International Journal of Engineering and Management Sciences (IJEMS) Vol. 2. (2017). No. 2.
DOI: 10.21791/IJEMS.2017.2.1.

Security analysis of a „Location-stamping" protocol for GPS coordinates

É. ÁDÁMKÓ

University of Debrecen, Faculty of Engineering, Department of Basic Technical Studies,
adamko.eva@eng.unideb.hu

Abstract. Due to the rapid growth of GNSS based techniques in everyday life a service which can provide certified location information given by GPS coordinates became a worth considering idea. We designed two protocols that can achieve this goal, these can provide authenticate location and time information for any device which has a GPS receiver. In this article, I would like to prove -with the help of ProVerif software tool-, the latter statement. I investigated the authenticity and data integrity properties of the protocol.

Introduction

Despite the fact that the range of services based on the GNSS system has been increased in the last decade, only a small part of the system's security issues got enough attention in the literature. These security issues, or vulnerabilities are mostly related to the stable operation of the GNSS system itself, such as guaranteeing uninterrupted service or providing more accurate geographic data. Besides the issues, very little is said about the cryptographic reliability of these services or the entire GNSS system. On the other hand, this approach is a very important part of the GNSS system's application because there exist several services which should provide cryptographically authentic data, so it would be essential to make location determination secure and reliable. To solve the above-mentioned problem and provide cryptographically authentic location information, two stamping protocols were made in 2013. Our protocol can provide an authenticate location and time information for any device which has a GPS receiver. In this article, I deal only with the higher safety level protocol from [1]. Interested reader can get some examples of these services and both protocols in [1].

During the construction of the communication protocols, I always try to eliminate all the known attacks as far as possible. In case of cryptographic protocols, it is even more important because it is not enough to pay attention of the transmission channel's proper working, but a protocol like this should provide that the messages sent through the channel are not altered by any third party. Obviously, the malicious falsification of the messages only one of the many challenges, what a cryptographically authenticated protocol should solve. To achieve an appropriate safety leveled protocol, through designing it needed to use cryptographic primitives what can provide the proper level of security. However, using the right component, it is not proved that the protocol has the same, high-level of security. To get real information about the working of the protocol, and finding the vulnerabilities

DOI: 10.21791/IJEMS.2017.2.1.

there are several formal analysis methods available. To formalize the proof, I used the Applied-π calculus and I implemented the automatic verification with the help of the ProVerif software tool.

1. Location stamping

The basic idea is that the data received and/or computed by the GPS device are sent to a trusted organization, which can be for example a certification authority. The GPS device digitally signs the message. If this information is compatible with other information available by the organization, then it signs the GPS coordinates. In detailed our aim is to get the raw data before somebody could modify them. We try to build our authentic software in a very deep layer of the process, so we would like to build it in the driver level.

1.1 Protocol participants and notations

GPS is the Global Positioning System. The satellites of this system provide the data from which the GPS receiver calculates the coordinates of the actual position.

MD is the Mobile Device. The device with a GPS receiver, we make the positioning and the authentication with the help of this.

CA is the Certification Authority. It provides the authenticate time and location stamp, this is an organization, which is independent from the measurement and can guarantee that nobody modified the results we got.

AS is the Authentic Software. This software makes the authentication for the raw data (which come from the satellites) and for the calculated GPS coordinates.

CS is the Calculator Software. This software calculates the GPS coordinates of the actual position from the raw data come from the satellites.

M is the text or photo or some other data that we want to authenticate with a location-stamp.

h(…) is the hash function.

H is the hash value of the M.

c(…) is the calculator function, which calculates the current position from the raw data that come from the satellites.

RD is the raw data come from one of the satellites of Global Positioning System.

GPSc is the GPS coordinate calculated by the calculator software.

SAS(…) is the signature of the data in parenthesis with the private key of the authentic software.

SCA(…) is the signature of the data in parenthesis with the private key of the certification authority.

VAS(…) is the verification of the data in parenthesis with the public key of the authentic software.

VCA(…) is the verification of the data in parenthesis with the public key of the certification authority.

DOI: 10.21791/IJEMS.2017.2.1.

si is the i-th signed data.

TIME is the time information.

n is the nonce value

ALS is the authentic location stamp, generated by the certification authority.

1.2 Protocol

1. MD calculates the hash value of M : H = h(M)
2. MD→AS : H||M
3. AS digitally signs the H with its private key: $s_1 = S_{AS}(H)$
4. GPS→AS : RD
5. AS digitally signs the hash value of RD with its private key: $s_2 = S_{AS}(h(RD))$
6. AS→CS : RD
7. CS calculates the actual position from RD : GPSc = c(RD)
8. AS←CS : GPSc
9. AS digitally signs the hash value of GPSc with its private key: $s_3 = S_{AS}(h(GPSc))$
10. AS concatenates H, s_1 , RD, s_2 , GPSc and s_3 and takes its hash value and then digitally signs this hash value with its private key: $s_4 = S_{AS}(h(H||RD||GPSc||s_1||s_2||s_3))$
11. AS→CA : $H||RD||GPSc||s_1||s_2||s_3||s_4$
12. CA verifies that the raw data were signed by AS and CA verifies that GPSc can be computed from RD. $V_{AS}(s_2)$, c(RD) =? GPSc
13. if the answer is true for both questions in 12. then CA generates a nonce value: n
    $s_5 = S_{CA}(h(H||RD||GPSc||s_1||s_2||s_3||s_4||n))$
    AS←CA : $H||RD||GPSc||s_1||s_2||s_3||s_4||n||s_5$
14. AS digitally signs the nonce value with its private key: $s_6 = S_{AS}(h(n))$
    AS→CA : $H||RD||GPSc||s_1||s_2||s_3||s_4||n||s_5||s_6$
15. CA verifies that really the AS signed the nonce value: $V_{AS}(s_6)$
16. if the answer is true in 15. then CA makes the authentic location-stamp:
    $ALS=S_{CA}(h(H||RD||GPSc||n||s_1||s_2||s_3||s_4||s_5||s_6||TIME))$
    AS←CA : ALS
17. AS verifies that really the CA signed the location-stamp that it got: V_{CA} (ALS)
18. if the answer is true in 17. then AS accepts the authentic location-stamp
19. if the answer is false in 17. then AS starts a new location-stamp request with 4.
20. if the answer is false in 15. then CA rejects to generate the authentic location-stamp
    AS←CA : rejection
21. if the answer is false in 12. then CA rejects to generate the authentic location-stamp
    AS←CA : rejection

1.3 Protocol description

The protocol, described in the previous subsection, have three important participants, these are the satellites of the Global Positioning System, the mobile device and the certification authority. The mobile device generates a print of the data, - we want to stamp with an authentic location stamp- initially with an eligible hash function, this is necessary because of the digital signing.

After this the authentic software, which is built in the driver of the mobile device, gets the data from the three GPS satellites, and then it digitally signs these data with its own private key presently. This

DOI: 10.21791/IJEMS.2017.2.1.

signing is required in order that nobody is able to falsify during the computational process the raw data arriving from the satellites. The authentic software located in the driver of the mobile device so it can protect the data from the attack of any software installed on the operation system of the mobile device.

Once the authentic software digitally signed and stored the raw data, sends it to the calculator software. The calculator software calculates the current GPS coordinates from the present raw data and sends back the result to the authentic software. The authentic software digitally signs these data too.

Now we arrived at the point that the authentic software is able to ask for an authentic time stamp from the certification authority. So, the authentic software sends a request to the certification authority, this request contains the data hash value, the raw data and the calculated coordinates concatenated and digitally signed with its own private key.

Then the certification authority generates a nonce value in order to ensure the freshness of the protocol and gives it back to the software. The authentic software appends the nonce to the previous request and turns it back to the certification authority, which checks that really the authentic software sent the request. If the result of the verification is right then the certification authority checks that GPSc can be computed from the raw data, if the answer is true, then it generates the location stamp, which also includes a time stamp too.

DOI: 10.21791/IJEMS.2017.2.1.

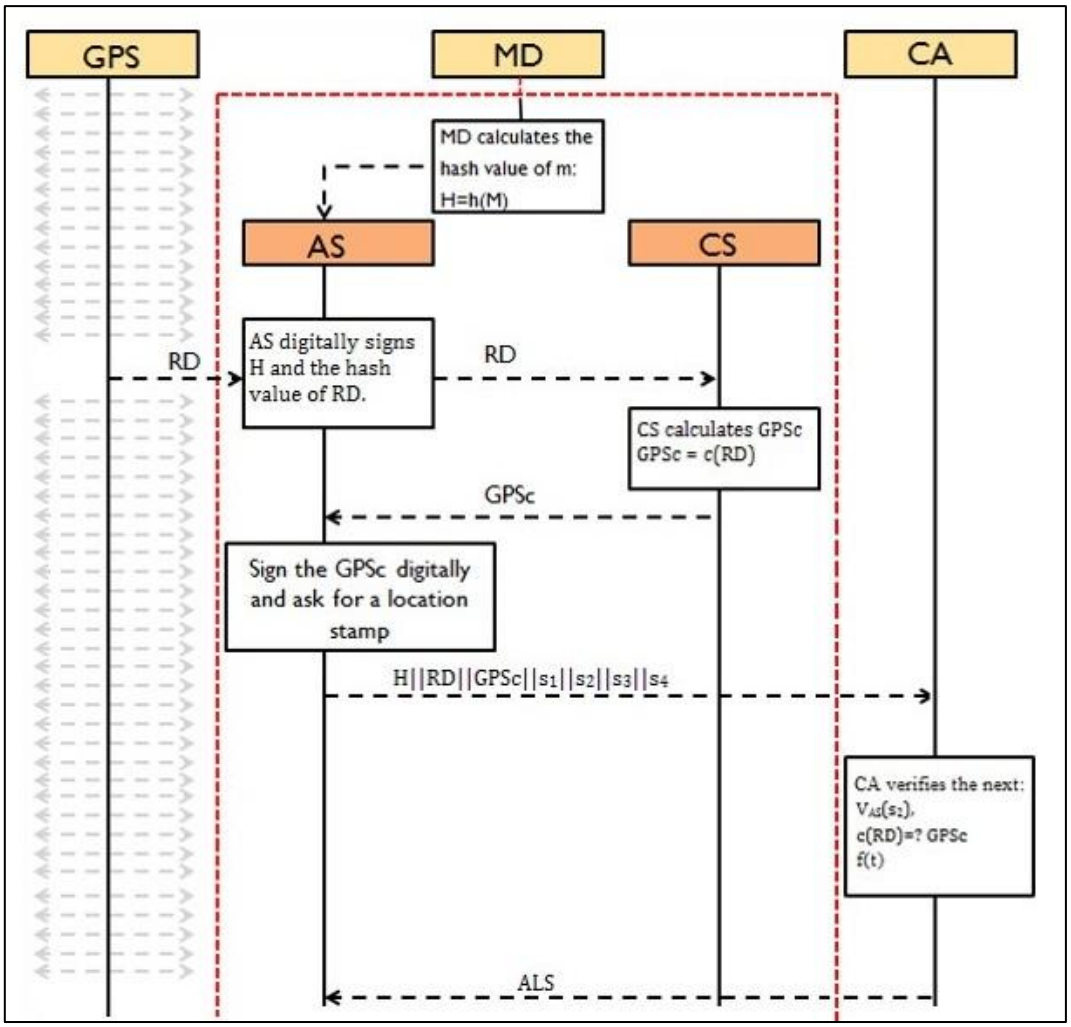

Figure 1. "Location stamping" protocol

2. Applied-π calculus and ProVerif

In this section, I give a brief introduction to the basics of the Applied-π calculus and ProVerif. The Applied-π calculus is a special language, which can model concurrent processes of communication protocols, and interactions between these processes. It was introduced in 2001 by Abadi and Fournet, and it is based on the π-calculus. The Applied-π calculus offers wide collections of complex cryptographic primitives, like digital signatures, hash functions, proofs of knowledge etc. This calculus was made to make formal analysis of cryptographic communication protocols, it models cryptographic primitives with functions and equational theories. The calculus has been used in analyzation of several types of protocols, like certified email [11], electronic voting [12] or authentication protocols [13]. Hereunder in Table 1. the basic notations of the calculus can be seen, for detailed information consider examining [5].

Applied-π	
$L, M, N, T, U, V ::=$	*Terms*
$a, b, c, \ldots, k, \ldots, m, n, \ldots, s$	*Names*
x, y, z	*Variables*

DOI: 10.21791/IJEMS.2017.2.1.

$f(M_1, \ldots, M_l)$	*Function application*
$P, Q, R ::=$	*Processes (or plain processes)*
0	*Null process*
$P \mid Q$	*Parallel composition*
$!P$	*Replication (infinite number of copies of P running parallel)*
$\nu n. P$	*Name restriction ("new") (makes a new, private name n then behaves as P)*
$if\ M = N\ then\ P\ else\ Q$	*Conditional*
$u(x). P$	*Message input*
$\bar{u}\langle N\rangle. P$	*Message output*

Table 1. Basic notations in Applied-π calculus

ProVerif is a software tool, which based on the Applied-π calculus, and it is capable of analysing properties reachability, correspondence assertions, and observational equivalences in an automatic way. With the help of ProVerif you can formalize many different cryptographic primitives, like encryption, signatures, hash functions. If ProVerif results that a property is satisfied, then the model guarantees the property, but ProVerif may not be able to prove a property that holds. For security evaluations ProVerif uses queries that might be a fact or a correspondence. I define events in the model as important stages and I test whether if an event *a* has been executed, then event *b* has been previously executed before. In this lecture, I would like to prove some security properties for our „Location-stamping" protocol. I survey the following security requirements: authenticity and data integrity. Hereunder in Table 2. the basic notations of the ProVerif can be seen, for detailed information consider examining [2].

ProVerif	
a, b, c, k, m, n, s	*Names (a,b,c usually denotes chanells)*
x, y, z	*Variables*
$const\ c.$	*Constant (c is the name of the constant)*
$fun\ f/n\ .$	*Constructor (f is the name the constructor, n is the number of its arguments)*
$reduc\ g(f(m), k) = m.$	*Destructor(g is the name of the destructor, m is the argument of the constructor and k is the argument of the destructor)*
$free\ a$	*C*hannel
let	*Keyword to create processes*
new	*Keyword to generate new value to variables*
$event\ e(x)$	*Event (e is the name and x is the argument of the event)*
$in(a, m)$	*Replication (infinite number of copies of P running parallel)*
query ev: $e_1(x)$ ==>	*Correspondence assertion*
ev: $e_2(y)$.	*(for each occurrence of $e_1(x)$, there is a previous occurrence of $e_2(y)$for some y, where $e_1(x)$,$e_2(y)$ are events)*
query evinj: $e_1(x)$ ==>	*Injective correspondence*
evinj: $e_2(y)$.	(correspondence assertion for one-to-one relationship)

Table 2 . Basic notations in ProVerif

DOI: 10.21791/IJEMS.2017.2.1.

3. ProVerif Formalization

3.1. Declarations

```
(*Channel*)
free a.
(*Hash function*)
fun h/1.
(*Digital signature*)
fun pk/1.
fun sign/2.
reduc checkSign(sign(m,k),pk(k)) = m.
private reduc getMess(sign(m,k)) = m.
(*Coordinate calculator function*)
fun calc/1.
reduc checkCalc(calc(RD)) = RD.
```

In the foregoing code “a” is a public communication channel on which the participants can communicate, “h” is the hash function I use to shorten the messages. Furthermore to formalize the protocol I defined a method “pk” to create public key to a digital signature process, with secret key as an input parameter, the digital signature process is denoted by “sign”. To be able to check the correctness of the digital signatures I created the “checkSign” and to get back the original message from the signed message the “getMess” methods. “calc” function can calculate accurate GPS coordinates from the raw data given by the satellites, and “checkCalc” is able to verify the result of the calculation of the GPS coordinates. To formalize the protocol correctly, I need one main process and two subprocesses one for each participant -Mobile Device and Certification Authority-, in this proof I assume that the Mobile Device is reliable and works correctly.

3.2. Mobile Device Process

```
let processMD =
  new m;
        event startedMD(sskMD);
  let h1 = h(m) in
  let s1 = sign(h1,sskMD) in
  new RD;
  let h2 = h(RD) in
  let s2 = sign(h2,sskMD) in
  let GPSc = calc(RD) in
  let h3 = h(GPSc) in
  let s3 = sign(h3, sskMD) in
  let h4 = h((h1,RD,GPSc,s1,s2,s3)) in
  let s4 = sign(h4,sskMD) in
  out(a,m);
  out(a,(h1,RD,GPSc,s1,s2,s3,s4));
        event requestMD(h1,RD,GPSc,s1,s2,s3,s4);
```

DOI: 10.21791/IJEMS.2017.2.1.

```
in(a,(ALS,t2));
      event getALS(ALS,spkCA);
if getMess(ALS) = (h((h1,RD,GPSc,s1,s2,s3,s4)),t2) then
       event DataTheSameMD();
    if checkSign(ALS,spkCA) = (h((h1,RD,GPSc,s1,s2,s3,s4)),t2) then
                        event AcceptSignatureMD(spkCA).
```

Mobile Device Process starts with creating a new message, and then follow the process of the protocol explained earlier in the 1.2 section. During the process I created events which stand there like cornerstones of the correctness proof.

Name	***Parameter***	***Explanation***
startedMD/1	*Secret key of the Mobile Device*	*Marks that the Mobile Device get a new message to location stamp.*
requestMD/7	*Hash and original values of the signed message, raw data, and GPS data.*	*Marks that the Mobile Device asked for a new location stamp for the Certification Authority.*
getALS/2	*Authenticated Location Stamp and public key of the Certification Authority.*	*Marks that the Mobile Device get the Authenticated Location Stamp from the Certification Authority.*
DataTheSameMD/0		*Marks that the location stamped message is identical as the original one.*
AcceptSignatureMD/1	*Private key of the Certification Authority.*	*Marks that the Certification Authority made the signature.*

Table 3. Events in the Mobile Device Process

3.3. Certification Authority Process

```
let processCA =
        in(a,m);
                event startedCA(spkCA);
        in(a,(h1,RD,GPSc,s1,s2,s3,s4));
                event getRequest(h1,RD,GPSc,s1,s2,s3,s4);
        if getMess(s1) = h1 then
                if getMess(s2) = h(RD) then
                        if getMess(s3) = h(GPSc) then
                                event DataTheSameCA();
if checkSign(s2,spkMD) = h(RD) then
                event AcceptSignatureCA(sskMD);
                if checkCalc(GPSc) = RD then
                        event AcceptCoordinateCA(RD);
let h5 = h((h1,RD,GPSc,s1,s2,s3,s4)) in
        let ALS = (sign((h5,t2),sskCA)) in
        out(a,(ALS,t2));
                event sendALS(ALS,sskCA).
```

DOI: 10.21791/IJEMS.2017.2.1.

Certification Authority Process go through the steps of the protocol I explained in 1.2 section and contains several events to help prove the secure properties of the protocol.

Name	***Parameter***	***Explanation***
startedCA/1	*Secret key of the Certification Authority*	*Marks that the Certification Authority get a new message from the Mobile Device to create a location stamp on it.*
getRequest/7	*Hash and original values of the signed message, raw data, and GPS data.*	*Marks that the Certification Authority get the request with the right properties.*
DataTheSameCA/0		*Marks that the message sent by the Mobile Device is the same which is in the request.*
AcceptSignatureCA/1	*Private key of the Mobile Device.*	*Marks that the message sent in the request is identical as the original one.*
AcceptCoordinateCA/1	*Raw satellite data.*	*Marks that the GPS coordinate sent in the request can calculated from the raw data sent in the request.*

Table 4. Event in the Certification Process

3.4. Main Process

```
process
  new sskMD;
  new sskCA;
  let spkMD = pk(sskMD) in
  out(a,spkMD);
  let spkCA = pk(sskCA) in
  out(a,spkCA);
  ((!processCA)| (!processMD))
```

In the main process, I create two secret keys to the participants, generate two public keys to them and put the public keys on the channel, it will allow both participant to be able to check the source of the messages. Then I will start the subprocesses, and run it continuously.

4. Proved properties

Correctness proof with Applied-π, and ProVerif based on the declared events in the processes, I investigated the correspondence usually between two of the events. There are two type of relations that can be examine, the correspondence assertion which is a one to many and injective

DOI: 10.21791/IJEMS.2017.2.1.

correspondence which is a one to one relation. The differencies between the notation of these queries can be seen in table 2. The basic difference between the two type of query that the injective one can prove that if an event occurred then the other event occurred the same times, while the other one examine only the occurrence not the number of it.

4.1. Authenticity

Authenticity is the property which prove that the origin of some information is the one who claim it to be.

Definition 1.
Let us state that our protocol fulfils Authenticity if the following conditions hold:
The Mobile Device successfully authenticates the Certification Authority and the Certification Authority successfully authenticates the Mobile Device.

Theorem 1.
Our „Location-stamping" protocol accomplish the Authenticity property.

Proof 1.
CA and MD authentication are achieved because the following queries return logical value true:
(*Proof of the authentication of MD to CA*)
query ev:AcceptSignatureCA(x) ==> ev:startedMD(x).
(*Proof of the authentication of CA to MD*)
query ev:AcceptSignatureMD(x) ==> ev:startedCA(x).

4.2. Data Integrity

Data Integrity is a property which provide that a message is not altered through any process.

Definition 2.
Let us state that our protocol fulfils Data Integrity if the following conditions hold:
The M(message), RD(raw data) and the GPSc(GPS coordinate) sent by the MD has not been tampered through the transmission and the ALS(authentic location stamp) sent by the CA has not been tampered through the transmission.

Theorem 2.
Our „Location-stamping" protocol accomplish the Data Integrity property.

Proof 2.
Data Integrity is achieved because the following queries return logical value true:
(*Proof of data integrity for CA*)
query ev: DataTheSameCA().
(*Proof of data integrity for MD*)
query ev: DataTheSameMD().

DOI: 10.21791/IJEMS.2017.2.1.

5. Complexity

Through generating a location stamp I need to calculate hash printing for six times, apply digital signature for seven times, verify correctness of digital signatures for three times and calculate GPS coordinates from raw data for two times. To determine the computational complexity of the location stamping protocol I need to consider the complexity of different hash functions and digital signature schemes too. Based on [10] the computational complexity of the RSA is $O(n^3)$, and ElGamal digital signature schemes is O(log n),where n is the size of the modulus, which is also the size of the key. I can skip the complexity of the GPS coordinate calculation of the result, because the size of the input data for this part of the algorithm never will change, nor the size of the input message nor the type of the signature or hash function affect to it. Computational complexity of different type of hash function like Message Digest Algorithm (MD) and Secure Hash Algorithm (SHA) is O(n). That means, the complexity of our algorithm is based only on the type of the digital signature.

Digital signature	***Complexity of the location stamping method***
RSA	$O(n^3)$
ElGamal	*O(n)*

Table 5. Complexity.

6. Conclusion

As I considered earlier the protocol proposed in [1] fulfils all the properties which is expected from a well prepared cryptographic communication protocol, like authenticity and data integrity. Although it is true there are other important properties as unreusability, no-framing or non-repudiation that still requires proofing in the case of our method.

References

[1] É. Ádámkó, A. Pethő. "Location-stamp for GPS coordinates." Acta Universitatis Sapientiae, Informatica 5.1 (2013): 63-76.

[2] B. Blanchet, B. Smyth. "ProVerif: Automatic cryptographic protocol Verifier, user manual and tutorial." (2011).

[3] M. D. Ryan, B. Smyth. "Applied pi calculus." (2011).

[4] M. T. Goodrich, R. Tamassia. Algorithm design: foundation, analysis and internet examples. John Wiley & Sons, 2006.

[5] M. Abadi, C. Fournet. "Mobile values, new names, and secure communication." ACM SIGPLAN Notices 36.3 (2001): 104-115.

DOI: 10.21791/IJEMS.2017.2.1.

[6] L. Aszalós, A. Huszti. "Payment approval for PayWord."Information Security Applications. Springer Berlin Heidelberg, 2012. 161-176..

[7] A. Sanjeev, B. Barak. Computational complexity: a modern approach. Cambridge University Press, 2009.

[8] Imem, Ali Al. "Comparison and evaluation of digital signature schemes employed in ndn network." arXiv preprint arXiv:1508.00184 (2015).

[9] D. Eastlake, P. Jones. "US secure hash algorithm 1 (SHA1)." (2001).

[10] K. Hansen, T. Larsen, and K. Olsen. "On the efficiency of fast rsa variants in modern mobile phones." arXiv preprint arXiv:1001.2249 (2010).

[11] Abadi, Martín, and Bruno Blanchet. "Computer-assisted verification of a protocol for certified email." *International Static Analysis Symposium*. Springer Berlin Heidelberg, 2003.

[12] Kremer, Steve, Mark Ryan, and Ben Smyth. "Election verifiability in electronic voting protocols." *European Symposium on Research in Computer Security*. Springer Berlin Heidelberg, 2010.

[13] Abadi, Martín, Bruno Blanchet, and Cédric Fournet. "Just fast keying in the pi calculus." *ACM Transactions on Information and System Security (TISSEC)* 10.3 (2007): 9.