An alternative version of HTTPS to provide non-repudiation security property

A flexible component-based approach for secured transactions in a mobile environment

Stassia Resondry, Karima Boudaoud, Michel Kamel, Yoann Bertrand and Michel Riveill
Univ. Nice Sophia Antipolis, CNRS, I3S, UMR 7271
06900 Sophia Antipolis, France
Email: {zaffimih,karima,kamel,bertrand,riveill}@polytech.unice.fr

***Abstract*— The number of mobile devices connected to the Internet is rapidly growing, inducing security issues that cannot be prevented by common mechanisms such as HTTPS. Indeed, mobile environments require light algorithms that can reduce the power-consumption and extend battery life. Moreover, HTTPS does not offer fine-grained control over the security properties such as integrity, confidentiality or authenticity. This lack of flexibility can be problematic for both power-consumption and security robustness. To overcome these issues, we have proposed in previous works a modular architecture, called LECCSAM, based on security components to secure any communication protocol by adding the required security properties. In the context of HTTP, it provides an alternative version of HTTPS by adding the integrity, confidentiality, and authenticity properties to HTTP separately or in block (i.e. only one property or any combinations of two or more properties), depending on the user needs and usage context. In this paper, we propose to extend this alternative version of HTTPS with the non-repudiation property. Preliminary results of the performance evaluation are encouraging.**

Keywords—Security properties; non-repudiation; security components; HTTPS; communication protocols.

I. INTRODUCTION

The rapid growth of mobile market has forced IT actors to migrate well-known protocols to the smartphone environment. These protocols need to take into account several parameters such as power-consumption and resource-hungriness in order to fit smartphone's prerequisites. In a traditional computer environment, the most common protocols used to access the Web is HTTP and its secured version HTTPS. HTTPS offers great robustness, but its high power-consumption [1] and lack of flexibility [2] do not make it suitable for mobile usage. Indeed, HTTPS provides, in block, only integrity, authenticity and confidentiality and does not offer the possibility to choose which properties to use depending on the available resources (i.e. the computational power of the device), the remaining battery or the data's sensitivity. As a result, HTTPS cannot be proposed as a suitable solution for secured mobile communications.

In [2], we have proposed a solution called LECCSAM (Low Energy Consuming and Component based Security Architecture for Mobiles) that allows end-users to specify the security properties that they would like to see applied on their data when using HTTPS. Actually, thanks to LECCSAM a security property-based HTTPS has been proposed to provide a flexible HTTPS that adapts to users' needs. The main goal of LECCSAM is to secure users' data, exchanged between a smartphone and a server or another smartphone, by applying five security properties: integrity, confidentiality, authenticity, non-repudiation and access control thanks to eponymous security components.

A detailed description of LECCSAM architecture and the main components (including the following security properties: integrity, confidentiality and authenticity) has been presented in [2]. In this paper, we focus on the fourth security component i.e. non-repudiation component and more specifically on its design, implementation and use.

This paper is structured as follows. In section II and section III, we briefly define the non-repudiation property and present the existing non-repudiation protocols. In section IV, we give an overview on LECCSAM. In sections V and VI, we focus on the description of the design and implementation of the non-repudiation component. In section VII, we show some results of the performance evaluation. Finally, in section VIII, we conclude this paper and propose future works.

II. NON-REPUDIATION PROPERTY

Non-repudiation guarantees that a party cannot deny having received/sent the message. Even, if it is not the most used security property, it can come in handy for scenarios involving trust during sensitive exchanges. Different types of non-repudiation have been proposed, depending on who (sender or recipient) is applying the non-repudiation mechanism. From the sender point of view, one would be willing to be sure that her/his message was received by the recipient (*non-repudiation of receipt* (NRR)) or her/his message was well sent to the recipient (*non-repudiation of submission* (NRS)) or her/his message has been delivered to the recipient (*non-repudiation of delivery* (NRD)). From the recipient point of view, one would be willing to be sure that the message she/he received has been sent by a genuine sender (*non-repudiation of origin* (NRO)).

NRR is quite simple to implement because it only needs a nuncio (i.e. a particular document that attests the validity of the transaction). The nuncio is generated by the sender and transmitted to the recipient. In order to terminate the transaction, the recipient needs to send the nuncio back.

978-1-4799-0959-9/14/$31.00 ©2014 IEEE

The NRO is easy to implement too. This mechanism requires most of the time a digital signature in order to prove to the recipient that the sender is genuine. The digital signature can be done with traditional algorithms such as SHA [3] or MD5 [4].

Things tend to be a little bit more complex for NRS as this kind of non-repudiation requires a Trusted Third Party (TTP) to allow a secure and trusted interaction between a recipient and a sender. A TTP must be trusted by both parties and act as a man-in-the-middle to share and store proofs like digital signatures or nuncios. We stated previously that a non-repudiation of receipt (NRR) needs only a nuncio, but a TTP can also be used to ensure that the nuncio is not altered during the transmission.

III. RELATED WORKS

In this section, we present existing protocols implementing non-repudiation property.

Even if non-repudiation is not the most commonly used security property, various works [5] have been conducted to provide efficient and reliable protocols that implement such a property. These non-repudiation (NR) protocols can be divided in five categories:

- NR without TTP ([6] [7]).
- NR with *inline* TTP, in which the TTP is involved in each message ([8], [9]).
- NR with *online* TTP, in which the TTP is involved in each session (i.e. for several messages) ([10], [11]).
- NR with *offline* TTP, in which the TTP is involved only if an incorrect behaviour is detected ([12], [13]).
- NR with *transparent* TTP, in which the offline TTP produces evidence indistinguishable from the evidences of both sender and recipient ([14] [15]).

As stated previously, non-repudiation without TTP can be easily performed between the sender and the recipient. This solution can have a lack of robustness if the sender or the recipient is not trustworthy. On the contrary, protocols with inline TTP offer robustness, but they come with a time-consuming drawback due to three-way transmissions (sender → TTP → recipient). Nevertheless, solutions with online TTP try to prevent this lack of performances by involving the third party only for session management, which is useful if more than one message is sent during a session. Finally, offline and transparent TTPs offer an optimistic approach that involves the third party only when a problem occurs. These mechanisms are efficient if most of the transactions are genuine and trustworthy. Despite the number of proposed solutions, none of them is using HTTP as a communication protocol. Therefore, the aim of our work is to add the non-repudiation property to an alternative version of HTTPs provided thanks to the LECCSAM architecture, that is described below.

IV. OVERVIEW OF LECCSAM

LECCSAM is an architecture that allows securing any communication protocol by adding the required security properties. A proof of concept has been provided to secure HTPP and XMPP by designing an alternative version of HTTPS and XMPPS. LECCSAM allows an end-user through her/his mobile device to choose the security properties that she/he wants to see applied on her/his data when sent to another user or server. This architecture is composed of several modules that have been described in details in [2]. However, here is a summary of the main components (see "Fig. 1"):

- The ***security components***, where each component is an assembly of existing cryptographic tools that fulfils the eponymous security property. In LECCSAM, five security components have been defined for each security property (integrity, confidentiality, authenticity, non-repudiation and access control).
- The ***policy engine*** specifies the security properties required to ensure the security level requested by the end-user for her/his data exchange.
- The ***manager*** is the main component of LECCSAM. Its role is to orchestrate the security components to secure the data exchange. According to the security properties specified by the policy engine, the manager loads the right security components and apply the eponymous security properties to the data.
- The ***interceptor*** is a module based on port listeners that intercepts all HTTP requests, extracts data and sends it to the manager in order to be secured. These listeners are located on both mobile device and LECCSAM side.

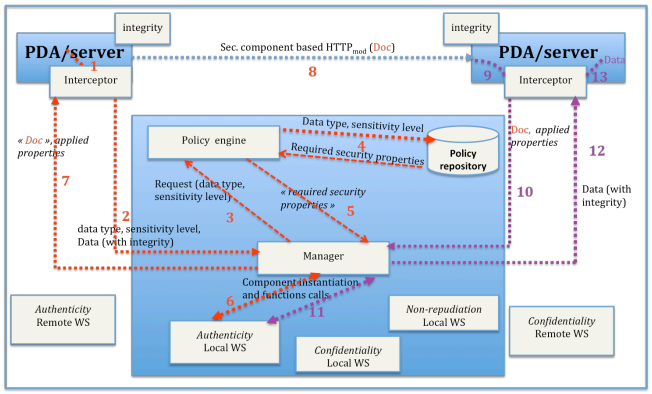

Fig. 1. LECCSAM architecture and workflow

In the next two sections, we focus on the description of the design and implentation of the non-repudiation component.

V. NON-REPUDIATION COMPONENT

In this section, we present the non-repudiation component that will be used in order to add the non-repudiation property to the alternative version of HTTPS.

A. Design choices

In order to design the non-repudiation component, we had to choose the suitable type of non-repudiation. As stated in section II, existing solutions offer both advantages and drawbacks. In our case, we have chosen the non-repudiation of delivery (NRD) as, from our point of view, it is the most appropriate version of non-repudiation. This type of non-repudiation allows a sender to be sure that her/his message was sent. Because NRD needs a third party, we have chosen LECCSAM as a TTP. Concerning the type of TTP, we have decided to use LECCSAM as an inline TTP. Consequently, it will be involved in every message transmission.

To be considered as trusted, LECCSAM needs to embed several things: symmetric/asymmetric key mechanisms, a logging mechanism and of course a non-repudiation component. The keys are used to ensure that the nuncio and the data are not altered while transmitted between the different entities. The logging mechanism is used to keep a track of all transmissions with non-repudiation policy. Finally, the non-repudiation component embeds the necessary mechanisms to ensure the non-repudiation property.

B. *Non-repudiation workflow*

The non-repudiation mechanism is a succession of several steps described in "Fig. 2". These steps are summarized below:

- First, the sender (i.e. mobile device) generates a request R that contains the data and the security level (i.e. data sensitivity) that the user wants to apply to her/his data.
- Then, the sender generates a nuncio and uses the symmetric key k1 that it shares with LECCSAM to encrypt the concatenation of the request and the nuncio $\{R \parallel N\}_{k1}$. The concatenated message is then sent to LECCSAM.
- LECCSAM receives the message and the manager analyses the security sensitivity and ask the policy engine to determine which security properties (SP) to apply. The manager applies then the identified security properties to the data to be sent.
- After that, LECCSAM sends back the modified data $\{R\}_{ps}$ to the sender.
- The sender analyses the received data. If it contains the non-repudiation property, it will wait for the nuncio's return for a certain time. In the meantime, it sends the modified data to the recipient (mobile device or server).
- When the recipient receives the data, it asks LECCSAM to verify the security properties. To do so, the recipient sends the modified data $\{R\}_{ps}$.
- LECCSAM receives the data, analyses the security properties and check the security properties applied previously (exp. check the integrity if this property has been applied) or decrypt the data if the confidentiality property has been applied.
- Then, LECCSAM encrypts the data with the symmetric key k2 that it shares with the recipient. After this operation, it sends the data $\{R\}_{k2}$ back to the recipient.

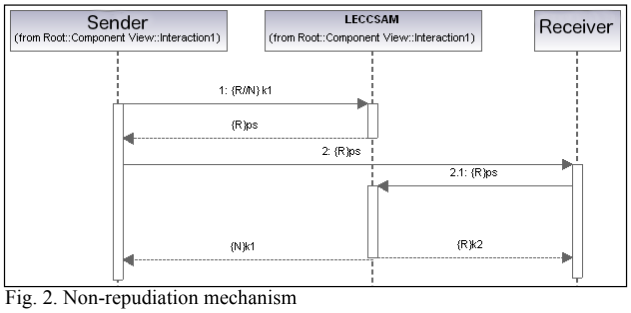

Fig. 2. Non-repudiation mechanism

Finally, LECCSAM encrypts the sender's nuncio $\{N\}_{k1}$ and sends it back to the sender. The sender decrypts then the nuncio with k1 in order to be sure that the data was properly delivered.

C. *Non-repudiation component*

This component embeds different types of cryptographic mechanisms. As shown in "Fig. 3", the data D to send is encrypted with a symmetric key K. This key is created by a random secret key generator. Then, the data $\{D\}_K$ is produced.

The second embedded cryptographic mechanism is an asymmetric mechanism. LECCSAM public key PK is used to encrypt the concatenation of the symmetric key K and the nuncio (K || N). Once this is over, the data $\{K \parallel N\}_{PK}$ is produced. Finally, both generated data $\{D\}_k$ and $\{K \parallel N\}_{PK}$ are concatenated and sent.

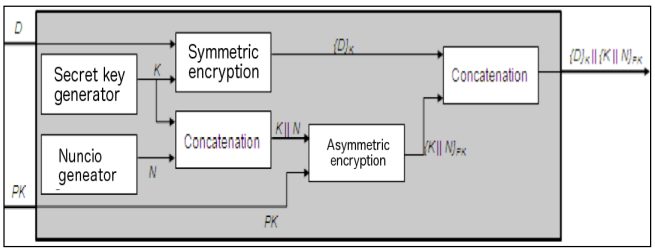

Fig. 3. Non-repudiation component

D. *Keys Exchange workflow*

Data exchanged between LECCSAM and mobile device (or server) are encrypted with AES standard [16]. In the first version of LECCSAM, we supposed that each entity (i.e. mobile/server and LECCSAM) had already secret keys. In the new version of LECCSAM, we decided to use Diffie-Hellman keys exchange [17] to generate secret keys for each entity. The key exchange between the mobile/server and LECCSAM is initiated before each communication session as a hacker can steal the secret key used for a previous communication. The following steps and "Fig. 4" describe this key exchange mechanism:

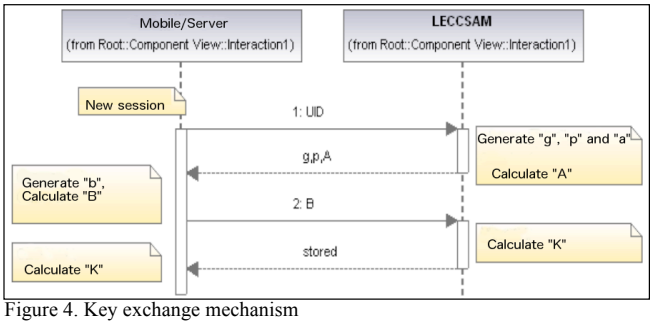

Figure 4. Key exchange mechanism

- The mobile device initiates a new session with LECCSAM by giving its UID, which is an SHA-1 value.
- LECCSAM generates 3 random parameters 'g', 'p' and 'a'. It sends to LECCSAM 'g', 'p' and 'A' such as $A = g^a \bmod_p$
- The mobile device chooses a random number 'b' after reception of 'g', 'p' and 'A'.
- The mobile device sends to LECCSAM 'B' such as $B = g^b \bmod_p$
- LECCSAM receives 'B' and generates 'K' such as $K = A^b \bmod_p$
- LECCSAM saves K and the UID and send K to the mobile device.

E. Key exchange component

We have created a component implemented locally (i.e. on both client and LECCSAM side) in order to perform the previous workflow. This mechanism is in charge of the transmission of 'g' and 'p' to each entity. As stated previously, these values are used by the mobile/server to generate the same secret key between LECCSAM and the mobile device/server. This key is used in order to encrypt communications between these two entities. "Fig. 5" describes the key exchange component.

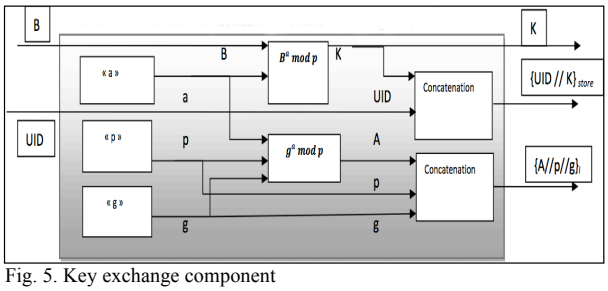

Fig. 5. Key exchange component

F. Logging module

This module is also implemented in both LECCSAM and mobile device/server side. It is used to reinforce non-repudiation component by keeping a track of every sent or received requests while using the non-repudiation security property. This information is stored in a specific format that includes time and date, UID and nuncios. In case of conflict, one can uses this module to prove or revoke a transaction.

VI. IMPLEMENTATION

In this section, we focus on the implementation of the non-repudiation security component.

A. Non-repudiation component

To implement the non-repudiation component, we have defined four classes:

- ***SunJCEAssymetricCiphering***, which is used to ensure asymmetric ciphering. The keys are created within LECCSAM side.
- ***SCNonRepudiation***, which ensures the non-repudiation property.
- ***Base64***, which is used to match the differences between Java J2SE and Android.
- ***LECCSAM***, which represents LECCSAM as a TTP.

In order to schedule the non-repudiation operations of "Fig. 3", we have defined a new component called ***non-repudiation manager.*** This manager is embedded in LECCSAM side and its role is to:

- Call the traditional manager to apply the non-repudiation property on the data.
- Retrieve and transmit the nuncio sent by the mobile device.
- Log the requests and the corresponding nuncios in order to keep a trace of the transactions.

B. Key exchange component

We have implemented a key exchange component (***KeyManagementEntity***) for the mobile device and LECCSAM. Thanks to this component, LECCSAM is able to distribute 'g' and 'p' to each mobile device. These values are used to generate the same secret key between a mobile device and LECCSAM in order to encrypt the communications between the two entities. To implement this component, we have defined five classes:

- ***IkeyManager***, which is the entry interface used to call the key Manager.
- ***KeyManager***, which ensures the key management mechanism (exchange and storage of the keys).
- ***KeyOp***, which realises operations such as saving, creating or deleting a key.
- ***DHLECCSAM***, which generates 'g', 'p' and 'd' parameters necessary for Diffie-Hellman keys exchange and generates 'B' parameters.
- ***DHCLIENT***, which generates the secret key depending on the values sent by LECCSAM.

C. Security components orchestration

Usually, the non-repudiation property is applied in addition to other security properties. To apply non-repudiation, the policy engine selects one of the following security properties combinations (see TABLE I).

TABLE I. SECURITY PROPERTIES COMBINATIONS

Security Properties	**Integrity Non-repudiation**	**Integrity Confidentiality Non-repudiation**	**Integrity Confidentiality Authenticity Non-repudiation**
Component order when Sending data	SCNonRepudiation SCIntegrity	SCConfidentiality SCNonRepudiation SCIntegrity	SCCConfidentiality SCNonRepudiation SCAuthenticity
Component order when Receiving data	SCIntegrity SCNonRepudiation	SCIntegrity SCNonRepudiation SCConfidentiality	SCAuthenticity SCNonRepudiation SCConfidentiality

D. HTTP transmission workflow

In this section, we explain the workflow between a mobile device and LECCSAM when the non-repudiation property is requested in addition to integrity and confidentiality properties.

Before sending a message, the user A's interceptor adds the types "***datatype***" and "***sensitivity***" to the HTTP request. The datatype depends on the context (student tests, medical records, etc.). The sensitivity is specified by the user thanks to a graphical interface. Once the request if forged, the interceptor sends it to LECCSAM:

```
GET /exam.txt/?profID=
c678832bf6f37e1c4e8c265b77e920bbe80cec97c94c64796ed6400504b08aeb9d3c
42b65c7296370ab2 HTTP /1.1
datasensitivity:3
connection:Keep-Alive
accept-encoding:gzip, deflate
datatype:exam
.....
c379d505c51280c623e738f2ff657b041cb8ea8671d46db2620731fl1qn151b8b7pf
Cf03a90f89e116m
```

After reception by LECCSAM's interceptor, the manager extracts the data. It verifies its integrity and change the

parameter "***scintegrity:add***" to "***scintegrity:successfull***" if the integrity is preserved. Then, it decrypts the data with the secret key exchanged at the beginning of the session.

After this operation, the manager has the decrypted information and the nuncio.

GET /exam.txt/?profID= **LILI** HTTP/1.1
datasensitivity:3
datatype:exam
scintegrity:: successful
….
My Message

In order to determine the right security properties to use, the manager sends "***datasensitivity***" and "***datatype***" to the policy engine. In this example, the policy engine sends to the manager the rule "***int_conf_NR***" to indicate that it has to apply the integrity, confidentiality and non-repudiation. However, as the integrity has already been applied by the mobile device, the manager will only apply the confidentiality and non-repudiation properties. After applying both properties, the manager sends the data to the interceptor of the mobile device, which in its turn sends the data to the mobile device of the user B.

GET / exam.txt /?profID= **DN2hhOSjtSS4D**.... HTTP/1.1
scintegrity:successful
scproperty : int_conf_NR
…
Te206VMDJ6kCFT6brRX8SYXu5okRr6eI4lOOrhoQ9honbL6w...

After receiving the data, the interceptor of the mobile device of the user B retrieves the security properties applied thanks to the field ***scproperty***. If the security property used is the integrity, the interceptor checks the integrity of the message thanks to the embedded integrity component. If other security properties are specified in the field ***scproperty***, the interceptor sends the data to LECCSAM in order to check the other security properties and/or decrypt the data.

According to "***int_conf_NR***", LECCSAM checks the security properties and retrieves the plaintext message. Once this is done, it encrypts the plaintext with the secret key shared with the mobile device of the user B and sends the data to this latter.

GET /exam.txt/?profID= **fb613078e8430eabe53e8**... HTTP/1.1
scintegrity:add
scproperty : int_conf_NR
.....
7014e4dfce0e0d31f19d28086c6cbe65cdb2V644ff/st89s5126lw...

The mobile device of the user B receives the data and uses its private key to decrypt the message in order to retrieve the original message.

GET /exam.txt/?profID= **LILI** HTTP/1.1
...
My Message

In the meantime, LECCSAM sends the nuncio to the mobile device of the user A in order to prove that the original message was delivered to the mobile device of the user B.

VII. TESTS AND RESULTS

In this section, we present the performance evaluations of LECCSAM and of the implementation of the alternative version of HTTPS, in terms of processing time and power-consumption,. We first give an overview about the test environment and the scenarios we used to conduct the tests. The second and third sub-sections present the results we obtained while performing our tests.

A. Test environment and scenarios

We used Battery Snap and Traceview [18] to determine both resources and energy-consumption while sending/receiving data. Concerning the device, we used a 2011 smartphone on Android 2.1 to perform the tests. Our solution was tested with the following scenarios:

- HTTP with integrity and non-repudiation (1)
- HTTP with integrity, confidentiality, authenticity (2)
- HTTP with integrity, confidentiality, authenticity and non-repudiation (3)
- Traditional HTTPS (integrity, authenticity and confidentiality) (4)

B. Time-consumption comparisons

In (1), we measured the time-consumption of a secure transaction between the mobile device and LECCSAM, where integrity and non-repudiation properties have been applied. . The overall time of the transaction was ***7,58 s***. It is quite long, but it is important to state that AES encryption/decryption processing takes at least 3 seconds. In (2), we did the same with all security properties except non-repudiation. The time-consumption for this scenario (already used for the previous version of LECCSAM [2]) is ***4,5 s*** per transaction. In (3), we used our new non-repudiation component with the previous configuration. In this case, the time-consumption was ***8 s.***

Finally, we have evaluated the time-consumption of the traditional HTTPS that ensures integrity, confidentiality and authenticity (4). We obtained an overall time of ***13 s***. These results are explainable by the fact that 90% of the time was used to perform additional tasks such as Webview (a view that displays Web pages within an Android activity). The tasks were not related to HTTPS, but it was not possible to deactivate these tasks in order to enhance HTTPS performances. Fig. 6 presents the results we have obtained with the previous scenarios.

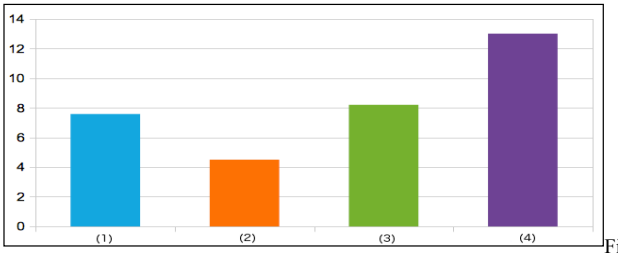

Fig. 6. Time-consumption results in seconds

As we can see, adding non-repudiation property increases the time of computation. This drawback is mostly due to the fact that non-repudiation needs additional operations (i.e. generation, encryption/decryption and transmission of the nuncio).

C. Energy-consumption comparisons

As stated previously, we used Battery Snap to perform our tests. Sadly, this tool shows only 10% of battery variation. In other words, it was not possible to determine exactly the cost of a single transaction. Thus, we have decided to calculate the necessary time to consume 10% of battery while using one of the previous scenarios. To do so, we have modified the program to send continuous transactions. For these tests, we have compared:

- HTTP with integrity, confidentiality, authenticity with key exchange mechanism (A)
- HTTP with integrity, confidentiality, authenticity without key exchange mechanism (B)
- HTTP with integrity, confidentiality, authenticity and non-repudiation (C)
- Traditional HTTPS (D)

While using settings (A), it took ***5,55 mns*** for the battery to drop from 100% to 90%. Moreover, it took ***4 mns*** to the traditional HTTPS (D) to use the same amount of energy. Concerning our new implementation, we can say that the scenario (C) is quite energy-consuming. Indeed, it took only ***3 mns*** for the mobile to loose 10% of battery. This energy-consumption is due to the fact that the communication session was continuously open while waiting for the nuncio's return. Thus, a comparison between tests (A) and (B) showed that the key exchange mechanism alone is not very energy-consuming. Fig. 7 presents our results.

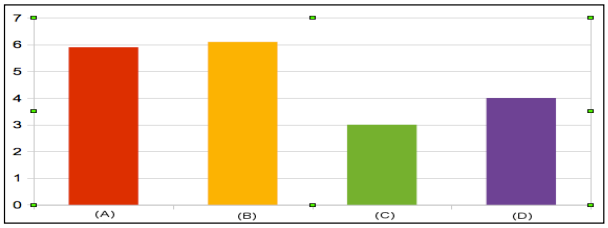

Fig. 7. Energy-consumption results in minutes

VIII. CONCLUSION

In this paper, we have presented the design and implementation of a non-repudiation security component that provides the eponymous security property. Thanks to this component, our architecture LECCSAM can provide an alternative version of HTTPS with a non-repudiation property, in addition to the integrity, confidentiality and authenticity properties.

Regarding the performance evaluations, our tests have shown that non-repudiation is quite consuming in terms of processing time and energy consumption. This is due mainly to the nuncio exchange and the cyphering mechanisms. Nevertheless, our implementation has shown that it is possible to add a non-repudiation property to HTTP in order to ensure that a message is delivered, which is not the case with the traditional HTTPS.

IX. FUTURE WORKS

In future works, we will focus on the optimisation of our non-repudiation component by testing other kinds of non-repudiation mechanisms (non-repudiation of receipt, non-repudiation of submission, etc.) and Trusted Third Party (online, offline, etc.).

As the test results have shown that the waiting time for the nuncio's return was responsible of the high energy-consumption; it will be interesting to resolve this problem by adding a shorter session timer or a mechanism that will allow the session to pause while waiting for the nuncio.

REFERENCES

[1] V. Gupa, M. Wurm. The Energy Cost of SSL in Depply Embedded Systems. Technical report Sun Microsystems, TR-2008-173, June 2008.
[2] M. Kamel, K. Boudaoud, S. Resondry, M. Riveill. A Low-Energy Consuming and User-centric Security Management Architecture Adapted to Mobile Environments. In Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM'2011), Dublin, Ireland, May, 23 - 27, 2011
[3] D Eastlake, P Jones - 2001 - RFC 3174, September
[4] R. Rivest, "RFC 1321: The MD5 message-digest algorithm," Technical Report, Internet Activities Board, April 1992.
[5] S. Kremer, O. Markowitch, and J. Zhou. An intensive survey of non-repudiation protocols. Computer Communications, 25(17):1606–1621, Nov. 2002.
[6] O. Markowitch, Y. Roggeman, Probabilistic non-repudiation without trusted third party, in: Second Conference on Security in Communication Networks'99, Amal¯, Italy, 1999.
[7] J. Mitsianis, A new approach to enforcing non-repudiation of receipt, manuscript (2001).
[8] T. Coffey, P. Saidha, Non-repudiation with mandatory proof of receipt, ACMCCR: Computer Communication Review 26
[9] B. Cox, J. D. Tygar, M. Sirbu, NetBill security and transaction protocol, in: USENIX Association (Ed.), Proceedings of the first USENIX Workshop of Electronic Commerce, USENIX, 1995, pp. 77-88.
[10] M. O. Rabin, Transaction protection by beacons, Journal of Computer and System Sciences 27 (2) (1983) 256-267.
[11] N. Zhang, Q. Shi, Achieving non-repudiation of receipt, The Computer Journal 39 (10) (1996) 844-853
[12] N. Asokan, M. Schunter, M. Waidner, Optimistic protocols for fair exchange, in: T. Matsumoto (Ed.), 4th ACM Conference on Computer and Communications Security, ACM Press, Zurich, Switzerland, 1997, pp. 6, 8-17
[13] S. Micali, Certified E-mail with invisible post offices, Available from author; an invited presentation at the RSA '97 conference (1997).
[14] O. Markowitch, S. Kremer, An optimistic non-repudiation protocol with transparent trusted third party, in: Information Security Conference 2001, Lecture Notes in Computer Science, Springer-Verlag, 2001.
[15] F. Bao, R. H. Deng, W. Mao, Efficient and practical fair exchange protocols with off-line TTP, in: IEEE Symposium on Security and Privacy, 1998.
[16] Daemen, J., & Rijmen, V. (2002). *The design of Rijndael: AES-the advanced encryption standard*. Springer.
[17] W. Diffie, M. E. Hellman, "New Directions in Cryptography," in IEEE Transactions on Information Theory, pp. 644-654, 1976.
[18] Google play application, Battery Snap https://play.google.com/store/apps/details?id=com.xelacorp.android.batsnaps&hl=fr