Intrusion Attack Tactics for the Model Checking of e-commerce Security Guarantees

Stylianos Basagiannis[1] Panagiotis Katsaros[1] Andrew Pombortsis[1]

[1] *Department of Informatics, Aristotle University of Thessaloniki*
54124 Thessaloniki, Greece
{basags,katsaros,apombo}@csd.auth.gr

Abstract. In existing security model-checkers the intruder's behavior is defined as a message deducibility rule base governing use of eavesdropped information, with the aim to find out a message that is meant to be secret or to generate messages that impersonate some protocol participant(s). The advent of complex protocols like those used in e-commerce brings to the foreground intrusion attacks that are not always attributed to failures of secrecy or authentication. We introduce an intruder model that provides an open-ended base for the integration of multiple attack tactics. In our model checking approach, protocol correctness is checked by appropriate user-supplied assertions or reachability of invalid end states. Thus, the analyst can express e-commerce security guarantees that are not restricted to the absence of secrecy and the absence of authentication failures. The described intruder was implemented within the SPIN model-checker and revealed an integrity violation attack on the PayWord micro payment protocol.

KEYWORDS: intrusion attacks, e-commerce protocols, model checking, SPIN

1 Introduction

Model-checking of cryptographic protocols takes place on a model of a small system running the protocol of interest together with an intruder model that interacts with the protocol. Security flaws are found by an appropriate state exploration approach that discovers if the system can enter an insecure state, that is, whether there is an attack upon the protocol.

The basic assumptions are summarized as follows: (i) *The encryption method used is unbreakable*, (ii) *The intruder can prevent any message from reaching its destination* and (iii) *The intruder can create messages of his own*. As a consequence of the foresaid assumptions, model-checking analyses treat any message sent by a honest user as a message sent to the intruder and any message received by a honest user as a message sent by the intruder. This setting refers to a system that becomes a machine that is used by the intruder to generate words (messages). The intruder's behavior is defined as a message deducibility rule base governing composition and decomposition of messages, encryption and decryption with known keys, as well as memorization and use of eavesdropped information.

In section 2 we provide an overview of the most influential model checking approaches. All of them use the general *Dolev and Yao intruder model* [1], but the intruder's goal is restricted in finding out a message that is meant to be secret or in generating messages that impersonate some protocol participant. *Failures of secrecy* or *authentication* reveal a previously unknown attack on the analyzed protocol.

However, *security guarantees cannot always be expressed as absence of secrecy or authentication failure*. A typical case is the well-known family of *replay attacks*, where the intruder aims to the playback of previously recorded messages in an attempt to sabotage an ongoing protocol session: in [2] the authors show that failures of information exchange timeliness that make possible message replays do not always manifest themselves as secrecy failures. Hence, replay attacks are basically analyzed [3] with special-purpose modal logics, like the BAN logic (named after the initials of its inventors [4]). Another complication is that recent studies ([5]) concluded in that *authentication is a protocol dependent notion and there is not a unique definition of authentication that all secure protocols satisfy*.

Sections 3 and 4 introduce a philosophically different approach in designing the intruder. *We also adopt the assumptions of the Dolev - Yao intruder*, but instead of specifying its behavior with a set of rules governing deducibility of messages, *we attempt to combine multiple attack tactics based on a careful analysis of how they proceed*. Attack tactics are formalized and are then combined into a single Dolev - Yao intruder within the SPIN model-checking environment ([6], [7]). Four types of attack tactics have been implemented so far, namely: (i) Replay and integrity violation attacks, (ii) Type-flaw attacks, (iii) Impersonation attacks, (iv) Parallel session attacks.

Although we cannot claim that our approach covers all possible attack tactics, *we do not exclude known attacks that are not reflected as failures of secrecy or authentication*. The developed Dolev - Yao intruder constitutes a supplemental model-checking mean, used as *an open-ended base for implementing more specialized attack tactics*. This makes possible to reveal attacks, which cannot be detected by existing security model checkers, like for example attacks that subvert non-repudiation [8], fairness, accountability, abuse-freeness [9] or other *e-commerce security guarantees*.

An interesting aspect is the comparatively smaller state spaces that make possible *analyses to not be restricted to small systems running the protocol of interest*. This allows application of the proposed intruder model to larger and more complex systems, thus opening new potentialities in revealing for example multi-protocol attacks [10] on cryptographic protocols that are executed in the same environment.

With the described approach we discovered an integrity violation attack on the PayWord micro payment protocol [11]. The obtained results are shown in section 6.

2 Related work

One of the first systems that used the Dolev - Yao intruder and the secrecy failure approach was the Interrogator tool [12]. Given a final state in which the intruder knows some word, which should be secret, the Interrogator tries all possible ways of

constructing a path by which that state can be reached. If it finds a path, then it has identified a security flaw.

Finite state analysis of cryptographic protocols has been developed in a range of published works, which implement the secrecy or authentication failure approach within the frame of specialized security analysis tools like BRUTUS [13] or within general purpose model checkers like Murφ [14] and FDR [15].

The most detailed description of a Dolev - Yao intruder [16] is given for the so-called "Lazy Spy". The "Lazy Spy" was initially expressed [17] in the traces model of CSP/FDR and was later integrated into Casper [18], a front-end for semi-automated CSP description of security protocols. Casper works based on a custom-made set of rules governing deducibility of messages through encryption and uses a lazy exploration strategy, which examines the subset of intruder states reachable by the protocol rules. NRL Protocol Analyzer [19] is another well-known tool with a similar Dolev - Yao intruder.

In [20] the authors provide a thorough review of the most important state space analysis contributions until 1999. A more recent contribution for the model checking of secrecy and authentication is the so-called "lazy intruder" [21] for the on-the-fly model checker of the AVISPA security toolset [22]. The "lazy intruder" avoids an explicit enumeration of the possible messages the intruder can generate, by storing and manipulating constraints about what must be generated. The resulted symbolic representation is evaluated in a demand-driven way and this approach reduces the search tree without excluding any attacks.

3 The Intruder Model

We adopt the pessimistic assumption that the intruder has absolute control over the used communication network, as well as the basic Dolev - Yao assumptions mentioned in section 1 regarding his abilities. More precisely, the intruder *eavesdrops* or *intercepts* messages and analyzes them if he possesses the keys required for decryption. Also, the intruder *can generate messages* from his knowledge and can send them to any protocol participant. The new messages are created from already known messages by applying one or more of four (4) basic operations: *encryption*, *decryption*, *concatenation* and *projection*.

Any attempt to enumerate all meaningful messages that the intruder can send will inevitably lead to an enormous branching of the resulting state space. The model checking approaches of section 2 attempt to preserve the generality of the intruder model while applying specialized techniques to overcome the foresaid problem. However, they are only applicable to a small system running the protocol of interest. If no attack is found, there is still an open possibility for an attack upon some larger system (*absence of model-checking completeness* [23]).

We aim in a less general but complementary approach for the generation of new messages based on an open-ended base of predefined attack tactics. The structure and the number of all possible fake messages are restricted by the patterns and the number of initial messages of the available attack tactics. The intruder model can be thought as two concurrent processes, where the first aims to eavesdrop/intercept exchanged

messages and the second performs a non-deterministically selected attack tactic against the ongoing protocol session(s) (Figure 1).

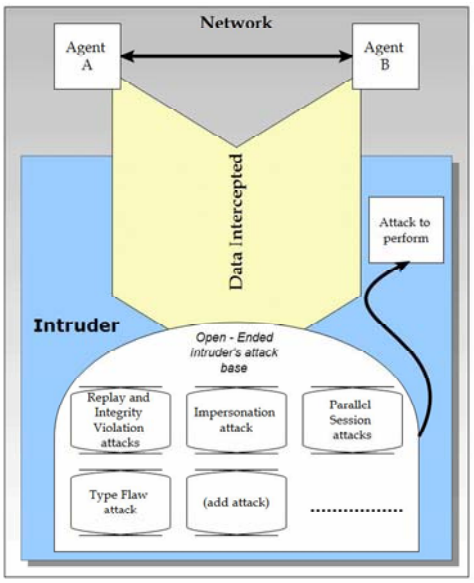

Figure 1. The intruder process

Upon reception of a fake message, by some victim, the performed attack step succeeds and the subsequent execution trace is possible to reach an invalid end state or a correctness assertion violation. If the victim does not accept the sent fake message, falls into a *fail-stop state*, where the "recipient" does not continue with the ongoing protocol execution. Protocol correctness, whether it is expressed as reachability of an invalid end state or an assertion check is thus not restricted to secrecy or authentication guarantees.

An *atomic message* may come from one of the sets:

- *Keys*, with members that represent the keys used to encrypt messages, such that every key $k \in Keys$ has an inverse $k^{-1} \in Keys$. For symmetric cryptography the decryption key is the same as the encryption key, i.e. $k = k^{-1}$.
- *Agents*, with members that represent the names of the *honest protocol participants*.
- *Nonces*, which is an infinite set of randomly generated numbers. Members of *Nonces* are used as timestamps that is, any message containing one of them can be assumed having been generated after the nonce itself was generated.
- *Data*, with members that represent the plaintext strings exchanged between the protocol's participants.

We denote by I the *intruder* ($I \notin Agents$). Also, we define the binary relation,

$$is\_key\_of = \{(k, id): k \in Keys, id \in Agents \cup \{I\}, \text{"}key\ k\ is\ used\ by\ the\ participant\ id\text{"}\}$$

such that $|is\_key\_of\ (k)| = 1$ in the case of public key cryptography or $|is\_key\_of\ (k)| = 2$ in the case of symmetric cryptography.

The set *Msgs* of *exchanged messages* is defined inductively over the *disjoint union*

$$AMsgs = Keys \cup Agents \cup \{I\} \cup Nonces \cup Data$$

that represents the set of atomic messages ($Set_i \cap Set_j = \varnothing$ for any two Set_i, Set_j of the unified sets). More precisely:

- If $\alpha \in AMsgs$ then $\alpha \in Msgs$.
- If $msg_x \in Msgs$ and $msg_y \in Msgs$ then $msg_x \cdot msg_y \in Msgs$, where $\cdot$ represents message concatenation.
- If $msg \in Msgs$ and $k \in Keys$ then $\{msg\}_k \in Msgs$.

Each $\mathsf{ag} \in Agents$ may attempt to execute the protocol for a bounded number of times say $\#Ses_{\mathsf{ag}}$ and each such attempt is a separate *protocol session noSes*, such that $1 \le noSes \le \#Ses_{\mathsf{ag}}$. In a protocol session, ag plays either the role of the *initiator* or the *responder*.

We denote by $sent_n^{\mathsf{ag}_{noSes}}$ the finite-length concatenation sequence of messages sent by $\mathsf{ag} \in Agents$ in the course of session *noSes*:

$$sent_n^{\mathsf{ag}_{noSes}} = (sent_{n-1}^{\mathsf{ag}_{noSes}} \cdot msg_n)$$

with the first term equal to the *null sequence* that is, $sent_0^{\mathsf{ag}_{noSes}} = ()$. The sequence $sent_n^{\mathsf{ag}_{noSes}}$ represents participant's ag *history* for session *noSes*, after having sent msg_n. We denote by $rcvd_n^{\mathsf{ag}_{noSes}}$ the finite-length concatenation sequence of messages received by ag in the course of session *noSes*. In a given time instant the acquired *participant's knowledge* for the ongoing protocol execution is given as

$$\mathsf{ag}_{\text{knowledge}} = \bigcup_{\mathsf{ag}_j} \{rcvd_{\max(i)}^{ag_j}\} \cup \mathsf{ag}_{in\_knowledge},$$

for all $1 \le j \le \#Ses_{\mathsf{ag}}$, where $\mathsf{ag}_{in\_knowledge}$ represents the *initial knowledge base* of ag (keys, agent identities and so on) and $i > 0$ represent the terms of the received message concatenation sequences. A protocol session for a honest participant $\mathsf{ag} \in Agents$ is defined formally as a 5-tuple $\langle \mathsf{ag}, j, \mathsf{ag}_{knowledge}, \mathsf{ag}_{history}^{j}, P \rangle$, where $1 \le j \le \#Ses_{\mathsf{ag}}$ and P is a *process description* given as a sequence of *actions* to be performed. We consider the actions **send** and **receive** for sending and receiving messages to/from other protocol's participants.

The assumptions mentioned in section 1 for the general Dolev - Yao intruder imply that in a given time instant the acquired *intruder's knowledge* for the ongoing protocol execution is given as

$$I_{knowledge} = \bigcup_{\mathsf{ag}_j} \{sent_{\max(i)}^{ag_j}\} \cup I_{in\_knowledge},$$

for all $1 \le j \le \#Ses_{\mathsf{ag}}$, $\mathsf{ag} \in Agents \cup \{I\}$, where $I_{in\_knowledge}$ represents the initial intruder's knowledge base and $i \ge 1$ represent the terms of the eavesdropped message concatenation sequences.

The *protocol model* is given as the asynchronous composition of the models for each protocol session, including the intruder, whose behavior depends on the defined *attack tactics*. Attack tactics are non-deterministically selected and are then executed within a single thread of control. Each possible *execution* of the model corresponds to a finite alternating sequence of *global states* and actions:

$$\tau = s_0\, \alpha_1\, s_1\, \alpha_2 \ldots s_n, \text{ for some } n \in N$$

such that $s_{i-1} \xrightarrow{a_i} s_i$ for $0 < i \leq n$ and for the transition relation $\rightarrow$ defined as

$$\rightarrow \subseteq S \times PS \times A \times Msgs \times S$$

where S is the set of global states, PS is the set of protocol sessions and A is the set of action names.

4 Attack Tactics

We formalized and subsequently implemented a series of basic attack tactics. First, we present the elementary tactics that are also used in forming more complex attack scenarios. The implemented attack tactics are the ones that are most often reported in related bibliography.

4.1 Message INterCePTion (INCPT)

Message interception takes place after the occurrence of some action **send** (ag, *v*, *msg*)[1], for some ag, $v \in Agents$ and some $msg \in Msgs$, if there is no **receive** (*v*, *u*, *msg*)[2] with $u \in \{ag, I\}$ in the suffix execution trace. When intercepting an encrypted message $\{msg\}_k$ there is no **receive** (*v*, *u*, $\{msg\}_k$) action in the suffix execution trace.

4.2 Replay attack tactics

Replay attacks take place when the intruder redirects eavesdropped or altered messages within one (or possibly more) interleaved protocol session(s). We adopt the replay attack classification of [5] and we formalize the following replay attack tactics (Figure 3).

REFlections (R-REF):

In a *reflection attack* the intruder resends an altered version of a previously sent message back to its sender. *Run-internal reflections* are performed within the same protocol session. *Interleaving reflections* use contemporaneous protocol sessions and *classic reflections* use messages obtained from already finished protocol sessions.

The R-REF attack takes place anytime after the occurrence of some action **send** (*v*, ag, *msg*), with *msg* representing any non-encrypted $msg \in Msgs$ or after the occurrence of some action **send** (*v*, ag, $\{msg\}_k$) such that $I \notin is\_key\_of(k) \wedge k^{-1} \in I_{knowledge}$.

The foresaid actions result in a global state where either

$exists(msg, sent^{v_j}_{\max(i)})$[3] = true or respectively $exists(\{msg\}_k, sent^{v_j}_{\max(i)})$ = true

[1] The action whereby ag sends *msg* to *v*

[2] The action whereby *v* receives *msg* from *u*

[3] Boolean predicate indicating if the string *str* appears in message $msg \in Msgs$

for some $1 \le j \le \#Ses_v$, with $i \ge 1$ representing the terms of an eavesdropped message concatenation sequence.

In the performed reflection attack the intruder alters *msg* based on $I_{knowledge}$ and uses the altered $msg' \in Msgs$ in an action **send** (I, v, msg') or **send** (I, v, $\{msg'\}_{k'}$) for some $k' \in I_{knowledge}$ such that $v \in is\_key\_of\,(k')$.

The R-REF attack succeeds only when v performs the action **receive** (v, I, msg') or respectively the action **receive** (v, I, $\{msg'\}_{k'}$) with the following potential outcomes:

- run-internal reflection

  $exists(msg',\ rcvd^{v_j}_{\max(i)}) = \texttt{true}$ or $exists(\{msg'\}_{k'},\ rcvd^{v_j}_{\max(i)}) = \texttt{true}$

- classic or interleaving reflection

  $\exists\, j' \neq j$: $exists(msg',\ rcvd^{v_{j'}}_{\max(i)}) = \texttt{true}$ or $exists(\{msg'\}_{k'},\ rcvd^{v_{j'}}_{\max(i)}) = \texttt{true}$

DEFlections (R-DEF):

In a *deflection attack* the intruder redirects a possibly altered sent message to some participant that is neither the message's recipient nor the sender. *Run-internal deflections* are performed within the same protocol session. *Interleaving deflections* use contemporaneous protocol sessions and *classic reflections* use messages obtained from already finished protocol sessions.

STraight Replays (R-STR):

In a *straight replay attack* the intruder resends a previously sent message to its intended destination. If the eavesdropped message is replaced by an altered version, this attack is also known as *INTegrity Violation attack* (INTV).

Depending on whether this attack is performed within the same session or contemporaneous or non-interleaved sessions, straight replays are also characterized either as run-internal, interleaving or classic replays.

4.3 Type flaw attack tactics (TFLAWS)

A *type flaw attack* arises when the recipient of a message accepts that message as valid, but imposes a different interpretation on the bit sequence than the protocol participant who created it. Type flaw attacks follow the action sequences of the replay attack tactics and may be optionally combined with a message interception (INCPT), in order to prevent reception of intercepted message by its recipient such as to perform a type flaw based message replay.

I triggers a type flaw attack possibly after having altered an eavesdropped $msg \in Msgs$ based on $I_{knowledge}$, thus resulting in some $msg' \in Msgs$. The subsequent action performed by I is either **send** (I, v, msg') or **send** (I, v, $\{msg'\}_{k'}$) for some $k' \in I_{knowledge}$ such that $v \in is\_key\_of\,(k')$.

This attack tactic succeeds if in the global state after the occurrence of the action **receive** (v, I, msg') or respectively **receive** (v, I, $\{msg'\}_{k'}$) there is some atomic message *amsg*, such that

$$exists(amsg,\ rcvd^{v_j}_{\max(i)}) = \texttt{true},\ 1 \le j \le \#Ses_v$$

with $i \ge 1$ representing the terms of $rcvd^{v_j}_n$ and for two sets Set_i and Set_j from the "disjoint" union *Amsgs*,

$$amsg \in Set_i \cap Set_j$$

The described insecure global state expresses the fact that it is possible for an atomic message that was originally intended to have one type (e.g. nonce) to be interpreted as having another type (e.g. key or data). However, this possibility occurs only when both types are represented as bit sequences of the same length, so that when the intruder positions an atomic message in place of a type flawed one, the recipient is fooled into accepting the used atomic message as the one expected according to the owned process description (*P*).

We note that type flaw attacks [24] may not lead to a direct security compromise, since it is possible that the plaintext bit string of the atomic message used by *I* to be unknown to him (the secrecy is preserved). However, if for example a nonce is used as a key, this is not a good key, because the main concern in generating nonces is to be unique in a protocol session, as opposed to keys that basically have to be non-predictable. Type flaw attacks may result in failures of security properties beyond the typical secrecy and authentication properties, like for example anonymity and non-repudiation [25].

4.4 Simple IMPersonation attack (IMP)

An *insecure state* (precondition) for the performance of an IMP attack is any state where *I* can read the contents of a protocol message sent by some $\texttt{ag} \in Agents$, who acts as initiator of a new protocol session:

$$\{\exists\ sent_1^{\texttt{ag}_{noSes}} \in I_{knowledge},\ \texttt{ag} \in Agents,\ 1 \le noSes \le \#Ses_{ag}:$$

$$\{\ sent_1^{\texttt{ag}_{noSes}} = msg \text{ for some non-encrypted } msg \in Msgs\}$$

$$\vee\ \{\ sent_1^{\texttt{ag}_{noSes}} = \{msg\}_k :\ is\_key\_of\ (k) = I \vee (is\_key\_of\ (k) \ne I \wedge k^{-1} \in I_{knowledge})\}\}$$

The IMP attack tactic takes place when the intruder performs the following three subsequent actions against some victim $v \in Agents$, such that $v \notin is\_key\_of\ (k)$ and $v \ne \texttt{ag}$:

$$\textbf{send}\ (I, v, msg'),\ \textbf{receive}\ (I, v,\ sent_1^{v_{newSes}}),\ \textbf{send}\ (I, \texttt{ag},\ sent_1^{v_{newSes}})$$

where $msg' = sent_1^{\texttt{ag}_{noSes}}$, when the latter is a non-encrypted message or otherwise $msg' = \{msg\}_{k'}$, with $k' \in I_{knowledge}$ and $v \in is\_key\_of\ (k')$. Also, v_{newSes} is a unique session identifier for session *newSes*, in which victim *v acts as responder* and the boolean predicate $exists(v,\ sent_1^{v_{newSes}})$ is `false`. If the last mentioned predicate is `true`, `ag` realizes that the responder in session $\texttt{ag}_{noSes}$ is not the one selected and subsequently aborts the corrupted protocol session.

4.5 Parallel session attack tactics (PARSES)

Parallel session attacks take place by subsequent interleaving replays among contemporaneous protocol sessions, in which the intruder manipulates protocol participants in multiple roles (initiator or responder), in order to subvert the protocol's goals.

The intruder can under special conditions use the cryptographic protocol dialogs:

- As an *oracle* that is, to foretell the contents of otherwise perfectly encrypted messages (refer to the oracle session attack shown in [26]).
- To impersonate a protocol participant (e.g. the BAN-Yahalom attack in [5]).

or possibly to subvert properties beyond secrecy and authentication.

In a parallel session attack the execution sequence τ includes a series of action cycles that open with some action **send** (`ag`, *v*, *msg*) or **send** (`ag`, *v*, $\{msg\}_k$) and this results in

$exists(msg,\ sent^{ag_j}_{\max(i)}) = \texttt{true}$ or respectively $exists(\{msg\}_k,\ sent^{ag_j}_{\max(i)}) = \texttt{true}$

I either opens a new protocol session v'_{newSes} or responds to an already opened session say v'_m (with $v' \in Agents$ including `ag` and *v*), for which the last action of the process description *P* is not included in the prefix execution sequence of τ. The attack is performed possibly after having altered the eavesdropped $msg \in Msgs$ (based on $I_{knowledge}$), thus resulting in sending some $msg' \in Msgs$ by **send** (*I*, *v*′, *msg*′) or **send** (*I*, *v*′, $\{msg'\}_{k'}$) for some $k' \in I_{knowledge}$ such that $v' \in is\_key\_of(k')$. The interleaving replay succeeds if the action cycle ends with a *receive* action by *v*′, yielding a global state such that

$exists(msg',\ rcvd^{v'_m}_{\max(i)}) = \texttt{true}$ or respectively $exists(\{msg'\}_{k'},\ rcvd^{v'_m}_{\max(i)}) = \texttt{true}$

with max(*i*) = 1, if *m* represents a new protocol session (*newSes*).

A number of successive interleaving replays may end up in a *fail-stop global state* or in either an invalid end state or violation of a *protocol correctness assertion*. The latter possibility reveals a previously unknown parallel session attack.

5 The PayWord micro payment protocol

We focus on the analysis of the PayWord micro-payment protocol that was first proposed by Rivest and Shamir in [11]. PayWord is a credit based off-line protocol implemented by the use of *hash chains* that are called *chains of PayWords*. In our work we will assume the use of the MD5 hash function [27] denoted by `w(i)`. Three participants are involved in a protocol session: the *Customer*, the *Broker* and the *Vendor*. The Customer (*C*) establishes an account with the Broker (*B*) who issues a certificate containing customer's information and *B*'s name. This certificate will authorize *C* to construct PayWord chains validating himself to the some Vendor (*V*). The basic steps of PayWord micro-payments are shown in Figure 2.

Upon reception of the foresaid certificate (`certC`), *C* computes the PayWord chain *w* in reverse order based on a randomly chosen term. Then, he signs the so-

called commitment (*M*) of the PayWord protocol which consists of the calculated first term of the chain (`w(0)`) along with the required customer information; *M* is sent to *V*. In every single payment, a chain term of type, `P:(w(i),i)` is sent to *V* until the last payment, `P:(w(I),I)`. We consider the (attacked) variable-size payment scenario, where the value of each payment varies between `1` and `n`. *V* verifies the payments `P`, by applying the hash function `w` to the last valid payment *v* times, where *v* is the value of the requested payment (`w(i-v)`). At the end of the day, *V* reports to *B* the last (highest-indexed) payment (`w(I),I`) - where `I=max(i)` - received from *C* within the current day, together with the owned *C*'s commitment.

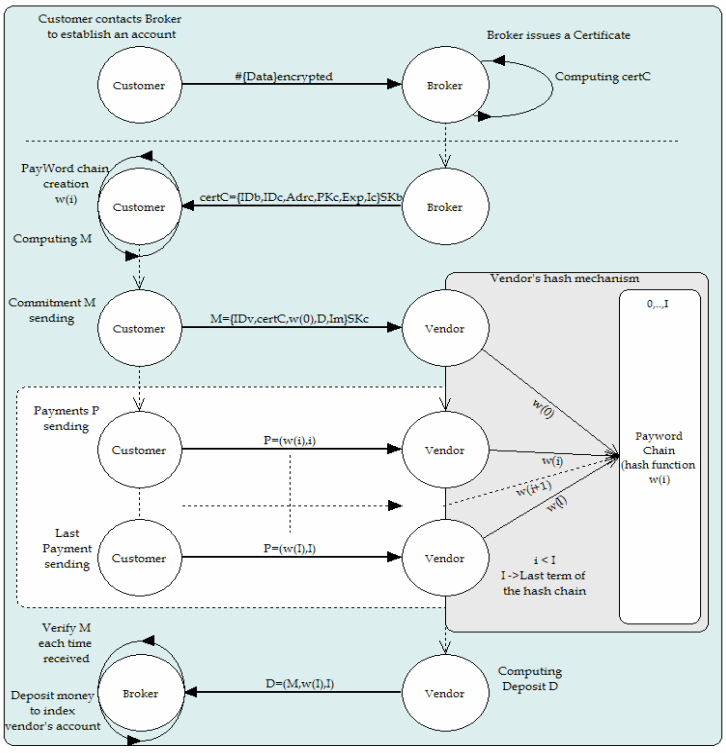

Figure 2. The PayWord micro payment protocol

Table 1. Glossary of the PayWord protocol notation

`IDc`	Customer ID	`Addrc`	Customer address
`IDb`	Broker ID	`certC`	Customer certificate
`IDv`	Vendor ID	`Exp`	Certificate expiration timestamp
`SKb`	Broker's key	`Ic`	Customer's information
`PKc`	Customer's public key	`Im`	Vendor's information
`SKc`	Customer's secret key	`D`	Date

While the use of the hash chain ensures reduced computational requirements for *V*, the attack found on the protocol is based on *V*'s mechanism, when accepting an al-

tered “hashed” message. Provided the intruder’s ability to perform hash function calculations by MD5, the detected attack takes place when the intruder intercepts and alters a variable-size payment request.

6 Verification results

This section provides simulation and verification results obtained within the SPIN model checking environment for the developed PayWord model, when combined with the described intruder. The simulation output is shown by the automatically generated Message Sequence Chart.

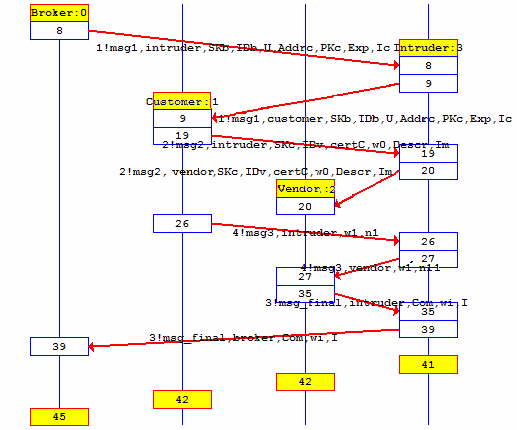

Figure 3. INTV attack of a variable-size payment (P): *V* accepts an altered message

```
pan: invalid end state (at depth 26)
pan: wrote pan_in.trail
(Spin Version 4.2.6 -- 27 October 2005)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim         - (not selected)
        assertion violations - (disabled by -A flag)
        cycle checks        - (disabled by -DSAFETY)
        invalid end states  +

State-vector 112 byte, depth reached 27, errors: 1
    23 states, stored
     0 states, matched
    23 transitions (= stored+matched)
     0 atomic steps
hash conflicts: 0 (resolved)

Stats on memory usage (in Megabytes):
0.003   equivalent memory usage for states (stored*(State-vector + overhead))
0.293   actual memory usage for states (unsuccessful compression: 10603.04%)
        State-vector as stored = 12716 byte + 8 byte overhead
2.097   memory used for hash table (-w19)
0.320   memory used for DFS stack (-m10000)
0.144   other (proc and chan stacks)
0.089   memory lost to fragmentation
2.622   total actual memory usage
```

Figure 4. Verification output

Figure 3 shows the detected INTV attack. In state 19 *C* sends a commitment (*M*), which is not affected by the intruder and continues with the first variable-size payment attempt (*P*). In state 27 the intruder alters message `(w1,n1)` thus resulting in the fake message `(w1',n1-1)`, which is eventually accepted by *V*. Finally, *V* dispatches message *D* (deposit) and the protocol session ends with a successful INTV attack (encoded as an invalid end state).

Figure 4 shows the obtained verification output that revealed the described attack scenario. The performed state space search reports an error and generates a counter-example reflecting a feasible path to the defined invalid end state. By the use of the error trail simulation feature of SPIN we roll back the protocol execution and identify the detected flaw.

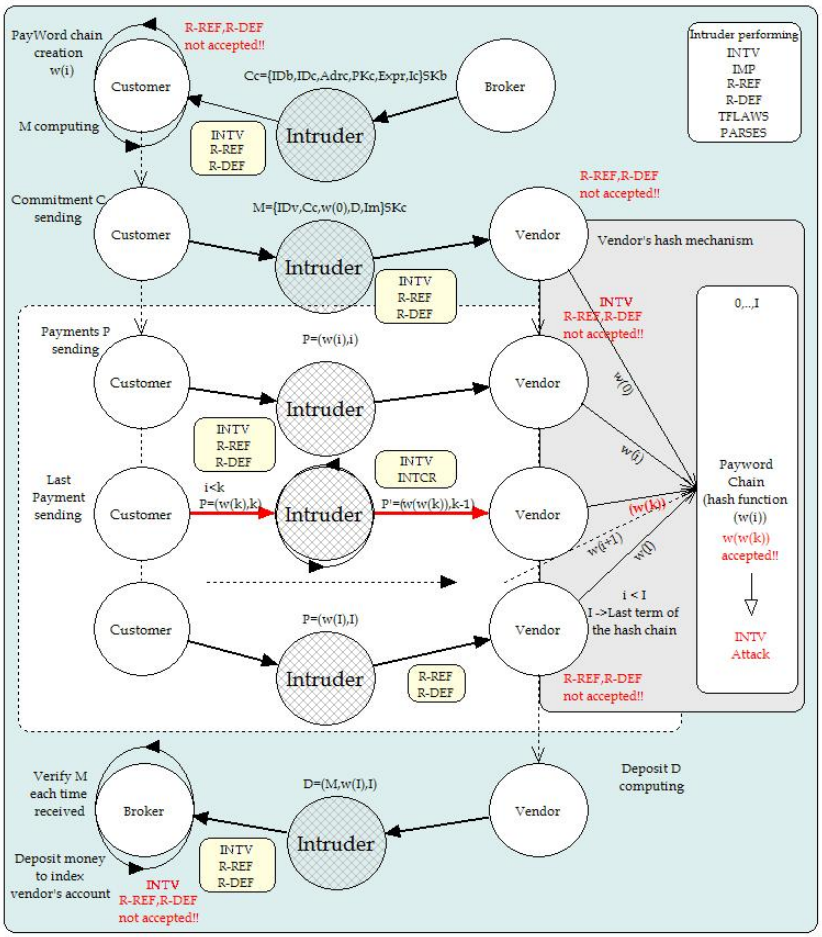

Figure 5. Attack tactics on the PayWord micro payment protocol

Figure 5 summarizes the attack tactics attempted by the described intruder and the participants' responses in all protocol steps. Failed attack scenarios are noted as "not

accepted!!" and result in fail-stop model states. The detected INTV attack is shown within the frame in the right-hand side that represents *V*'s hash mechanism.

7 Conclusion

This work introduces an open-ended Dolev - Yao intruder that combines elementary and more complex attack tactics in an attempt to subvert security protocol guarantees. We provided a formalized description of the most often reported attack tactics, which were implemented within the SPIN model-checking environment. The obtained intruder was applied to a range of electronic payment protocols and revealed an integrity violation attack on the PayWord micro-payment protocol.

Although the proposed model is bound to the absence of model checking completeness - as all published intruder models - it constitutes a supplemental model-checking mean, capable to reveal violations of protocol correctness properties, beyond those checked by existing security model checkers.

The proposed intruder is open to extensions aiming to integrate more specialized attack tactics that may subvert e-commerce security guarantees like non-repudiation, fairness, accountability, abuse-freeness and so on.

References

1. Dolev, D. and Yao, A., On the security of public-key protocols, IEEE Transactions on Information Theory 2/29, 1983, pp. 198-208.
2. Woo, T. Y. C. and Lam, S. S., A semantic model for authentication protocols, In Proc. of the IEEE Symposium on Research in Security and Privacy, 1993.
3. Meadows, C. A., Formal verification of cryptographic protocols: A survey, Advances in Cryptology International Conference on the Theory and Application of Cryptology (Asiacrypt '94), LNCS 917 Springer-Verlag, 1995, pp. 133-150.
4. Burrows, M., Abadi, M. and Needham, R., A logic of authentication, ACM Transaction on Computer Systems 8/1, 1990, pp. 18-36.
5. Syverson, P. and Cervesato, I., The logic of authentication protocols, In Proc. of the 1st International School on Foundations of Security Analysis and Design (FOSAD 2000), LNCS 2171, Springer-Verlag, 2001, pp. 63-137.
6. The SPIN model checker official website, available at http://spinroot.com/
7. Holzmann, G. J., Design and Validation of Computer Protocols, Prentice-Hall, 1991.
8. Kremer, S., Markowitch, O. and Zhou, J., An intensive survey of fair non-repudiation protocols, Computer Communications, 25/17, 2002, pp. 1606-1621.
9. Shmatikov, V. and Mitchell, J. C., Finite-state analysis of two contract signing protocols, Theoretical Computer Science, 283, 2002, pp. 419-450.
10. Cremers, C. J. F., Feasibility of multi-protocol attacks, In Proc. of the First International Conference on Availability, Reliability and Security, IEEE Computer Society Press, 2006.
11. Rivest, R. L. and Shamir, A., Payword and Micromint: Two simple micropayment schemes, In Proc. of the Fourth International Workshop on Security Protocols, LNCS 1189 Springer-Verlag, 1996, pp. 69-87.

12. Millen, J. K., Clark, S. C. and Freedman, S. B., The Interrogator: Protocol Security Analysis, IEEE Transactions on Software Engineering 13/2, 1987.
13. Clarke, E. M., Jha, S. and Marrero, W., Verifying security protocols with Brutus, ACM Transactions on Software Engineering and Methodology 9/4, 2000, pp. 443-487.
14. Mitchell, J. C., Mitchell, M. and Stern, U., Automated analysis of cryptographic protocols using Murφ, In Proc. of the IEEE Symposium on Research in Security and Privacy, IEEE Computer Society, 1997, pp. 141-153.
15. Roscoe, A. W., Modeling and verifying key-exchange protocols using CSP and FDR, In Proc. of the 8th IEEE Computer Security Foundations Workshop ,IEEE Computer Society, 1995, pp. 98-107.
16. Roscoe, A. W., The theory and practice of concurrency, Prentice Hall, 1997.
17. Roscoe, A. W. and Goldsmith, M., The perfect spy for model-checking cryptoprotocols, In Proc. of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997.
18. Lowe, G., Casper: a compiler for the analysis of security protocols, In Proc. of the IEEE Computer Security Foundations Workshop, IEEE Computer Society, 1997, pp. 18-30.
19. Meadows, C., Kemmerer, R. and Millen, J., Three systems for cryptographic protocol analysis, Journal of Cryptology 7/2, 1994, pp.79-130.
20. Gritzalis, S., Spinellis, D. and Georgiadis, P., Security protocols over open networks and distributed systems: formal methods for their analysis, design, and verification, Computer Communications 22, 1999, pp.697-709.
21. Basin, D., Modersheim, S. and Vigano, L., OFMC: A Symbolic Model-Checker for Security Protocols, International Journal of Information Security, 2004.
22. AVISPA: Automated validation of internet security protocols and applications, 2003, FET Open Project IST-2001-39252. http://www.avispa-project.org
23. Lowe, G., Towards a completeness result for model-checking of Security Protocols, In Proc. of the 11th Computer Security Foundations Workshop. IEEE Computer Society Press, 1998.
24. Clark, J. and Jacob, J., A survey of authentication protocol literature: version 1.0, Technical Report, University of York, 1997.
25. Heather, J., Lowe, G. and Schneider, S., How to prevent type flaw attacks on security protocols, In Proc. of the 13th IEEE Computer Security Foundations Workshop, IEEE Computer Society, 2000, pp, 255-268.
26. Carlsen, U., Cryptographic protocol flaws – Know your enemy, In Proc. of the 7th IEEE Computer Security Foundations Workshop, IEEE Computer Society, 1994, pp. 192-200.
27. Rivest, R. L., The MD5 Message-Digest Algorithm, Internet informational RFC 1321, 1992.