

Software Requirements for JScript

Jean Lucas Ferreira, Ocean Cheung, Harit Patel

October 9, 2015

Contents

1	The Purpose of the Project	4
2	The Client, the Customer and other Stakeholders	4
2.1	The Client	4
2.2	The Customer	4
2.3	Other Stakeholders	4
3	Users of the Product	4
3.1	The Hands-On Users of the Product	4
3.2	Priorities Assigned to Users	4
3.3	User Participation	5
3.4	Maintenance Users and Service Technicians	5
4	Mandated Constraints	5
4.1	Solution Constraints	5
4.2	Implementation Environment of the Current System	5
4.3	Partner of Collaborative Applications	5
4.4	Off-the-Shelf Software	6
4.5	Anticipated Workplace Environment	6
4.6	Schedule Constraints	6
4.7	Budget Constraints	6
5	Naming Conventions and Definitions	6
5.1	Definition of all Terms	6
5.2	Data Dictionary for Any Include Models	7
6	Relevant Facts and Assumptions	7
6.1	Facts	7
6.2	Assumptions	8
7	The Scope of the Work	8
7.1	The Current Situation	8
7.2	The Context of the Work	8
7.3	Work Partitioning	9
8	Scope of the Product	9
8.1	Product Boundary	9
8.2	Product Use Case	10
9	Functional and Data Requirements	10
9.1	Functional Requirements	10

10 Look and Feel Requirements	12
10.1 Appearance Requirements	12
10.2 Style Requirements	12
11 Usability and Humanity Requirements	12
11.1 Ease of Use Requirement	12
11.2 Personalization and Internationalization Requirements	13
11.3 Learning Requirements	13
11.4 Understandability and Politeness Requirements	13
11.5 Accessibility Requirements	13
12 Performance Requirements	13
12.1 Speed and Latency Requirements	13
12.2 Safety-Critical Requirements	13
12.3 Precision or Accuracy Requirements	13
12.4 Reliability and Availability Requirements	13
12.5 Robustness or Fault-Tolerance Requirements	13
12.6 Capacity Requirements	14
12.7 Scalability or Extensibility Requirements	14
12.8 Longevity Requirements	14
13 Operational and Environmental Requirements	14
13.1 Expected Physical Environment	14
13.2 Requirements for Interfacing with Adjacent Systems	14
13.3 Productization Requirements	14
13.4 Release Requirements	14
14 Maintainability and Support Requirements	14
14.1 Maintenance Requirements	14
14.2 Supportability Requirements	14
14.3 Adaptability Requirements	15
15 Security Requirements	15
15.1 Access Requirements	15
15.2 Integrity Requirements	15
15.3 Privacy Requirements	15
15.4 Audit Requirements	15
15.5 Immunity Requirements	15
16 Cultural and Political Requirements	15
16.1 Cultural Requirements	15
16.2 Political Requirements	15
17 Legal Requirements	16
17.1 Compliance Requirements	16
17.2 Standards Requirements	16

18 Open Issues	16
19 Off the shelf solutions	16
20 New Problems	16
21 Tasks	16
22 Migration to the New Product	17
23 Risks	17
24 Costs	17
25 User Documentation and Training	17
26 Waiting Room	17
27 Ideas for Solutions	17

Project Drivers

1 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.

2 The Client, the Customer and other Stakeholders

2.1 The Client

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

2.2 The Customer

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

2.3 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.

3 Users of the Product

3.1 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.

3.2 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.

3.3 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

3.4 Maintenance Users and Service Technicians

Maintained by the development team, testers and contributors.

Project Constraints

4 Mandated Constraints

4.1 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: Developers wishing to use the library shouldn't have to change their developing environment. Rather all they require is NodeJS to be installed on their system. Similarly, the users of the library shouldn't 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.

4.2 Implementation Environment of the Current System

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

4.3 Partner of Collaborative Applications

N/A.

4.4 Off-the-Shelf Software

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

4.5 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).

4.6 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

4.7 Budget Constraints

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

5 Naming Conventions and Definitions

5.1 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.

5.2 Data Dictionary for Any Include Models

input

- **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.

output

- **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.

6 Relevant Facts and Assumptions

6.1 Facts

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

6.2 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

7 The Scope of the Work

7.1 The Current Situation

JScript 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.

7.2 The Context of the Work

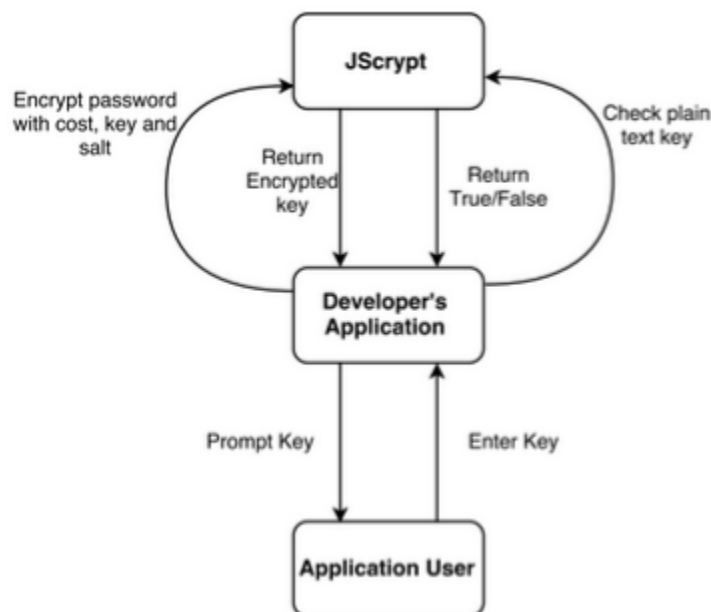


Figure 1: Context of the Work

7.3 Work Partitioning

Refer to Table 1

Table 1: 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)	JScript determines whether the plain-TextKey matches an encrypted key

8 Scope of the Product

8.1 Product Boundary

Refer to Figure 1

8.2 Product Use Case

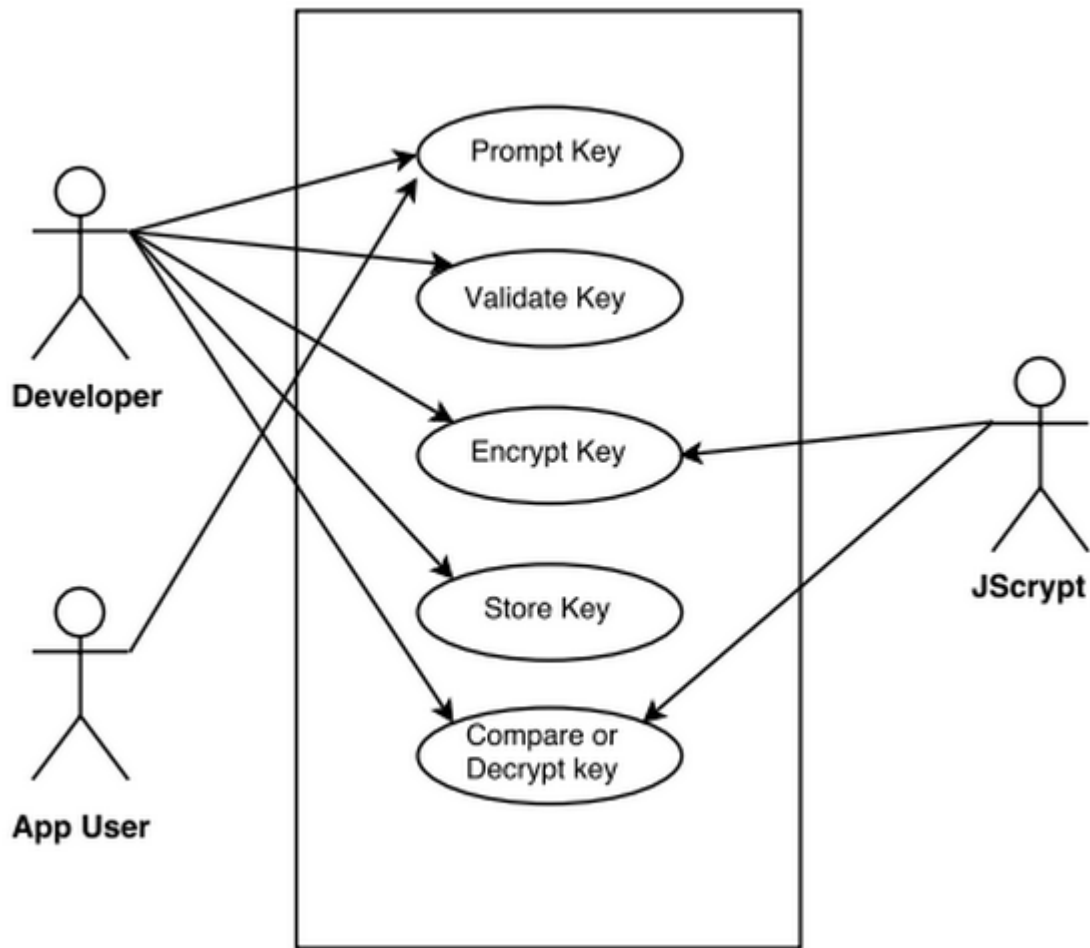


Figure 2: Uses Case Diagram

9 Fuctional and Data Requirements

9.1 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: 6

Description: TThe product shall be able to take an encrypted string and decrypt it to restore to plain text.

Rationale: Information needs to be decryptable so that it can still be used and read by authorized programs and users.

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 a plain text string and an encrypted string for equality.

Rationale: The product needs to be able to confirm information between plain text and encrypted strings without decrypting the encrypted string.

Fit Criterion: No unauthorized applications can decrypt the encrypted string.

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 #: 4

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 #: 5

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 JScripts encrypted strings.

Fit Criterion:No unauthorized applications can decrypt the encrypted string easily through a brute force method.

Requirement #: 6

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

10 Look and Feel Requirements

10.1 Appearance Requirements

N/A

10.2 Style Requirements

N/A

11 Usability and Humanity Requirements

11.1 Ease of Use Requirement

Library should be easily implemented in any application, by developers with basic experience in NodeJS and JavaScript.

11.2 Personalization and Internationalization Requirements

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

11.3 Learning Requirements

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

11.4 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.

11.5 Accessibility Requirements

N/A

12 Performance Requirements

12.1 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.

12.2 Safety-Critical Requirements

N/A

12.3 Precision or Accuracy Requirements

N/A

12.4 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.

12.5 Robustness or Fault-Tolerance Requirements

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

12.6 Capacity Requirements

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

12.7 Scalability or Extensibility Requirements

N/A

12.8 Longevity Requirements

N/A

13 Operational and Environmental Requirements

13.1 Expected Physical Environment

Application running on NodeJS will be required to run the library

13.2 Requirements for Interfacing with Adjacent Systems

The library will be available for any operating system

13.3 Productization Requirements

N/A

13.4 Release Requirements

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

14 Maintainability and Support Requirements

14.1 Maintenance Requirements

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

14.2 Supportability Requirements

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

14.3 Adaptability Requirements

N/A

15 Security Requirements

15.1 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.

15.2 Integrity Requirements

N/A

15.3 Privacy Requirements

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

15.4 Audit Requirements

N/A

15.5 Immunity Requirements

N/A

16 Cultural and Political Requirements

16.1 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.

16.2 Political Requirements

N/A

17 Legal Requirements

17.1 Compliance Requirements

JScript must follow the same algorithmic structure for encrypting and decrypting as Ekblowfish in order to satisfy the security of the key encryption.

17.2 Standards Requirements

JScript must comply with all encryption standards

Project Issues

18 Open Issues

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

19 Off the shelf solutions

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

20 New Problems

JScript 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.

21 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

22 Migration to the New Product

N/A

23 Risks

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

24 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.

25 User Documentation and Training

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

26 Waiting Room

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

27 Ideas for Solutions

- Users of JScript 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.