Formal Validation of OFEPSP+ with AVISPA*

Jorge L. Hernandez-Ardieta, Ana I. Gonzalez-Tablas, and Benjamin Ramos

Department of Computer Science, University Carlos III of Madrid
Avda. de la Universidad 30, 28911 Leganes, Spain.
jlopez.ha@gmail.com, aigonzal@inf.uc3m.es, benja1@inf.uc3m.es

Abstract. Formal validation of security protocols is of utmost importance before they gain market or academic acceptance. In particular, the results obtained from the formal validation of the improved Optimistic Fair Exchange Protocol based on Signature Policies (OFEPSP+) are presented. OFEPSP+ ensures that no party gains an unfair advantage over the other during the protocol execution, while substantially reducing the probability of a successful attack on the protocol due to a compromise of the signature creation environment. We have used the Automated Validation of Internet Security Protocols and Applications (AVISPA) and the Security Protocol ANimator for AVISPA (SPAN), two powerful automated reasoning technique tools to formally specify and validate security protocols for the Internet.

Key words: Fair exchange, security protocol, formal validation, AVISPA, SPAN.

1 Introduction

Formal validation of security protocols is of utmost importance before they gain market or academic acceptance. Some standard and widely used security protocols for the Internet have been proved to suffer from critical design flaws that an attacker can exploit to subvert their security. The reason is that their security goals were merely informally evaluated, obviating potential attack paths. Automated reasoning techniques are commonly used to evaluate the protocols in a formal way, increasing the assurance respecting the purported security. In this sense, the Automated Validation of Internet Security Protocols and Applications (AVISPA) [2] and the Security Protocol ANimator for AVISPA (SPAN) [12] tools have been used to validate the correctness and safety of the improved Optimistic Fair Exchange Protocol based on Signature Policies (OFEPSP+).

OFEPSP [13], the protocol from which OFEPSP+ has been designed, is a protocol oriented to Internet transactions where two parties need to exchange information in a fair and secure manner. The origin sends a signed message to

* The authors would like to thank the AVISPA project team, and specially Laurent Vigneron and Luca Viganò, for their useful comments on the preliminary versions of OFEPSP+ HLPSL specification. The authors wish also to thank the reviewers for their valuable remarks.

the receiver while the receiver sends back a proof of receipt of the message. Therefore, both parties are making a commitment in the transaction: the origin cannot repudiate having sent the message and the receiver cannot repudiate its reception. As a fair exchange protocol, OFEPSP ensures that no party gains an unfair advantage over the other during the protocol execution. Therefore, either both parties obtain the expected information or none of them obtains any useful information from the other. As an optimistic protocol, a Trusted Third Party (TTP) is included in the design but participating only when a party's misbehavior or protocol error occurs.

Many fair exchange protocols found in literature are designed using symmetric encryption, assuring the undisclosure of the message sent by the origin until the receiver has made a commitment in the transaction [16]. In our case, OFEPSP is based on signature policies [10]. A signature policy is a document that collects a set of rules to create and validate electronic signatures, and under which an electronic signature can be determined to be valid in a particular transaction context. Signature policies are defined both in human readable form and structured form using an agreed syntax and encoding, like ASN.1 or XML. The signature policy reference is included as a signed property in each signature performed by the parties, allowing any verifier to ascertain if the signature matches the requirements imposed in the policy. Therefore, signature creation and verification processes can be completely carried out in an automatic and transparent way in accordance with the signature policy rules.

In order to tie down the origin and receiver respecting the exchanged information, most protocols use digital signatures. Due to the cryptographic basis of digital signatures, a correctly verified signature is known to have been computed with a particular private key. The common (and mistaken) established assumption is that the entity which owns the private key cannot repudiate having performed such a signature as nobody else could have computed it. If applied to fair exchange protocols, the signed information sent by the origin and the proof of receipt cannot be repudiated by the corresponding parties. However, several feasible and practical attacks on the signature creation environment are found in the literature [1, 11, 14, 15, 17]. Therefore, an attacker gaining access to the origin's private key or just the signature creation environment could impersonate her in a transaction, no matter how the underlying protocol is designed.

To reduce this risk, we improved OFEPSP, dividing the origin's environment in a Signature Creation Environment (SCE) and a Transaction Confirmation Environment (TCE). This division substantially reduces the probability of a successful attack on the protocol as both environments are needed to create the valid evidence. The security of both the SCE and the TCE can be compromised. However, the probability of a successful distributed attack on SCE and TCE by two different malwares, and which collaborate in order to undermine the protocol security, is, in any case, substantially lower that the probability of compromising the security of a single environment. OFEPSP+ assures that an attacker will not gain any benefit from a potential security compromise on either the SCE or the TCE, enforcing the non-repudiation property of the evidence generated during a

protocol run. To the best of our knowledge, this is the first time a fair exchange protocol takes the security of the parties' environments into consideration for the overall protocol design. OFEPSP+ can be applied to a variety of scenarios like contract signing or e-commerce transactions, where the origin corresponds to an end user using a potentially compromised SCE, like her Personal Computer.

In this article, OFEPSP+ is briefly presented and the formal validation of the protocol detailed. The preliminary results are promising, but some work must still be done to obtain a high assurance of the protocol security. In this first approach, the formal validation covered masquerading and integrity attacks. Fairness could not be modeled yet taking into account standard AVISPA features. Nonetheless, future work will cover the use of additional predicates and special goal formulas in order to incorporate fairness in the validation.

The article is organized as follows. The next Sect. 2 describes OFEPSP+ protocol. Section 3 covers the formal analysis of the protocol by means of AVISPA and SPAN tools. And finally, we conclude the article in Sect. 4.

2 The improved Optimistic Fair Exchange Protocol based on Signature Policies

This Sect. describes the improved Optimistic Fair Exchange Protocol based on Signature Policies (OFEPSP+). The basic notation and definitions are given first. Afterwards, OFEPSP+ set of protocols is explained in "Alice & Bob" notation.

2.1 Basic Notation and Definitions

Following notation is necessary for the correct understanding of further Sects.:

SP	Signature policy used during the protocol
$X \rightarrow Y : m$	Party X sends message m to party Y
$X \leftarrow Z : SP$	Party X retrieves SP from a repository located at party Z
$S_x\,(m)$	Signature of party x generated on message m
$S_x\,(m\|SP)$	Signature of party x generated on message m under SP conditions

$POO = S_O\,(m, \ell, tpl\_id|SP)$
Proof of origin of message m.
$POR = S_R\,(m, \ell, tpl\_id|SP)$
Proof of receipt of message m.
$NRO = S_O\,(POR|SP)$
Non-repudiation evidence of origin of message m generated on POR.
$NRR = S_R\,(NRO|SP)$
Non-repudiation evidence of receipt of message m generated on NRO.
$NRA_O = S_O\,(NRR|SP)$
Non-repudiation evidence of acknowledgment generated by origin on NRR. This is the valid evidence that completes the protocol.

$NRA_{TTP} = S_{TTP}(NRR|SP)$
Non-repudiation evidence of acknowledgment generated by the TTP on NRR. This is the valid evidence when the recovery protocol has been executed.

Each protocol run is identified by a protocol identifier ℓ. A template is used by the parties in order to fix the information to be sent by the origin. This template is referenced by the template identifier *tpl_id*. The template must be defined by the receiver according to the transaction needs. The message m sent by the origin must be further processed by the receiver taking into account the template information.

2.2 Main Protocol

OFEPSP+ consists of one main protocol, explained in this Subsect., and two subprotocols, called recovery subprotocol and abort subprotocol, which are further explained. During the exchange process, the origin must use two different platforms, called the Signature Creation Environment (SCE) and the Transaction Confirmation Environment (TCE). SCE is the platform where the signature application is installed and used by the origin to create electronic signatures on her behalf. The origin must use a hardware cryptographic device with signing capabilities (e.g. smart card) for that purpose. This cryptographic device is regarded as part of the SCE platform. On the other hand, the origin acknowledges the performed signatures by means of a second cryptographic device (and thus different signing key) used in a different platform with another signature application (e.g. mobile device with cryptographic capabilities), that is, the TCE.

In the main protocol, the origin initiates the transaction by sending the signed message to the receiver by means of SCE. Afterwards, the origin confirms the initiated transaction by using TCE. The receiver will exchange several intermediate evidences with both the SCE and TCE, until the final valid evidence NRA is generated. Next, the main protocol is formalized using the notation of SubSect. 2.1 above:

(1) $O_{SCE} \leftarrow R : tpl\,[tpl\_id]\,, S_R\,(tpl\,[tpl\_id])$
First, the origin (O) requests the template identified by *tpl_id* to the receiver (R) (1) by means of the SCE platform.

(2) $O_{SCE} \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}\,(SP)$
In next step (2) the origin retrieves the signature policy SP necessary to communicate with the receiver. Once the origin has obtained the template and the signature policy, she can produce the message and POO taking into account this information.

(3) $O_{SCE} \rightarrow R : m, \ell, tpl\_id, POO$
Afterwards (3), the origin starts the protocol itself by sending the message m, a unique protocol identifier ℓ, the template reference and the POO.

(4) $R \leftarrow$ *TTP-SP*: $SP, S_{TTP-SP}(SP)$
(5) $R \rightarrow O_{TCE} : m, \ell, tpl\_id, POO, POR$

The receiver retrieves the signature policy (4), if not obtained yet, and validates the received POO. Afterwards, the receiver generates and sends the POR to the origin's TCE (5). The receiver must also send all the information received from the origin in step (3) in order to allow her to validate the initiated transaction using the second platform TCE. Step (4) can be avoided for efficiency purposes if the receiver accesses the TTP-SP once and afterwards manages a local copy of the signature policy, provided that it is within its validity period.

(6) $O_{TCE} \rightarrow R : NRO$

The origin must generate the NRO only if the information received in step (5) corresponds to a desired transaction and POO and POR are correctly verified.

(7) $R \rightarrow O_{SCE} : NRR$

Once the origin has confirmed the transaction by means of the NRO, the receiver sends the NRR to the origin's SCE platform (7).

(8) $O_{SCE} \rightarrow R : NRA_O$

In the last step (8) the origin completes the transaction by sending the NRA to the receiver.

Although not shown above, evidences POO, POR, NRO, NRR and NRA must be time-stamped. The time-stamping procedure must be carried out according to known standards, and implies the participation of a Time-Stamping Authority (TSA). The time-stamp is an assertion of proof given by the TSA that the datum existed before the specified time.

2.3 Recovery Subprotocol

The recovery subprotocol allows the receiver to obtain evidence NRA in case of a protocol interruption or origin's misbehavior, and must be executed if the receiver does not receive the NRA within a specific time interval. OFEPSP+ recovery subprotocol consists of next steps:

(1) $R \rightarrow TTP : H(m, \ell, tpl\_id), \ell, POO, POR, NRO, NRR$
if (protocol aborted) then
(2a) $TTP \rightarrow R : S_{TTP}(S_O(abort, \ell | SP) | SP)$
else
(2b) $TTP \leftarrow$ *TTP-SP*: SP
(3b) $TTP \rightarrow R, O_{SCE}, O_{TCE} : NRA_{TTP}$

In (1) the receiver sends the produced evidences POO, POR, NRO and NRR to the TTP. Also, and in order to protect the privacy of the parties, the information signed in POO and POR - the message, the protocol identifier and the template reference - is not sent in plain text but the hash of their concatenated

values. Yet the TTP is still able to verify POO and POR by directly using the hash, provided that a digital signature scheme based on public key cryptography is used (i.e. RSA, DSA, ECDSA). The TTP must decrypt POO and POR - using the corresponding public keys -, obtaining the hash of the signed information, which must correspond to the value of $H(m, \ell, tpl\_id)$. ℓ is sent in (1) to allow the TTP to retrieve and update the information associated to the transaction.

If the protocol has already been aborted, the TTP merely forwards the abort evidence to the receiver (2a). On the other hand, the TTP generates the NRA_{TTP} taking into account the referenced signature policy - (2b) and (3b) -, but only in the first request. The evidence must be stored in a local database along with the received information. Subsequently, the TTP will reuse it, improving the efficiency.

It is important to remark that the signatures that correspond to the evidences generated in the protocol (POO, POR, NRO, NRR and NRA) must be generated according to electronic signature standards (please refer to [13] for further information). Thereby, a reference to the signature policy used in the protocol is included as a signed property in the specific electronic signature format chosen for the transaction (i.e. XAdES, CAdES). This permits the TTP to know and retrieve the signature policy and avoids an attacker to modify the referenced signature policy.

2.4 Abort Subprotocol

The abort subprotocol allows the origin to abort the protocol execution if a receiver's malicious behavior is suspected or an error during the protocol run has occurred. OFEPSP+ abort subprotocol consists of next steps:

(1) $O_{SCE|TCE} \rightarrow TTP : abort, \ell, S_O(abort, \ell|SP)$
if (recovery protocol executed) then
(2a) $TTP \rightarrow O_{SCE|TCE} : NRA_{TTP}$
else
(2b) $TTP \leftarrow$ *TTP-SP*: SP
(3b) $TTP \rightarrow O_{SCE|TCE} : S_{TTP}(S_O(abort, \ell|SP)|SP)$

For efficiency purposes, $S_{TTP}(S_O(abort, \ell|SP)|SP)$ is only generated in the first time (2b) and (3b), being reused in subsequent executions of the abort subprotocol. On the other hand, if the protocol has been recovered (2a), the TTP just retrieves the NRA_{TTP} from its data base, forwarding it to the origin.

3 OFEPSP+ Validation with AVISPA and SPAN

This Sect. covers the validation process using the Automated Validation of Internet Security Protocols and Applications tool (AVISPA) and the Security Protocol ANimator for AVISPA (SPAN).

AVISPA [2, 3] provides a suite of applications for building and analyzing formal models of security protocols. AVISPA incorporates four backends: the On-the-Fly Model-Checker (OFMC) [6], the Constraint-Logic-based model-checker (CL-AtSe) [18], the SAT-based Model-Checker (SATMC) [4], and the Tree Automata based Automatic Approximations for the Analysis of Security Protocols (TA4SP) [7]. These modules implement different automated reasoning techniques to formally analyze the protocol specification. On the other hand, SPAN [12] offers a graphical user interface that allows the protocol designer to easily interact with AVISPA capabilities.

Protocol models must be written in the High Level Protocol Specification Language (HLPSL) [5, 8], which allows the protocol designer to describe the security protocol and specify its intended security properties. HLPSL details have been omitted for lack of space.

AVISPA adopts the standard intruder model of Dolev and Yao (DY model) [9], in which the intruder has complete control over the network but cannot break cryptography. The intruder may intercept, analyze, and/or modify messages (as far as he knows the required keys), and send any message he composes to whoever he pleases, posing as any other agent. The goal of OFEPSP+ validation was to check the correctness and safety of the protocol respecting DY model.

The validation methodology followed can be summarized in next 3 steps:

1. OFEPSP+ specification in HLPSL.
2. HLPSL correctness verification.
3. OFEPSP+ security validation.

3.1 OFEPSP+ Specification in HLPSL

OFEPSP+ complexity lies in the existence of four entities. The first two entities, SCE and TCE, are managed by the origin of the protocol, but, in HLPSL, had to be considered as two different roles played by a different agent each. The other two entities, Receiver and TTP, were modeled as the receiver and the TTP roles, respectively, played by the corresponding agents. Each role implemented its own state transition system according to the steps indicated in the protocol described in Sect. 2. Thanks to the *set* structure available in HLPSL, the TTP's evidence database could be modeled and the TTP behavior accurately defined.

Restrictions Applied
The protocol steps described in Sect. 2 could be modeled, except next four issues.

Signature Policy-based Design
OFEPSP+ fairness property is enforced by the signature policy-based design, assuring that an incomplete evidence (POO, POR, NRO or NRR) does not tie down any of the parties involved. The existence of NRA is imperative to prove the commitment made by both parties, and its creation (either by the origin or the TTP) must fulfill the policy constraints and requirements. We found that the usage of signature policies could not be modeled in HLPSL. HLPSL allows

translating an Alice & Bob chart into a more detailed and expressive language. However, not every protocol behavior can be mapped in HLPSL. Therefore, the protocol steps related to the policy retrieval were discarded during the HLPSL definition.

Time-Stamping
Time-stamps are needed in the protocol to allow the parties to know when the recovery or abort subprotocols must be executed. The trigger is a timeout defined in the signature policy for each case. However, HLPSL only allows the establishment of a generic timeout as an incoming message to a role. This feature avoided us to model the specific timeouts, and thus, the time-stamping processes were obviated. We have checked that this constraint has not modified neither the protocol behavior nor its security goals fulfillment.

Template Usage
In OFEPSP+ the origin must create the message m according to the template constraints. This measure restricts the semantics of the signed information, avoiding most of semantic attacks currently developed [1, 14, 15]. However, and as previously mentioned, in this first approach we wanted to evaluate the security measures against DY model. For that reason, the template retrieval by the origin in the first step of the main protocol was discarded as well.

Server Role Capabilities
Finally, the TTP (server) role was initially too complex for the backend analyzers of AVISPA, due to the number of transition combinations considered during the validation process. Note that our TTP is designed as an e-notary, storing and managing the evidences generated during a protocol run in which the TTP takes part. Besides, there were three situations where the TTP could intervene: an abort requested by both the SCE and the TCE, and a recovery requested by the receiver. This led to a huge role definition with 9 transitions. Taking into account that CL-AtSe considers, by default, that each transition can be applied at most 3 times, the backend had to manage 27 possible transitions during the analysis. It seemed to be too complex.

For that reason, we simplified the role server, excluding the TCE from making abort requests. The origin is still able to abort the protocol by using the SCE. Thus, the server role was modeled considering next three states: 0 as the initial state; 1 as the state when an abort has been done first; and 2 as the state when a recovery has been done first. In each state, the server can receive both abort and recovery requests, leading to 6 defined transitions.

As a result of previous modifications applied, the server role cannot respond to several parallel sessions. The transitions are sequential, that is, once a transaction has been aborted or recovered, the server will stay in that state (1 or 2, respectively) for the rest of requests, no matter if they come from another session. Nonetheless, and as will be seen further, we were able to test the protocol with parallel sessions between SCE, TCE and Receiver, looking for possible attacks though the server behaved in this way.

We know that we have limited the possible space of attacks, but the decision was made in order to allow the backends to correctly analyze the protocol.

Analysis Scenarios
HLPSL must cover the definition of two special roles: session and environment. Respecting the session role, we just instantiated the roles sce, tce and receiver with the adequate information. Mention that the agents SCE and TCE, since both are managed by the same origin, own a pre-shared knowledge: the message to be sent. The template reference is also a pre-shared knowledge between the SCE, TCE and the Receiver agents.

One of the most important parts of the HLPSL is the initial knowledge allocated to the intruder. In this sense, and for test purposes, we defined five different template references (tpl_id, tpl_id2, tpl_id3, tpl_id4) and four different messages (msg, msg2, msg3, msg4), assigning the subset {msg2, msg3, tpl_id2, tpl_id3, tpl_id4} as knowledge to the intruder.

Afterwards, we defined several analysis scenarios with different sessions configurations (see section 3.3 for details). Due to the constraints applied to the server role capabilities commented above, we instantiated just one server role in each scenario.

Security Goals
AVISPA supports three types of goals so far: *secrecy_of*, (strong) *authentication_on* and *weak_authentication_on*. In the latter, the piece of data used to authenticate the agent can be reused by an attacker, so reply attacks are not considered by the analyzers. As AVISPA does not explicitly support fairness and non-repudiation security goals, some fair exchange/non-repudiation protocols modeled with AVISPA used *secrecy_of* goal to achieve it. However, OFEPSP+ does not provide confidentiality on any item. Fairness is achieved by means of the signature policy fulfillment, as explained above. Therefore, currently we have only modeled authentication goal respecting the exchanged evidences.

3.2 HLPSL Correctness Verification

In this step, the aim was twofold: verify the syntactic/semantic correctness and executability of the HLPSL specification; and that the HLPSL specification implemented the intended protocol behavior.

For the syntactic, semantic and executability verification, next AVISPA and SPAN tools were used:

HLPSL2IF: This tool translates the HLPSL specification into the Intermediate Format (IF). A successful translation implies that the syntax of the protocol is correct. OFEPSP+ HLPSL file was correctly translated into the corresponding IF.

Protocol simulation: By simulating the protocol with SPAN, the semantic of the protocol's HLPSL is verified and a Message Sequence Chart (MSC) visualized. In our case, the semantic was correctly verified but the MSC could not be shown. It frequently happens when the transition labels and state values are not perfectly set. In any case, it does not imply an error in the specification.

OFMC search tree option: OFMC offers the possibility to browse the search tree through a path indicated by the indexes of the successors to follow. As a result, one can decide which choice point take in a specific point of the search tree deduced from the IF. This option allows the tester to check if every transition can be taken during a protocol run. In our case, every transition could be chosen at some time in the search tree.

CL-AtSe no executability option: CL-AtSe offers the possibility of tracing the protocol specification without being analyzed. The output shows the so called *Initial System State*, representing both the intruder and honest participants states in CL-AtSe just after reading and interpreting the IF file. While the intruder state is just represented by a list of knowledges, the honest participants are described by a set of instantiated roles, called *Interpreted protocol specification*. This option is useful to check that CL-AtSe interprets the protocol transitions as expected. Each role consists in a tree where unary nodes are protocol steps and n-ary nodes are choice points. In our case, each possible transition was represented in the tree.

SATMC check only executability: With this option, SATMC checks on executability of actions/rules without any intruder, allowing the tester to debug the specification. The output trace showed that every rule could be executed.

Session compilation with OFMC: With session compilation (sessco), OFMC finds a replay attack even without a second parallel session. It first simulates a run of the whole system and in a second run, it lets the intruder take advantage of the knowledge learned in the first run. Sessco is also handy for a quick check of executability. However, as stated in AVISPA documentation, if one role can loop (i.e. remain in the same control state forever and make infinitely many steps), sessco is not possible, and OFMC aborts with an error message. That is our case in some transitions of role server, and thus, we could not use this option.

CL-AtSe no executability option and *OFMC search tree option* helped us also to ascertain that the HLPSL specification matched the intended protocol behavior.

3.3 OFEPSP+ Security Validation

The results obtained from validating OFEPSP+ with OFMC and CL-AtSe backends are shown in next Tables 1, 2 and 3. In case of SATMC, the result was always "Inconclusive". Tests reports showed us that SATMC did not find an attack, but it warned that, with SATMC backend, intruder is not allowed to generate fresh terms (as in sce role). As a consequence, attacks based on such an ability would

not be reported. TA4SP was not used because it does not support *sets* up to now. The analysis scenarios referred in Tables below are described in Table 4[1].

Table 1. Validation results with OFMC and CL-AtSe respecting a single session with legitimate agents and single sessions with intruder playing the role of a legitimate agent

Analysis scenario	OFMC	CL-AtSe
cfg1	SAFE	SAFE
cfg2	SAFE	SAFE
cfg3	SAFE	SAFE
cfg4	SAFE	SAFE

Configurations applied in Table 1 were aimed at finding attacks in a normal session (cfg1) or sessions when the intruder impersonates one of the legitimate agents - SCE (cfg2), TCE (cfg3) or Receiver (cfg4).

Table 2. Validation results with OFMC and CL-AtSe respecting parallel sessions with legitimate agents playing different roles

Analysis scenario	OFMC	CL-AtSe
cfg5	SAFE	SAFE(*)
cfg6	SAFE	SAFE
cfg7	SAFE	SAFE
cfg8	SAFE	SAFE

Configurations shown in Table 2[2] were focused on violating the security goals when two coherent parallel sessions are executed (cfg5) and when a legitimate party is playing a role for which is not intended to in case of parallel sessions (cfg6, cfg7 and cfg8). We realized that each participant's identifier had to be included in each evidence generated in order to avoid this sort of attack. In particular, we used the public keys of SCE, TCE, Receiver, and TTP.

Table 3[3] contains the set of configurations where an intruder is playing the role of legitimate agent(s) when two parallel sessions are executed. Note that the knowledge own by the intruder in each configuration differs. The aim was to find possible security goals violations in a session when carried out from an intruder running in another different session. Mention that when the intruder

[1] Intruder is denoted as 'i'. We did not instantiated any session with the intruder playing the role of the server because we consider the TTP to be honest.

[2] OFMC executed with maximum search depth of 17. CL-AtSe executed with -nb 1 option (maximum 1 loop iteration in any trace) for test marked with (*)

[3] Results marked with (*) mean that OFMC was launched with maximum search depth of 17

Table 3. Validation results with OFMC and CL-AtSe respecting parallel sessions with intruder playing as legitimate agent(s)

Analysis scenario	OFMC	CL-AtSe
cfg9	SAFE(*)	SAFE
cfg10	SAFE(*)	SAFE
cfg11	SAFE(*)	SAFE
cfg12	SAFE	SAFE
cfg13	SAFE	SAFE
cfg14	SAFE	SAFE

plays a legitimate role in a session, the goals involving him are not considered by AVISPA (otherwise, he could always achieve an attack).

Based on the results obtained from the tests, our protocol fulfills the security goals G2 - Message authentication (includes message integrity), G17 - Accountability, G18 - Proof of Origin and G19 - Proof of Delivery for evidences POO, POR, NRO, NRR and NRA. Goals description are given in Deliverable 6.1 "List of selected problems" in AVISPA project [2].

Due to our protocol design, evidences are not protected against reply attacks (G3), and thus goal Entity Authentication (G1) is not achieved either. For that reason, only *weak_authentication_on* goal was assigned. We think that including a nonce in the requests generated by the receiver would enforce goals G1 and G3 for NRO, NRR and NRA. However, tests conducted including that nonce did not reach a conclusion, maybe due to the huge number of transitions the backends had to analyze. As a result, our assumption could not be proved.

4 Conclusions and Work in Progress

In this article an improved Optimistic Fair Exchange Protocol based on Signature Policies (OFEPSP+), which evolves a previously presented protocol [13], has been briefly presented and analyzed.

The protocol has been evaluated with the Automated Validation of Internet Security Protocols and Applications (AVISPA) and the Security Protocol ANimator for AVISPA (SPAN) against the intruder model of Dolev-Yao. The methodology followed to specify the protocol in the High Level Protocol Specification Language (HLPSL) and to validate it by means of AVISPA backends has been presented along with the results obtained. We found several problems during the specification stage, like modeling the intended protocol behavior in HLPSL, or checking that the protocol features that could not be defined in HLPSL did not affect the security properties evaluated (e.g. the signature policy-based design).

The preliminary results are promising, but some work must still be done. We are currently working on modifying the transition labels and state values in order to get a protocol simulation, what will allow us to interact in a protocol

Table 4. Analysis scenario configurations for the tests

Analysis scenario	sessions configuration
cfg1	$session\,(sce, tce, r, s, ..., msg, tpl\_id)$
cfg2	$session\,(i, tce, r, s, ..., msg, tpl\_id)$
cfg3	$session\,(sce, i, r, s, ..., msg, tpl\_id)$
cfg4	$session\,(sce, tce, i, s, ..., msg, tpl\_id)$
cfg5	$session\,(sce, tce, r, s, ..., msg, tpl\_id)$ $session\,(sce, tce, r, s, ..., msg, tpl\_id)$
cfg6	$session\,(sce, tce, r, s, \ldots, msg, tpl\_id)$ $session\,(r, tce, sce, s, \ldots, msg, tpl\_id)$
cfg7	$session\,(sce, tce, r, s, \ldots, msg, tpl\_id)$ $session\,(sce, r, tce, s, \ldots, msg, tpl\_id)$
cfg8	$session\,(sce, tce, r, s, \ldots, msg, tpl\_id)$ $session\,(tce, sce, r, s, \ldots, msg, tpl\_id)$
cfg9	$session\,(sce, tce, r, s, \ldots, msg, tpl\_id)$ $session\,(i, tce, r, s, \ldots, msg, tpl\_id)$
cfg10	$session\,(sce, tce, r, s, \ldots, msg, tpl\_id)$ $session\,(sce, i, r, s, \ldots, msg3, tpl\_id3)$
cfg11	$session\,(sce, tce, r, s, \ldots, msg, tpl\_id)$ $session\,(sce, tce, i, s, \ldots, msg4, tpl\_id4)$
cfg12	$session\,(i, tce, r, s, \ldots, msg2, tpl\_id2)$ $session\,(sce, i, r, s, \ldots, msg3, tpl\_id3)$
cfg13	$session\,(i, tce, r, s, \ldots, msg2, tpl\_id2)$ $session\,(sce, tce, i, s, \ldots, msg3, tpl\_id3)$
cfg14	$session\,(sce, i, r, s, \ldots, msg, tpl\_id)$ $session\,(sce, tce, i, s, \ldots, msg, tpl\_id)$

run as an active intruder. Afterwards, we will enrich the test scenarios to increase the level of assurance respecting the security goals fulfillment. Finally, and as the biggest challenge, fairness will be specified by means of special predicates and goal formulas. We hope that these results and current work will allow us to implement a proved secure fair exchange protocol.

Acknowledgments. This research has been partially supported by the Ministry of Industry, Tourism and Trade of Spain, in the framework of the project CENIT-Segur@, reference CENIT-2007 2004. (https://www.cenitsegura.es)

References

1. Alsaid, A., Mitchel, C. J.: Dynamic content attacks on digital signatures. Information Management & Computer Security 13, 4, 328 - 336 (2005)
2. AVISPA: Automated validation of internet security protocols and applications (2003) FET Open Project IST-2001-39252. http://www.avispa-project.org/
3. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Hankes Drielsma, P., Heam, P.-C., Kouchnarenko, O., Mantovani, J., Modersheim, S., von Oheimb, D., Rusinowitch, M., Santos Santiago, J., Turuani, M., Vigano, L., and Vigneron, L.: The AVISPA Tool for the automated validation of internet security protocols and applications. In K. Etessami and S. Rajamani, editors, 17th International Conference on Computer Aided Verification, CAV2005, Lecture Notes in Computer Science, 3576, 281285, Edinburgh, Scotland. Springer (2005)
4. Armando, A., and Compagna. L.: SATMC: a SAT-based model checker for security protocols. In Proc. of the 9th European Conference on Logics in Artificial Intelligence (JELIA04), LNAI, 3229, 730733, Lisbon, Portugal. Springer-Verlag (2004)
5. AVISPA. Deliverable 2.1: The High-Level Protocol Specification Language. Available at http://www.avispa-project.org/publications.html (2003)
6. Basin, D. A., Sebastian, M., and Vigano, L.: Ofmc: A symbolic model checker for security protocols. Int. J. Inf. Sec., 4(3), 181 208 (2005)
7. Boichut, Y., Heam, P.-C., Kouchnarenko, O., and Oehl, F.: Improvements on the Genet and Klay Technique to Automatically Verify Security Protocols. In Proc. of Automated Verification of Infinite States Systems (AVIS04), 1 - 11. ENTCS (2004)
8. Chevalier, Y., Compagna, L., Cuellar, J., Hankes Drielsma, P., Mantovani, J., Modersheim, S., and Vigneron, L.: A High Level Protocol Specification Language for Industrial Security-Sensitive Protocols. In Proc. SAPS04. Austrian Computer Society (2004)
9. Dolev, D., and Yao, A.: On the Security of Public-Key Protocols. IEEE Transactions on Information Theory, 2(29) (1983)
10. ETSI TR 102 041 v1.1.1. Signatures policies report. European Telecommunications Standards Institute (2002)
11. Girard, P., Giraud, J-L.: Software attacks on smart cards. Information Security Technical Report, 8, 1, 55 - 66 (2003)
12. Glouche, Y., Genet, T., Heen, O., and Courtay, O.: A Security Protocol Animator Tool for AVISPA. In ARTIST2 Workshop on Security Specification and Verification of Embedded Systems, Pisa (2006)
13. Hernandez-Ardieta, J. L., Gonzalez-Tablas, A. I., Alvarez, B. R.: An Optimistic Fair Exchange Protocol based on Signature Policies. Computers & Security, 27 (7-8), 309 - 322. ISSN 0167-4048. Ed. Elsevier (2008)

14. Jsang, A., Povey, D., Ho., A.: What You See is Not Always What You Sign, in The proceedings of the Australian UNIX User Group. Melbourne (2002)
15. Kain., K.: Electronic Documents and Digital Signatures. Master Thesis (2003)
16. Kremer, S., Markowitch, O., Zhou, J. An intensive survey of fair non-repudiation protocols. Computer Communications 25, 1601 - 1621 (2002)
17. Spalka, A., Cremers, A. B., Langweg, H.: Trojan Horse Attacks on Software for Electronic Signatures. Informatica **26**, 2, 191 - 203 (2002)
18. Turuani, M.: The CL-Atse Protocol Analyser. In F. Pfenning, editor, Proc. of 17th International Conference on Rewriting Techniques and Applications, RTA, Lecture Notes in Computer Science, Seattle (WA), Aug. Springer (2006)