScrypt, in multiple languages, which can be used as reference.

21 New Problems

JScrypt is simply taking a string parameter, manipulating the string and returning an output. This process should not create any new problems for users of this library.

22 Tasks

- Requirements document revision 0
- Test plan revision 0
- Proof of concept demonstration
- Design document revision 0
- Revision 0 demonstration
- User's guide revision 0
- Test report revision 0
- Write final revisions to documentation

23 Migration to the New Product

N/A

24 Risks

In the event of a security breach, developers may hold the creators of JScrypt liable for creating an insecure encryption method.

25 Costs

There are no costs associated with the development and use of this project. There are indirect costs which developers will have to pay for if they choose to use a server hosting service to deploy the application with.

26 User Documentation and Training

Users will be able to download JScrypt through the public repository it will be uploaded onto. There will also be a readme markdown file describing how to use the different JScrypt APIs.

27 Waiting Room

- Future releases may have documentation in other spoken languages.
- JScrypt could possibly be implemented to directly connect with a database system.

28 Ideas for Solutions

- Users of JScrypt will be requested to translate the documentation in any language they are comfortable with.
- Create functionality to connect to a database and store encrypted password directly. Could possibly accomplished by other developers contributing to the library.