Implementation and Evaluation of a Secure and Efficient Web Authentication Scheme using Mozilla Firefox and WAMP

Yassine SADQI, Ahmed ASIMI, Younes ASIMI, Zakaria TBATOU,Azidine GUEZZAZ
Departments of Mathematics and Computer Science, Information Systems and Vision Laboratory (LabSIV), Faculty of Sciences, Ibn Zohr University B.P 8106, City Dakhla, Agadir, Morocco

***Abstract*— User authentication and session management are two of the most critical aspects of computer security and privacy on the web. However, despite their importance, in practice, authentication and session management are implemented through the use of vulnerable techniques. To solve this complex problem, we proposed new authentication architecture, called StrongAuth. Later, we presented an improved version of StrongAuth that includes a secure session management mechanism based on public key cryptography and other cryptographic primitives. In this paper, we present an experimental implementation and evaluation of the proposed scheme to demonstrate its feasibility in real-world scenarios. Specifically, we realize a prototype consisting of two modules: (1) a registration module that implements the registration, and (2) an authentication module integrating both the mutual authentication and the session management phases of the proposed scheme. The experimental results show that in comparison to traditional authentication and session management mechanisms, the proposed prototype presents the lowest total runtime.**

Keywords*— *authentication; session management; web security; cryptographic primitives; computer security and privacy; security implementation; authentication scheme; Mozilla Firefox; WAMP

I. INTRODUCTION (*Heading 1*)

Despite its widely studied security problems [1]–[13], password authentication through an HTML form is the dominant mechanism for authenticating users in modern web applications [4], [14], [15]. More specifically, the lack of authentication standard HTML form and the limited security background of webmasters have created a set of unique design and implementation choices that contain multiple security vulnerabilities [3], [12]. While authentication experts proposed a wide range of secure alternatives [16][17]–[20], Bonneau et al. [4] showed that the majority of these schemes offer more security than passwords, but they are difficult to use and / or expensive to deploy [4].

In [13] we have proposed a new authentication architecture, called StrongAuth, to enhance security without sacrificing usability and deployability. Specifically, our proposal does not require any additional equipment except a modern web browser. Later, we presented an improved version of StrongAuth including a secure session management mechanism [21]. This version covers the complete cycle of authentication in the context of web applications, consisting of mutual authentication phases and the subsequent HTTP requests authentication [21].

In this paper, we realize a prototype consisting of two modules: A registration module that implements the registration phase of the proposed scheme [21], and another authentication module that integrates both the mutual authentication and the session management phases. On the one hand, we integrate the client architecture component as an extension of Mozilla Firefox that can easily install it using Mozilla Firefox Add-ons Manager and the server component as a service of PHP applications within the WAMP platform, which allows us to avoid recompiling the source code of Mozilla Firefox and have an independent server component of the application code; which facilitates the deployment. On the other hand, we evaluate the performance of registration and authentication modules to evaluate the theoretical study presented in [21].

The rest of the paper is organized as follows: in section 2 we briefly review the registration, the authentication and session management phases of the proposed scheme [21]. In Section 3 we are particularly interested in the implementation of our prototype to show the feasibility of the proposed scheme. Section 4 presents our results and discussions of the experimental evaluation of the proposed prototype. Section 5 concluded the paper.

In the rest of this paper, we denoted by:

U_i	ith User.
ID_i	Unique identifier of user U_i.
P_i	Password of user U_i.
$Salt_i$	Cryptographic Salt of user U_i .
d	Web application domain name.
RW_i	Random value used at most once within the scope of a given session generated by the web application for U_i.
RB_i	Random value used at most once within the scope of a given session generated by the browser for U_i .
USK_i,UPK_i	Asymmetric key pair for user U_i generated by

attacks which require the knowledge of the public part of the RSA key.

- *AES-128 as symmetric encryption algorithm*: Today, AES specified in FIPS 197 [38] is the standard used in most security protocols (TLS, IPsec, etc.). The ANSSI and NIST recommend at least a 100-bit key for data to be protected until 2020, but ANSSI indicates in their report [38] that the use of a 128-bit key is preferable.
- *SHA-256 as a hash function*: Following multiple attacks against the SHA1 algorithm, the majority of applications decided to move to SHA-256 [39].
- *HMAC_SHA1 and HMAC_SHA256 as HMAC function*: While hash functions such as MD5 and SHA-1 are not longer considered safe due to reported collision attacks [40, p. 1], [41]. They may be used in the HMAC functions. HMAC does not require a collision-resistant hash function for its formal security proof [42], [43]. The use of more robust functions like SHA-2 [39] give more security guaranteed, but at a price in the performance level.
- *PBKDF 2 with 1000 iterations as key derivation function*: The use of a key derivation function that requires N iterations to get key increases the calculation cost to perform a dictionary attack on a password with t bits entropy form 2^t operations to 2^t * N operations. Therefore, it makes dictionary attacks and brute force more difficult. However, the computation required for the key derivation by legitimate users also increases with the number of iterations. Thus, there is an obvious compromise: A large number of iterations makes attacks more expensive, but affects performance for the authorized user. PBKDF Version 2 is defined in RFC 2898 [44]. NIST recommends a minimum of 1000 iterations [45].

B. Results and Discussion of the Registration Module

Table IV summarizes the execution time of our registration module, compared with the traditional registration (based on a username and password on HTTPS). The time reported is the average of 10 timings. The total run time includes the time of round trips and networks latency. We calculated the time needed to perform cryptographic operations both client side (Mozilla Firefox registration extension) and PHP applications side. This allowed us to assess the impact of these calculations on performance. As we can see in Table 4, the average total time performance of our proposal is 544.347 ms (with a standard deviation of 113.16 ms). This time is calculated starting the moment the user clicks the Sign up button (Figure 8), after entering their password to the success of this phase (Figure 15). It is clear that our registration module requires more time compared to the traditional registration of users (172.680 ± 32.085 ms). Of course, this is due to the operations integrated into our solution to enhance the security of this phase. Specifically, client-side cryptographic operations require 303.303 ms (± 80.607).

TABLE IV. AVERAGE EXECUTION TIME IN MS (± STANDARD DEVIATION) OF OUR REGISTRATION MODULE, COMPARED WITH THE TRADITIONAL USER'S REGISTRATION

Operation	Our Registration Module	Traditional Registration (Passwords over HTTPS)
Client cryptographic computations	303,303 ± 80,607	-
PHP application cryptographic computations	0,185± 0,018	0,127 ± 0,012
Total runtime	544,347 ± 113,16	172,680 ± 32,085

Figure 22 shows that over 96% of the client time is spent for the RSA key pair generation. Accordingly, the more we increase RSA keys length and the numbers of iterations of PBKDF, performances are affected. At the application side, we obtained almost similar performance to traditional registration. The origin of this small difference is the addition of a generation of a random value and two concatenation operations in our proposal. These are used to avoid storing the unencrypted public key of the user in the application database.

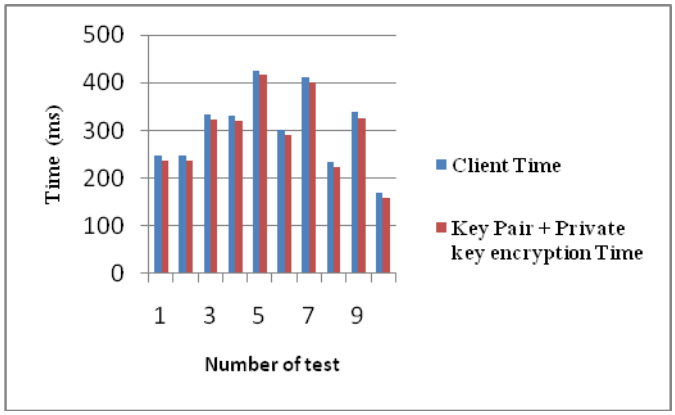

Fig. 22. Comparison of the generation time of 1024 bits RSA keys pair and the time required for other client-side operations

C. Results and Discussion of the Authentication Module

Table V summarizes the results of run time of our authentication module in comparison with traditional authentication (HTML form + HTTPS) and SSL / TLS client certificate authentication. From this table, we can clearly notice that compared to other mechanisms, our authentication module has the lowest total run time (about 148.415 ms 20.315 ms ±) and client certificate authentication SSL / TLS has the highest time (2923.5 ± 350.589). These results confirm those obtained during the theoretical performance analysis in [21].

Indeed, as we have already explained, the authentication phase of the proposed architecture does not require HTTPS, but relies on cryptographic parameters to enhance user authentication over an HTTP connection such as symmetric and asymmetric cryptography. The calculation of these parameters on the client side of our prototype takes 43.862 ms (2,376 ms ± standard deviation) and 15.268 ms (5,701 ms with standard deviation) on the web application side. While the proposed solution has an impact on performance that we think

is acceptable the performance compared to password authentication that requires only 0.065 ms in the application-side and no client-side computing; but security always has a cost, and we believe that the price of insecurity is much higher. Also, these calculations are not likely to affect the user experience.

TABLE V. AVERAGE EXECUTION TIME IN MS (± STANDARD DEVIATION) OF OUR AUTHENTICATION MODULE COMPARED WITH AUTHENTICATION BASED ON AN HTML FORM AND A PASSWORD OVER HTTPS AND WITH THE CLIENT SSL / TLS CERTIFICATE AUTHENTICATION.

Operation	Our Authentication Module	SSL/TLS client certificate authentication
Client cryptographic computations	43,862 ± 2,376	-
PHP application cryptographic computations	15,268 ± 5,701	-
Total runtime	148,415 ± 20,315	2923,5±350,589

Also, as discussed in [21], to create a secure session management mechanism, the proposed scheme attaches an HMAC signature in each HTTP request from the browser to the web application. This ensures the integrity and authenticity of HTTP requests. To assess the impact of this mechanism, we measured the time required to generate an HTTP request signed and time to validate it by the application. Table 6 presents the average time and standard deviations of our prototype in ms when using an HMAC_SHA1 and HMAC_SHA256 functions, compared with traditional session management (the use of cookies sent over HTTPS).

Again, the results in Table VI reaffirm those in [21]. In particular, it is clear to see that the computation time added by generating and validating a HMAC_SHA1 or HMAC_SHA2 is negligible compared to the total execution time (page load time). In other words, the user experience is not affected. Also, despite the cookies by session management requires no cryptographic operations on both the client side and the application side, but the use of HTTPS increases the total execution time required to load a requested page in a user (about 175.680 ms).

TABLE VI. COMPARING THE RESULTS OF BOTH HMAC_SHA1 AND HMAC_SHA256 FUNCTIONS APPLIED DURING THE SESSION MANAGEMENT

	Our authentication Module: Session management phase		HTTP cookies authentication over HTTPS
Operation	**HMAC_SHA1**	**HMAC_SHA256**	
Client cryptographic computations	1,846±0,246	1,974±0,24	-
PHP application cryptographic computations	0,0744± 0,036	0,098±0,0283	-
Total runtime	61,64±19,832	65,3593 ± 33,04	175,680 ± 42,124

V. CONCLUSION

In this paper, we demonstrated the implementation feasibility and experimental evaluation of a secure and efficient authentication scheme. We first presented the details of our proposed prototype implementation. Specifically, we separated the prototype in two modules to simplify the implementation process: A registration module that implements the registration phase and an authentication module which incorporates both mutual authentication and session management phases. In each module, the client component of the proposed prototype is developed as an extension of Mozilla Firefox browser that can easily install and the server component as a service of a PHP web application. This allowed us to avoid recompiling the source code of Mozilla Firefox and have an independent server component of the application code; which also facilitated the deployment procedure. After that, we focused on the experimental evaluation of the proposed prototype. Our experimental results confirmed the proposed scheme-theoretical analysis. In fact, even if the registration phases and mutual authentication of our prototype require expensive cryptographic operations (especially asymmetric cryptography) increasing the computing time, the session management phase will need only a negligible overhead. Compared to the related scheme, we showed that the proposed scheme not only improves the usability and deployability, but also improves the user authentication performances.

REFERENCES

[1] K. Fu, E. Sit, K. Smith, and N. Feamster, “The Dos and Don’ts of Client Authentication on the Web.,” in *USENIX Security Symposium*, 2001, pp. 251–268.

[2] M. Weir, S. Aggarwal, M. Collins, and H. Stern, “Testing metrics for password creation policies by attacking large sets of revealed passwords,” 2010, pp. 162–175.

[3] J. Bonneau and S. Preibusch, “The Password Thicket: Technical and Market Failures in Human Authentication on the Web.,” presented at the The Ninth Workshop on the Economics of Information Security, 2010.

[4] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano, “The quest to replace passwords: A framework for comparative evaluation of web authentication schemes,” presented at the 2012 IEEE Symposium on security and privacy, San Francisco, 2012, pp. 553–567.

[5] J. Bonneau, “The science of guessing: analyzing an anonymized corpus of 70 million passwords,” presented at the 2012 IEEE Symposium on Security and Privacy, San Francisco, 2012, pp. 538–552.

[6] B. Grawemeyer and H. Johnson, “Using and managing multiple passwords: A week to a view,” *Interact. Comput.*, vol. 23, no. 3, pp. 256–267, May 2011.

[7] D. Sandler and D. S. Wallach, “<input type=‘password’> must die,” presented at the Web 2.0 Security & Privacy, 2008.

[8] D. Florencio and C. Herley, “A large-scale study of web password habits,” presented at the the 16th international conference on World Wide Web, New York, 2007, pp. 657–666.

[9] E. Grosse and M. Upadhyay, “Authentication at Scale,” *Secur. Priv. IEEE*, vol. 11, no. 1, pp. 15 – 22, 2013.

[10] D. Florêncio, C. Herley, and B. Coskun, “Do strong web passwords accomplish anything,” presented at the 2nd USENIX Workshop on Hot Topics in Security, Boston, 2007.

[11] S. Grzonkowski, W. Zaremba, M. Zaremba, and B. McDaniel, “Extending web applications with a lightweight zero knowledge proof authentication,” New York, 2008, pp. 65–70.

[12] D. Stuttard and M. Pinto, *The web application hacker’s handbook finding and exploiting security flaws*. Indianapolis: Wiley, 2011.

[13] Y. Sadqi, A. Asimi, and Y. Asimi, “A Cryptographic Mutual Authentication Scheme for Web Applications,” *Int. J. Netw. Secur. Its Appl.*, vol. 6, pp. 1–15, 2014.

[14] M. Manulis, D. Stebila, and N. Denham, "Secure Modular Password Authentication for the Web Using Channel Bindings," in *Security Standardisation Research*, vol. 8893, L. Chen and C. Mitchell, Eds. Springer International Publishing, 2014, pp. 167–189.

[15] Y. Sadqi, A. Asimi, and Y. Asimi, "A Lightweight and Secure Session Management Protocol," *Lect. Notes Comput. Sci.*, vol. 8593, pp. 319–323, 2014.

[16] IETF, "RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2," 2008. [Online]. Available: http://tools.ietf.org/html/rfc5246.

[17] M. Wu, S. Garfinkel, and R. Miller, "Secure web authentication with mobile phones," presented at the DIMACS workshop on usable privacy and security software, 2004, pp. 9–10.

[18] B. Parno, C. Kuo, and A. Perrig, "Phoolproof phishing prevention," presented at the Financial Cryptography and Data Security (FC'06), Anguilla, British West Indies, 2006.

[19] Google, "About 2-step verification - Accounts Help." [Online]. Available: https://support.google.com/accounts/answer/180744?hl=en&ref_topic=1099588.

[20] Mozilla, "Mozilla Persona — simple sign-in with email — mozilla.org." [Online]. Available: http://www.mozilla.org/en-US/persona/.

[21] Y. Sadqi, A. Asimi, and Y. Asimi, "A Secure and Efficient User Authentication Scheme for the Web," *Int. J. Internet Technol. Secur. Trans.*, vol. 5, no. 1, pp. 43–63, 2015.

[22] Mozilla, "js-ctypes - Mozilla | MDN." [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/js-ctypes.

[23] Mozilla, "NSS - Mozilla | MDN." [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Projects/NSS.

[24] Mozilla, "Storage | MDN." [Online]. Available: https://developer.mozilla.org/en-US/docs/Storage.

[25] SQLite, "SQLite." [Online]. Available: https://www.sqlite.org/.

[26] Apache Foundation, "Apache HTTP Server Project." [Online]. Available: http://httpd.apache.org/.

[27] Apache Foundation, "mod_ssl - Apache HTTP Server Version 2.2." [Online]. Available: http://httpd.apache.org/docs/2.2/mod/mod_ssl.html.

[28] PHP, "Hash: Hash - Manual." [Online]. Available: http://php.net/manual/en/book.hash.php.

[29] PHP, "OpenSSL: PHP OpenSSL - Manual." [Online]. Available: http://php.net/manual/en/book.openssl.php.

[30] Oracle, "MySQL :The world's most popular open source database." [Online]. Available: https://www.mysql.com/.

[31] Y. Asimi, A. Amghar, A. Asimi, and Y. Sadqi, "New Random Generator of a Safe Cryptographic Salt per session (RGSCS)," *Int. J. Netw. Secur.*, vol. 18, no. 3, pp. 445–453, 2016.

[32] M. Bellare, R. Canetti, and H. Krawczyk, "Message authentication using hash functions: The HMAC construction," *RSA Lab. CryptoBytes*, vol. 2, no. 1, pp. 12–15, 1996.

[33] RFC 2104, "HMAC: Keyed-Hashing for Message Authentication," 1997. [Online]. Available: https://www.ietf.org/rfc/rfc2104.txt.

[34] IETF, "RFC 6265 - HTTP State Management Mechanism," 2011. [Online]. Available: http://tools.ietf.org/html/rfc6265. [Accessed: 08-May-2015].

[35] Mozilla, "nsIHttpChannel - Mozilla | MDN." [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM/Reference/Interface/nsIHttpChannel. [Accessed: 08-May-2015].

[36] ANSSI, "Référentiel Général de Sécurité: Règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques," 2014.

[37] E. Barker and A. Roginsky, "Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths," *NIST Spec. Publ.*, vol. 800, p. 131A, 2011.

[38] NIST, "FIPS 197: ADVANCED ENCRYPTION STANDARD (AES)," *Process. Stand. Publ.*, Nov. 2001.

[39] NIST, "FIPS 180-4: Secure Hash Standard (SHS)," *Process. Stand. Publ.*, 2012.

[40] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *Advances in Cryptology–CRYPTO 2005*, 2005, pp. 17–36.

[41] X. Wang and H. Yu, "How to break MD5 and other hash functions," in *Advances in Cryptology–EUROCRYPT 2005*, Springer, 2005, pp. 19–35.

[42] S. Turner and L. Chen, "RFC 6151 - Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms," 2011. [Online]. Available: https://tools.ietf.org/html/rfc6151.

[43] M. Bellare, "New proofs for NMAC and HMAC: Security without collision-resistance," in *Advances in Cryptology-CRYPTO 2006*, Springer, 2006, pp. 602–619.

[44] RSA Laboratories, "PKCS #5 v2.1: Password-Based Cryptography Standard," 2000. [Online]. Available: https://www.ietf.org/rfc/rfc2898.txt.

[45] M. S. Turan, E. Barker, W. Burr, and L. Chen, "Recommendation for password-based key derivation," *NIST Spec. Publ.*, vol. 800, p. 132, 2010.