information using the JavaScript library. When Client-side encryption is enabled, an RSA keypair is generated and the user will be given a specially formatted version of the public key. RSA is the algorithm that is used to encrypt data with a private key to produce a digital signature. The private key, however, is never revealed to the user or anyone else. The data is decrypted using the keypair's private key.
Public and private keys are created simultaneously using the same algorithm (RSA- Rivest-Shamir-Adleman). Private keys are used to decrypt text that has been encrypted with a public key.
d)Decryption of data
When the user requests to read the data from the database, the web application will check user's credentials (if the session is active) and get the key from the server to allow decryption of data.
e)User Authentication
Upon successful authentication, the user will be given a public key, which will be used for the encryption/decryption of the data. This private key will be stored on the server side, with all the user information which is used for decrypt the data. We are going to use OAuth 2, which is an open standard for authorization. This will be used to securely transfer the private key from the server to the encryption library.
f)Deletion of data
Secure deletion of data will be required to overwrite the space of data with zeros. This means that the data cannot be read again, as all of the values are set to zero.
If running over HTTPS, then things are more secure as the browser will detect a modified JavaScript file. The SSL layer of HTTPS protocol handles this.

5.1. Proposal

Hashing and encryption can be done within browsers through the JavaScript encryption library. Algorithm will use a JavaScript encryption library (proposed Stanford JS Encryption Library), where the library will be implemented into the browser (Firefox) as an extension. This extension will be based on the top of IndexedDB API and therefore every time during the reading or writing of data, the data will be encrypted. The library consists of encryption with private and public keys. The private key will be saved on the server. The public key will be given to the user and stored on the user's machine, the same way as a cookie. The extension will provide encryption/decryption of data on the user's machine, which will resolve the issue of storing data in an unencrypted state.

5.2. Algorithm Used

It differs from typical AES implementations (different approach that keeps the code small and speeds up encryption/decryption). The source code for the AES algorithm, also called Advanced Encryption Standard or the Rijndael algorithm. The benchmarking tests have shown that the Stanford JS Encryption library performs faster than other client side encryption libraries. The benchmark has been achieved in multiple browsers on Windows, Mac and Linux Operating Systems. One of the reasons we proposed to use and implement the library into algorithm was the speed and multiplatform usage.

The algorithm is going to contain the JavaScript encryption library, which will be implemented into the browser. The algorithm will consist of a few steps, with the higher security. This will allow the end user to save and retrieve data from IndexedDB. The data will be encrypted with the JavaScript library and a private and public key will be used to encrypt/decrypt this data.

5.3. Implementation

The model will add an extra layer between the web browser and IndexedDB API. The security model consists of an algorithm framework, which adds extra protection against issues identified, by reading each other's data through XSS vulnerabilities.

Algorithm is using JavaScript encryption library (proposed Stanford JS Encryption Library), where the library is implemented into the browser (Firefox) as an extension. This extension is placed on the top of IndexedDB API and therefore every time during the reading or writing of data, the data will be encrypted. The library consists of encryption with private and public keys. As expected, the private key will be saved on the server. The public key will be given to the user and stored on the user's machine, the same way as a cookie. The extension will provide encryption/decryption of data on the user's machine, which will resolve the issue of storing data in an unencrypted state. It will also provide better security for possible attacks, where the attacker can manipulate with user data.

The browser based local storage security model (BBLS) is relying on the web browser security model (WBSM), which is using Same origin policy. The security mechanism is not enough to preserve the security confidence among the end user.

The BBLS security model differs from WBSM in few ways, which includes the security mechanism. The main difference is that BBLS security model is trying to secure the data between browser and the end user file system, where comparing to WBSM, which is securing the data between web applications and user browser.

The goal of BBLS security model is to secure the data, which is stored in client side database. User should be able to visits other websites, without they databases to be compromised.

The current WBSM is not sufficient protection for complex web application and stored data on client side is becoming more important.

The security model consists of an encryption framework, which will help to secure the data. The encryption framework cannot though provide full security protection. We argue that such protection is not achievable in a single machine, since any single browser could be the target of an XSS attack. Therefore, functionality external to be browser needs to be implemented. For implementation to existing encryption library we will use Multifactor authentication (MFA). MFA is used to make the authentication process more secure by adding an extra layer of security. The extra authentication will need to be passed to make sure the encryption library decrypts the data. Mobile two-factor authentication use phones to replace fobs or software-based tokens that were commonly used for remote authentication. When a person tries to log into an online service, a security pin is sent to his or her mobile phone via voice or SMS message, rather than to the token.

5.4. Evaluation

To evaluate the security model, we will run tests to conclude the effectiveness of the model. This will include attacks, which will be bypassing the SOP trough XSS attacks. First we will perform and attack with existing security, without applying the security model.

Then we will add the security model, and perform the attack again. We suggest that the model will prevent an attacker to read data from other source, by adding the authentication process to place. Also the data stored will be encrypted, which means that even the authentication process is compromised, the data will not be available to read in unencrypted state.

Based on our findings, we can state that there is a case for browser-based databases. We have implemented a JavaScript encryption framework, which is a part of the security model implemented into the browser in a form of an extension. The proposed security model extension addresses the security issue that IndexedDB has as a product of its design. Also, the implemented security model fulfils the security requirements.

6. Conclusion and Future Work

Based on these findings, we can state, that there is a case for browser-based databases. Browser based databases though face security problems over and above those on the server, and this has inhibited their uptake. Nevertheless, despite the existing issues faced by browser-based storage, there is a future for the technology due to its convenience, performance, reduced reliance on continuously available network connection.

Considering the issues and concerns of storing data locally, browser based storage has the potential to be widely used, where the main advantage is the performance speed, cross platform (desktop, mobile, tablet) and browser availability. The advantages of local storage outweighs the disadvantages, keeping in mind that the issues identified can be corrected and browser-based storage can be widely used by developers without any concerns of security issues introduced as by design limitations.

Although the proposed security framework has been successfully applied to browser based local storage, further improvements can be made in extending the security and performance model. These could be addressed by extending the current model to use further security factors such as biometrics.

7. References

[1] Naseem, S.Z. Majeed, F. (2013) Extending HTML5 local storage to save more data; efficiently and in more structured way. Eighth International Digital Information Management (ICDIM).

[2] Zhanikeev, M. (2013) A Practical Software Model for Content Aggregation in Browsers Using Recent Advances in HTML5. 37th Annual Computer Software and Applications Conference Workshops (COMPSACW). pp.151-156, Japan 22-26 July 2013

[3] Ryck, P. Desmet, L. Philippaerts, P. Piessens, F. (2011) A Security Analysis of Next Generation Web Standards, (European Union Agency for Network and Information Security - ENISA). Tech. Rep.

[4] Anttonen, M. Salminen, A. Mikkonen, T. Taivalsaari, A (2011) Transforming the web into a real application platform: new technologies, emerging trends and missing pieces. ACM Symposium on Applied Computing. New York, NY, USA. Pp. 800-807.

[5] Chuang, T. T., Nakatani, K, Chen, J. C. H. and Huang, I. L. (2007). Examining the Impact of Organisational and Owner's Characteristics on the Extent of E-commerce Adoption in SMEs,

[6] Pool, P. W., Parnell, J. A., Spillan, J. E., Carraher, S. and Lester, D. L. (2006). Are SMEs Meetings the Challenge of Integrating E-commerce into Their Businesses? A Review of the Development, Challenges and Opportunities, International Journal Information Technology and Management, 5(2/3), pp.97-113.

[7] Jones, J. (2014) E-commerce: measuring, monitoring and gross domestic product. ONS. Available at: http://www.ons.gov.uk/ons/rel/gva/national-accounts-articles/e-commerce--measuring--monitoring-and-gross-domestic-product/index.html (Accessed: 20 September 2015).

[8] Ta, H., Esper, T., & Hofer, A. R. (2015). Business-to-Consumer (B2C) Collaboration: Rethinking the Role of Consumers in Supply Chain Management. Journal of Business Logistics, 36(1), 133-134.

[9] Xiaojing, L., Liwei, Z., & Weiqing, W. (2012). The mechanism analysis of the impact of eCommerce to the changing of economic growth mode. In Robotics and Applications (ISRA), 2012 IEEE Symposium on (pp. 698-700). IEEE.

[10] Boritz, E., Gyun, W., and Sundarraj, P. 2008. Internet privacy in E-commerce: Framework, review and opportunities for future research. In: Proceedings of the 41st Hawaii International Conference on System Sciences. Hawaii, January 7-10 2008, pp.204-256.

[11] Gehling, B., & Stankard, D. (2005). eCommerce security. In Proceedings of the 2nd annual conference on Information security curriculum development (pp. 32-37). ACM.

[12] Gómez, J. M., & Lichtenberg, J. (2007). Intrusion Detection Management System for ECommerce Security. Journal of Information Privacy and Security, 3(4), 19-31.

[13] Buja, G., Jalil, K. B. A., Ali, F. B., Mohd, H., & Rahman, T. F. A. (2014). Detection model for SQL injection attack: An approach for preventing a web application from the SQL injection attack. In Computer Applications and Industrial Electronics (ISCAIE), 2014 IEEE Symposium on (pp. 60-64). IEEE.

[14] Appelt, D., Nguyen, C. D., Briand, L. C., & Alshahwan, N. (2014). Automated testing for SQL injection vulnerabilities: An input mutation approach. In Proceedings of the 2014 International Symposium on Software Testing and Analysis (pp. 259-269). ACM.

[15] Summers, S., Schwarzenegger, C., Ege, G., & Young, F. (2014). The emergence of EU criminal law: cyber crime and the regulation of the information society. Bloomsbury Publishing.

[16] Karthik, R., Patlolla, D. R., Sorokine, A., White, D. A., & Myers, A. T. (2014). Building a secure and feature-rich mobile mapping service app using HTML5: challenges and best practices. In Proceedings of the 12th ACM international symposium on Mobility management and wireless access (pp. 115-118). ACM.

[17] Ayenson, M. Wambach, D. J. Soltani, A. Good, N. Hoofnagle, C. J. (2011) Flash cookies and privacy II: Now with HTML5 and ETag respawning. Computer and Information Systems Abstracts. [Online]. Available at: http://dx.doi.org/10.2139/ssrn.1898390 (Accessed: 10 February 2015).

[18] Strozzi, C. (1998) NoSQL A Relational Database Management System. Available at: http://www. strozzi. it/cgi-bin/CSA/tw7/I/en_US/nosql/Home% 20Page (Accessed: 20 September 2015).

[19] Kimak, S. Ellman, J. Laing, C. (2014) Some Potential Issues with the Security of HTML5 IndexedDB. In: System Safety and Cyber Security 2014 (IET Conference), 14-16th October 2014, The Midland Hotel, Manchester, UK.

[20] Casario, M. Elst, P. Brown, Ch. Wormser, N. Hanguez,C. (2011) HTML5 Solutions: Essential Techniques for HTML5 Developers. Publisher: FRIENDS OF ED; 1 edition ISBN: 1430233869.