The Use of Dynamic Sliding Window with IPSec

Mishall H. Awaad
Department of Computer Science, Education College, Thi-Qar University
mishallhammed@yahoo.com

Abstract:
IPSec (IP security) technology is used to protect the local computer networks and the internet from attacks. The most illustrious attacks against networks are the replay attacks where IPSec technology introduces a solution called sliding window through which the packets will be protected. This window involves packets each has a distinct sequence number. This sequence number prevents replay attacks. But, there is a problem related to this solution: when the packet is received late it will be discarded whether it was good or forged. This usually causes a big losses in the good packets. In this paper, the researcher discusses the use of the dynamic sliding window which is a window added to the original one. This window keeps most of the good packets that have been discarded (reorder packets) and which comes before the original window. The dynamic window moves automatically with the original window. This dynamic window makes the modified protocol introduce the best solution which is more efficient than the original protocol since it keeps a large set of good reorder packets. It has been found that the dynamic window saves more than %85 of the reorder packets.

Keywords: IPSec, original window, dynamic window, reorder packets.

إستخدام نافذة الإنزلاق الديناميكية مع تقنية IPSec

مشعل حمد عواد
جامعة ذي قار –كلية التربية للعلوم الصرفة – قسم علوم الحاسبات
mishallhammed@yahoo.com

الملخص:
إن تقنية IPSec مستخدمة لحماية شبكات الحاسوب المحلية والإنترنت من الهجمات، ومن أكثر الهجمات البارزة ضد الشبكات هي هجمة إعادة الإرسال Replay Attack ، حيث أن IPSec قدم حلا لحماية الشبكات من هذه الهجمات من خلال ما يسمى نافذة الإنزلاق Sliding Window، هذه النافذة تقومبحماية الحزم بإستخدام Sequence Number مميز لكل حزمة packet. لكن مع هذا الحل تحصل مشكلة . عندما تصل حزم متأخرة قبل نافذة الإنزلاقSliding Windowفإنه سوف يتم إلغائها سواء كانت حزم مزيفة أو صحيحة مما يسبب خسارة كبيرة في الحزم الصحيحة. في هذا البحثنحن نناقش إستخدام نافذة الإنزلاق الديناميكية Dynamic Sliding Windowحيث يتم إضافتها مع النافذة الأصلية، هذه النافذة الديناميكية ممكن أن تحافظ على الكثير من الحزم الصحيحة الملغاة(الحزم المعادة الطلب) والتي تكون قبل النافذة الأصلية، إن هذه النافذة الديناميكية تتحرك بشكل ألي مع النافذة الأصلية مما يجعل هذا البروتوكول المعدل حلا أفضل وأكثر كفاءة وأفضل من البروتوكول الأصلي من خلال الحفاظ على نسبة كبيرة جدا من الحزم الصحيحة الملغاة حيث يتم الحفاظ على أكثر من 85% من الحزم المعادة الطلب.
الكلمات الدليلية: التقنية الأمنية IPSec، النافذة الأصلية، النافذة الديناميكية، الحزم المعادة الطلب.

1. Introduction

Information on the Internet is carried using theInternet Protocol (IP), which does not inherently provideprivacy or other security. As a result, IP Security (IPSec)was developed to integrate security into IP [1].

IPSec is a suite of protocols that adds security to communications at the IP level. This suite of protocols is becoming more and more important as it is included as a mandatory security mechanism in IPv6. IPSec is mainly composed of two protocols; Authentication Header (AH) and Encapsulating Security Payload (ESP). The former allows authentication of each IP datagram's selected header fields or – depending on the operational mode that has been selected – of the entire IP datagram. The latter allows encryption – and optionally authentication – of the entire IP datagramor of the IP payload, depending on the operational mode that has been selected, namely the transport and the tunnel modes [2].

The concept of Security Association (SA) is fundamental to IPSec. A Security Association is a simplex "connection" that supplies security services tothe traffic carried by it. To secure typical bi-directionalcommunication between two peers, two SAs (one in eachdirection) are required. Security services are afforded toa SA by the use of AH, or ESP, but not both. Securityassociation establishment can be performed through aprotocol named Internet Key Exchange (IKE)[2][3].

As part of establishing a security association between two computers, the IP layers in the two computers are provided with shared secrets and keys.Once a security association is established from a source computer p to a destination computer q, the IP layer in p can send IPmessages over the established association to the IP layer in q. Allmessages sent over this established association are augmented withspecial headers. These special headers are generated by the IP layerin p and checked by the IP layer in q, using the shared secrets andkeys between p and q. If the checks performed by q for somereceived message m are satisfactory, then q concludes that m wasindeed sent by p, that m was not modified after it was sent by p, andthat m is not a second copy of a message that was received earlier byq; i.e. m is not a replayed message. To check whether a receivedmessage is a replayed message, IPSec uses an anti-replay windowprotocol [4].The anti-replay sliding window is used in IPSec to resist the replay attacks. Sequence number field is used as a freshness identifier to distinguish the new packet from the old one [5].

If the sequence number of the received message falls inside the window, the destination can determine whether the message is are played message or not by checking the information kept in the window. If the sequence number of the received message is larger than the number represented by the right edge of the window, the message is regarded as a fresh message and the window is shifted to the right, making this received sequence number the new right edge of the window. Note that for a message and its sequence number to be accepted, the message also needs to pass integrity check in the destination [6].

The anti-replay window protocol has been shown to be effective in preventing replay attacks. Every replayed message inserted by an adversary is guaranteed to be detected and discarded by the anti-replay window protocol. However, this simple protocol has a potential problem in which severe reorder of messages can cause the protocol to discard a lot of good messages. This problem needs to be solved because message reorder occurs very often in the Internet [6][7].In this paper, we use the abstract protocol notation to represent the protocols (the original window and the dynamic window), this notation is to be clarified in the section below:

2. Abstract Protocol Notation

The protocols in this paper are specified using a version of the Abstract Protocol Notation presented in [4]. We use this notation because it provides a well-defined set of semantics that is suitable for distributed environment. In this notation, each process in a protocol is defined by a set of constants, a set of variables, and a set of actions. For example, in a protocol consisting of two processes p and q, process p can be defined as follows:

processp
const<name of constant> : <type of constant>
…
<name of constant> : <type of constant>
var<name of variable> : <type of variable>
…
<name of variable> : <type of variable>
begin
<action>
[] <action>

[] <action>
end

The constants of process p have fixed values. The variablesof process p can be read and updated by the actions ofprocess p. Comments can be added anywhere in a processdefinition; each comment is placed between the two brackets{and}.

Each <action> of process p is of the form:
<guard> → <statement>
The guard of an action of p is either a boolean expressionover the constants and variables of p or a receive guard of theform:
rcv<message>**from** q.

Executing an action consists of executing the statement ofthis action. Executing the actions (of different processes) in aprotocol proceeds according to the following three rules.First, an action is executed only when its guard is true.Second, the actions in a protocol are executed one at a time.Third, an action whose guard is continuously true iseventually executed [6].
The <statement> of an action of p is a sequence of <skip>,<assignment>, <send>, <selection>, or <iteration> statementsof the following forms:

<skip> : **skip**
<send> : **send** <message>**to** q
<assignment> : <list of variables of p> :=

```
<list of expressions>
<selection> : **if** <boolean expression>→<statement>
…
[] <boolean expression>→<statement>
**fi**
<iteration> : **do** <boolean expression>→<statement>
**od**
```

Note that the <assignment> statement simultaneously can assignnew values to multiple variables. Consider for example thefollowing <assignment> statement

wdw[j], j := **false**, j+1

In this statement, the j-th element of the boolean array wdw isassigned the value false, and the value of variable j is incrementedby one.

3.Motivation

The local and Internet networks show a huge and important expansion in the two last decades in both environments of business and of personal.The local networks and internet might suffer from weak points which might be used by the attackers. The replay attacks against these networks cause many problems. IPSec is one of the most important security techniques in communication. The IPSec is used to secure the network by using the sliding window. Anyway, IPSec has a problem. Packet reordering is a problem in IPSec when using the sliding window, it happens a lot, in other words, it is not a rare phenomenon and it is hard to be settled. This problem occurs as a result of many reasons: router fluttering, local parallelism, etc.

The researcher's motivation is to solve the problem of packet reordering and reducing the loss of packets to the minimum.

4. Replay Attack

Replay attack is one of the most common and themost dangerous attack in industrious network.In all the authentication protocols that may exist attacks, more than 90 percent of attacks are replay attack [8].The appearance of the replay attack led to the use of the sliding window in the IPSec technology. Then, the problem of reorder packets show up, therefore, we will explain this attack in this section. The reply attack occurs when the attacker eavesdropping on the communication between the sender and the receiver. Then, the attacker willcapture the moving packets throughout the communication and sends them later on. This will make the attacker a legitimate user of the network. But the use of the sequence number completely prevents this attack because every packet has a unique sequence number [7].The replay attack may take place in the situation when theadversary has the knowledge of the anonymization algorithm [9].

5. The Anti-Replay Window Protocol

We will illustrate the original anti-replay window in detail. Two processes will be used: p and q. p is a sending process and q is a receiving process. The states of packet-testing will be illustrated in the process q, representing the processes (p and q), and the algorithm related to the window.In the anti-replay window protocol, a process p sends a continuous stream of messages to another process q. The sent messages may be lost or reordered before they are received by q. A message m is said to suffer a reorder of degree w (window size)iff the w-th message sent (by p) after m is received (by q) beforem [4].At any instant, an adversary can insert in the message stream from p to q a copy of any message that was sentearlier by p. Because of the inserted messages, there is apossibility that process q receives and delivers multiplecopies of the same message. To prevent this possibility, thetwo processes p and q are designed such that the followingtwo conditions are satisfied for a given value w.

w-Delivery:

Process q delivers at least one copy of every messagethat is neither lost nor suffered a reorder of degree wor more after it is sent by p.

Discrimination:

Process q delivers at most one copy of every messagesent by p [4] [5][6].

To satisfy these two conditions, p attaches a uniquesequence number to each message before sending themessage to q, and process q maintains a window of wconsecutive sequence numbers. For each sequence number sin the window, q maintains a Boolean variable indicatingwhether or not q has already received the message whosesequence number is s [6].As suggested by Figure1, there are three cases to consider when process q receives a message whose sequence number is s.

1.s≤r and s≥r-w (the sequence numbers in the window where r is maximum right edge in the window):

In this case, q can determine whether it has received this message before (and so it discards this message) or it has not received this message before (and so it delivers this message).

2. s>r (sequence number is larger than right edge for the window):

In this case, q determines that it has not received this message before and delivers the message. Also q slides the window such that s becomes the new right edge of the window. (This means that some sequence numbers near the old left edge are dropped off the window).

3. s<r-w (the sequence number is before the window) :

In this case, q cannot determine whether it has received this message before. To be on the safe side, q assumes that this message has been received before and discards the message[4].

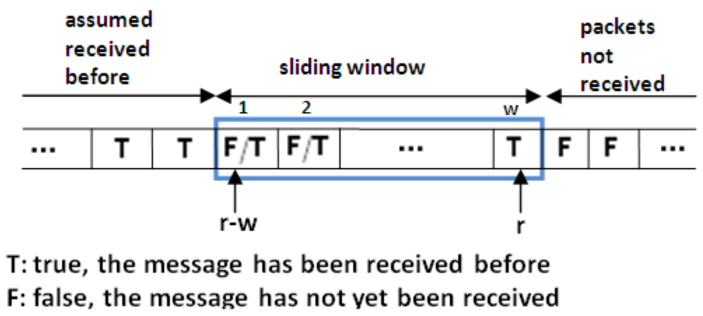
assumed received before
sliding window
packets not received
1
2
w
...
T
T
F/T
F/T
...
T
F
F
...
r-w
r
T: true, the message has been received before
F: false, the message has not yet been received

Figure 1: The anti-replay window

a- Original Algorithm

This algorithm will be illustrated to represent original sliding window through the following points:

```
1- Start (window size =32)
   1.1- Is the packet inside window?Yes: go to 2
   1.2- Is the packet to the right of the window?Yes:go to 5
   1.3- Is the packet to the left of the window? Yes:go to 7
2- Is the packet new? No: go to 7
3- ICV is calculated (where ICV is integrating check value)
4- Is the packet Authenticated?
Yes: go to 6
No: go to 7
5- The window is moved so that the packet's sequence
Number becomes the new right edge of the window
    Go to 8.
6- The corresponding slot in the window is marked true
   Go to 8.
7- The packet is discarded and a message is sent to the IPSec manager.
8- End.
```

b- Representation of Processes

Next, we present the anti-replay window protocol using the Abstract Protocol Notation described in the section 2. Process p can be defined as follows.

```
processp
varsp : integer {sequence number of last msg}
begin
true□ □ sp := sp + 1; send msg(sp) to q
end
```

Process q has the following three variables

```
varwdw : array [1..w] of boolean,{window}
r :integer, {right edge of window}
```

Array wdw is the window, and variable r is the maximum sequence number in the window. For each i, $1 \leq i \leq w$, wdw[i] = true iff process q has already received msg(s), where s = r-w+i. Process q can be defined as follows [4][6].

```
processq
constw : integer {window size, w > 0}
varwdw : array [1..w] of boolean,
{window}
r :integer, {right edge of window}
s :integer, {sequence number}
i, j : integer
begin
rcvmsg(s)from p □
if  r – w < s □ □ r □ □ {s is in window}
```

```
i := s – r + w;
ifwdw[i] □ □ {discard msg}skip
[] □ □ wdw[i] □ □ {deliver msg}
wdw[i] := true
fi
    [] s > r □ □ {s is to the right of window: }
    {delivermsg}
    r, i, j := s, s-r+1, 1;
    doi □ □ w □ □ wdw[j], i, j := wdw[i], i+1, j+1
    od;
    doj< w □ □ wdw[j], j :=false, j+1 od
    fi
    [] s □ □ r – w □ □ {s is to the left of window: }
    {discard msg}
    skip
end
```

6. Problem Statement in the Original Protocol

IPSec technology performs its task well in protecting networks from attacks through the use of sliding window. The basic problem in sliding window used in IPSec technology is discarded reorder packet that are before the window, thereforethis issue may lead to the loss of a lot of packets as they move from the source to the destination. To illustrate how these packets discarded suppose that process q receives msg (s) where s is the larger of r (the right edge in the previous window) in this case, the process q implement the sliding window process, which becomes s is the right edge of the new window. Any packet before the new window (i.e. before r-w) become packet discard, and thus, any packet be left of the new window becomes packet discard (reorder packet become discarded). Sometimes process q receives msg (s) and be s too far from the right edge of the previous window, which leads to the loss of many of the packets. We call this kind of message reorder as long-jump reorder because the distance between the newly received sequence number and the right edge of the old window is very long.

This scenario of long-jump reorder is not uncommon [6]. It can arise when p sends several message over some route to process q, then sends subsequent messages over a shorter route that just became available, to q. Some of the subsequent messages reach q before the earlier messages.When q receives the first subsequent message; q slides its window to the right a long distance. Later, when q receives the earlier messages, it detects that they are to the left of its window and discards them [6].

In this paper we provide the mechanism dynamic sliding window. In the next section we will explain this mechanism in detail. Dynamic window used to support the original window in maintaining discard good packets and thus reduce the packets loss significantly.

7. Dynamic Sliding Window

The proposed mechanism will explain in detail in this section. This mechanism will be used with the original window. We will show how the dynamic sliding window in maintaining the packets that are before the original window (packets that are discarded in the original protocol). Original protocol uses two processes p, q. Process p sends continuous stream of messages to q. Sent messages can be lost or reorder before these messages to be received by the process q. Also modified Protocol uses two processes p, q. But the modified Protocol reduces loss reorder good packets to a large extent. In dynamic sliding window we know r2 a variable refers to the right edge in the dynamic sliding window, and constant w2 it refers to the size of dynamic window. Dynamic window size (w2) is twice the size of the original window (w).

```
Const  w2:integer        { dynamic window size }
Var r2:integer           { dynamic window right edge }
```

Dynamic window may be overlapped with the original window. When the original window moving to the right, and comes packet to process q, this packet before the original window (left of the original window) and therefore this packet are not discarded as in the original protocol. But testing this packet is in dynamic window, if the packet in dynamic window, the process q will receive the packet after authenticated otherwise will be discarded the packet.As in figure 2, there are four cases illustrate how the process q deal with packet which contains the sequence number is s

1.$s \leq r$ and $s > r-w$ (the sequence numbers in the window):
In this case, q can determine whether it has received this message before (and so it discards this message) or it has not received this message before (and so it delivers this message).

2. $s > r$ (sequence number is larger than right edge for the window):
In this case, q determines that it has not received this message before and delivers the message. Also q slides the window such that s becomes the new right edge of the window[6].

3. ($r2-w2 < s$) and ($s \leq r2$) and ($s \leq r-w$) (Sequence number is before the original window and inside the dynamic window):
In this case process q test reorder packet is received before, if the packet received in the past must be discarded, but if the packet is not received, the process q sure authentication reorder packet and then receive that packet. Thus dynamic sliding window will reduce the loss of reorder packets.

4.$s < r2-w2$ (Reorder packets before the window dynamic):
In this case any reorder packet (a very few) to the left of the dynamic window will be discarded, whether these packets received or not received.

These cases represent the test packets within windows (the original and dynamic window). The first and second cases exist in the original protocol, the third and fourth cases added to the modified Protocol, during these last two cases less loss reorder packets. The original window jumps to the right when the sequence number is greater than the right edge of the original window, while dynamic window jumps to the right when the left edge of the original window is greater or equal to the right edge of the dynamic window. Dynamic window move automatically whenever packets have been received by the process q, where the dynamic window progressing to reach the packet did not receive, and this move offers the possibility to maintain the many packets of cancellation.

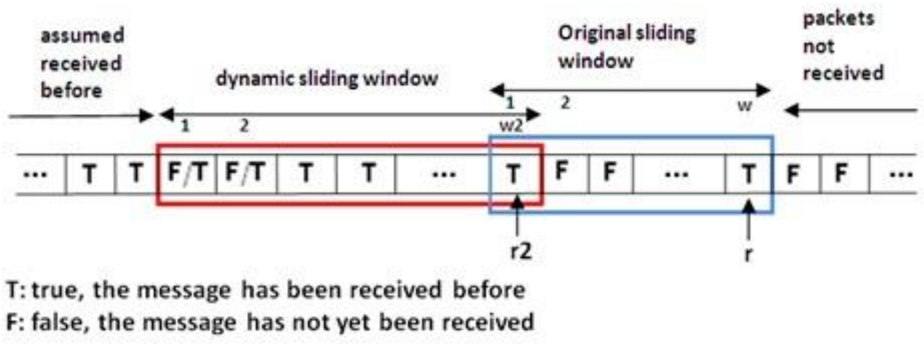
assumed received before
dynamic sliding window
Original sliding window
packets not received
1
2
w2
1
2
w
... T T F/T F/T T T ... T F F ... T F F ...
r2
r
T: true, the message has been received before
F: false, the message has not yet been received

Figure 2: The anti-replay window for dynamic sliding window and original sliding window.

a- New Algorithm

In this section we will explain the new algorithm used in the dynamic sliding window which keeps reorder packets, this algorithm is illustrated through the following points:

```
1- Start (use two sliding windows
   Where original window with size=32 and dynamic window with size=64)
  1.1- Is the packet inside the original window? Yes: go to 2
  1.2- Is the packet to the right of the original window? Yes:go to 5
  1.3- Is the packet to the left of the original window? Yes: go to 7
2- Is the packet new? No: go to 10
3- ICV is calculated
4- Is the packet authenticated?
   Yes: go to 6, No: go to 10
5- The original window is slide so that the packet's sequence number becomes the new
   right edge of the original window (with authenticate this packet). Go to 11
6- The corresponding slot in the window is marked true. Go to 11.
7- Is the packet new? No: go to 10.
8- Is the packet in dynamic window? No: go to 10.
9- Use dynamic window to allow the packet to receive on destination (process q). Go to 11.
10- The packet is discarded and message is sent to the IPSec manager.
11- End.
```

b- Representation of Processes

Secondly, we represent the original window with dynamic sliding window through the use of Abstract Protocol Notation. Process p is the same as specified in Section 5. Process q can be defined as follows.

```
processq
constw : integer {window size, w > 0}
w2 : integer
varwdw : array [1..w] of boolean, { original window}
wdw2 :array [1..w2] of boolean, { dynamic window}
r :integer, {right edge of original window}
    r2:integer, {right edge of dynamic window}
```

```
s :integer, {sequence number}
i, j : integer
begin
rcvmsg(s)from p □
if  r – w < s □ □ r □ □ {s is in window}
i := s – r + w;
ifwdw[i] □ □ {discard msg}skip
        [] □ □ wdw[i] □□ {deliver msg}
wdw[i] := true
fi
        [] s > r □ □ {s is to the right of window: }
        {delivermsg}
        r, i, j := s, s-r+1, 1;
        doi □ □ w □ □ wdw[j], i, j := wdw[i], i+1, j+1
        od;
        doj< w □ □ wdw[j], j :=false, j+1 od
        fi
        []r2-w2<s) and (s≤r2) and (s≤r-w) □ □ {s is to the left oforiginal window and inside
        dynamic window: }
i := s – r2 + w2;
ifwdw[i] □ □ {discard msg}skip
        [] □ □ wdw[i] □ □ {deliver msg}
wdw2[i] := true
fi
        []s<r2-w2□ □ {s is to the left of  dynamicwindow: }
        {discardmsg}
        skip
end
```

wdw2 array is a dynamic window, where wdw2 is used with variable r2(Sequence number Far right edge in the dynamic window).For each i, $1 \leq i \leq w2$, when wdw2 [i] =True (any that it received from process q), the wdw2 progress to the first packet is not received from the process q (wdw2[i]=false).Through the representation of process, the previously mentioned conditions (w-delivery and discrimination) carried out, and so we are checking repeat or access packet.However, the verification does not show how effective this protocol is in reducing message losses caused by long-jump reorders. In the next section will explain the results obtained through the modified Protocol and what improvement that has occurred on the original protocol.

8. Simulation results

Results obtained by modified Protocol (dynamic sliding window) will explain in detail. In this section we explain the simulation results for both protocols (original and modified Protocol), because of the difficulty of using real network to represent the discarded packets on the Internet, instead of writing programs to simulate the protocols, Simulations carried out through the use of processes (p,q), the process p sends packets stream and the process q receiving these packets based on the algorithm applied. Both protocols eliminates all replayed packets so no need to simulate this type of packets. And

therefore will be our focus in the simulation results on the reordered packets for both protocols.
Discarded reorder packets in the original protocol to the left of the original window, While discarded reorder packets (a very few) in the modified Protocol to the left of the window dynamic (dynamic window size twice the size of the original window as mentioned earlier) that lead to the reduction of loss ratio reorder packets. We used several tests for the implementation of both protocols with 1024 packet. Different probabilities used to jump the windows to the right. All probability applied to protocols for all experiments. Thus it was noted that there are a large number of discarded reorder packets in the original protocol, on the other hand are far fewer in discarded packets in modified Protocol. Results are shown in Figure 3 for both protocols.

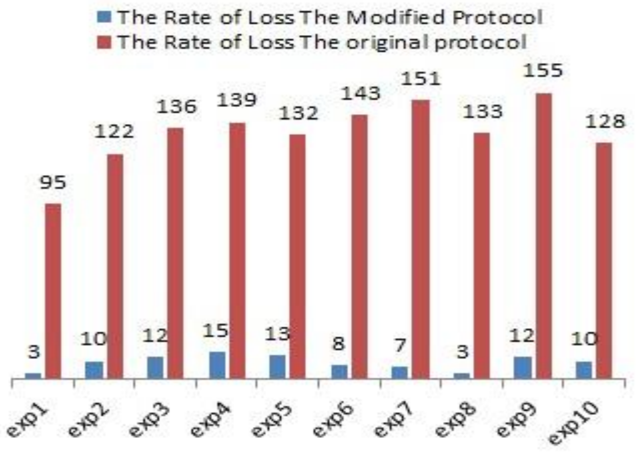

Figure 3: loss rate (15% from 1024 packet) packets discarded in original and modified protocol

Results shown in Figure 3 are the results of packets loss rate discarded. In simulation protocols we specified discarded packets loss rate to be 15% of the total packets.The modified Protocol provides highly efficient in protecting a large number of reorder packets, it is more efficient than the original protocol.The dynamic window moving to the right, whenever process q receive packet moving window to reach dynamic window to the first packet not received by process q, this dramatically reduces the number of discarded reorder packets.

9. Conclusion

In this paper we have discussed the problem that consists in IPSec. This problem is the reorder packets which occur when the window moves right. The movement of the window might lead to a loss of many packets lying in the left. The solution of this problemlies in using dynamic sliding window since it has scored a high degree of sufficiency in keeping the reorder packets. By make the dynamic window move into ways: in the first the left edge of the original window might be bigger or equal the right edge of the dynamic window. In the second, by receiving packet from the process q so the dynamic window will come to the right.

There might be some lose in the reorder packets in a rate bigger than the outputs of the simulation. This happens when a distant packet arrives to process q (when sequence number value is bigger than the total value of two windows w+w2). This case is rare.
We conclude in this paper that the new protocol introduces a solution to many reorder packets. It is more competent than the original protocol.

10. Future Works

Below we present future research directions:

1- Protecting the networks against DOS attacks by using IPSec technology, the attacker sends a sequence number, very recent one, on the right side of the window so that the window will move to a new sequence number. Consequently, a huge number of good packets will be neglected before the new window.
2- Specifying the size of the sliding window and the thresholds of the window by relying on the statistics of failure rate of IPSec technology.
3-

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

[1]G. C. Hadjichristofi, N. J. Davis andS. F. Midkiff, "IPSec Overhead in Wireline and Wireless Networksfor Web and Email Applications", IEEE, pp. 443-447, 2003.
[2] A. V. Taddeo, A. Ferrante and V. Piuri, " Scheduling Small Packets in IPSec-based Systems", the IEEE CCNC 2006 proceedings, pp. 676-680, 2006.
[3] A. V. Taddeo and A. Ferrante," Scheduling Small Packets in IPSec Multi-accelerator Based Systems", JOURNAL OF COMMUNICATIONS, VOL. 2, NO. 2, pp. 53-60, MARCH 2007.
[4] M. G. Gouda, C.-T. Huang, and E. Li, “Anti-Replay Window Protocolsfor Secure IP”, Proceedings of the 9th IEEE International Conference on Computer Communications and Networks, Las Vegas, October 2000.
[5] F. Zhao and S. Felix Wu, "Analysis and Improvement on IPSec Anti-Replay", IEEE, pp. 553-558, 2003.
[6] C.-T. Huang and M. G. Gouda, "An Anti-Replay Window Protocol with Controlled Shift", Department of Computer Sciences,The University of Texas at Austin, 2001.
[7] H. Johnson, F. Zhao, Y. Shin and S. F. Wu, "RBWA An Efficient Random-Bit Window-based Authentication Protocol ", IEEE computer society, pp. 1379-1383, 2003.
[8] W. Hao, and W. Qiaoli, " Research and Implementation of an Anti-replay Method based on WIA -P A network ", IEEE computer society, pp. 68-71, 2010.
[9] L. Sun, Z. Luo, Y. Wu and Y. Wang, " A Technique for Preventing Replay Attack in Road Networks ", College of Mathematics and Computer Science, IEEE, pp. 807-810 , July 14-17, 2012.