A parallel model on Internet Protocol Security based on Pi-calculus

Hui Kang
Jilin University
Computer Science and Technology
Changchun,China
kanghui@jlu.edu.cn

Qiuwen Yin
Jilin University
Computer Science and Technology
Changchun,China
yinqw15@mails.jlu.edu.cn

Zi Hui
Jilin University
Computer Science and Technology
Changchun,China
294042460@qq.com

Fang Mei *
Jilin University
Computer Science and Technology
Changchun,China
meifang@jlu.edu.cn

***Abstract*—IPSec protocol provides security services for network layer of OSI model. With information security becomes acute, its efficiency and security need to be improved. In this paper, we propose a parallel model for IPSec protocol framework based on Pi-calculus, analysing the part that can be paralleled in running process of the protocol and using Pi-calculus to describe message delivery. Then we use Pict, which is a programming language based on Pi-calculus, to implement and verify that compared to serial model, this parallel model gets obvious improvement on both running time and memory using.**

Keywords-component; IPSec protocol; parallel model; Pi calculus; Pict

I. INTRODUCTION

IPSec protocol[1] is a serious of protocol standards made by IETF(Internet Engineering Task Force),which is put forward to provides security services for network layer of OSI model including the protect of packets and defense against network attacks.With IPv6 replacing IPv4 increasingly, IPSec is in a wide range of uses and rising trend.So it is quite significant to improve the implementation of IPSec protocol suite in a more efficient way.

Pi-calculus[2] has a great obvious superiority in simulating dynamic communication topology,with strong ability on elaborate modeling and formal description of parallel and distributed system. Therefore, according to IPSec protocol family as well as its application and prototype of IPSec protocol used on Linux 2.6[3], an IPSec protocol parallel model is put forward based on Pi-calculus. We realized IPSec protocol model by Pict and C programming language, and compared them in running time and memory usage,then we verified that there is a definite improvement in efficiency of IPSec protocol parallel model.

II. IPSEC PROTOCOL FRAMEWORK PARALLEL MODELING

There are three parts of IPSec framework in Linux 2.6 kernel，including incoming, forwarding and outgoing packets process.

Incoming process: First,receive data from data link layer and check head of IP packet to make sure its integrity. Then start router discovery in route cache. If routing information could be found, loop forwarding these packets, which means

sending a number of packets to the handler functions pionters point to. If not, create route cache, then forward them. As for local packets, recombine the IP fragments and deliver a complete packet to upper layer protocol. And there are three security strategies to do some check, including using IPSec, not using IPSec and discard. Using IPSec means unpacking and search encapsulation type, while not using means deliver to transport layer directly.

Forwarding process: As for the packets from incoming process, check their security and construct routing table entry according to security strategies. The packets need to be safely encapsulated many times until an IPSec packet can be sent out, adding a routing table entry every time encapsulated. In this part, it only needs to deal with the ones that are not IPSec packets.

Outgoing process: For the packets from transport layer coming into this module, searching the route, if it exists, IPSec packets will be encapsulated. If not, it will add a new route entry and encapsulate IPSec packets and head of IP packets. This module is mainly used for the packets from transport layer to data link layer with them well encapsulated.

The mainly difference between Pi-calculus and process calculus like CCS and CSP is that name is the only one thing existing in the communication of Pi-calculus, which cannel, variable, data and parameter all can be regarded as a name. So Pi-calculus has two kinds of entities: process and name, and we can learn more about its grammar on relevant references.

By analysing parallelizable parts of IPSec protocol, combining the concurrency of message delivery in Pi-calculus, we define a parallel model of IPSec framework based on Pi-calculus.

The parallel model can be divided into seven modules through an abstract representation of packets receiving, forwarding and sending. The modules include data link layer, local packets processing, input processing, forward processing, output processing, encapsulation processing and transport layer. In Figure 1,the letters a,b,c,g,h,j,k,m and n indicate channels used for message delivery. Figure 2 indicates transition graphs of the parallel model after analysing every module in detail. Channel ch0,ch1...ch27 are used to transmit messages. Transport process and formal description will be given in next section.

*Corresponding author

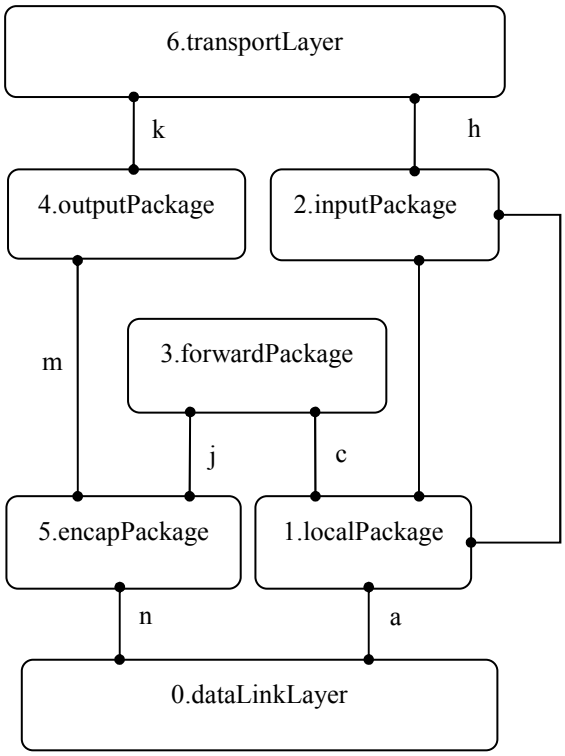

Figure 1. Channel graph of seven modules in IPSec parallel model

III. FORMALIZED DESCRIPTION OF PARALLEL MODEL BASED ON PI-CALCULUS

After understanding the process of IPSec protocol framework and analyzing the parallel model of the Pi-calculus created, the details of its overall and internal operations can be described well.

- The channels and messages associated with the data link layer module can be represented as $\tilde{d} = \{a, n, sk\_buff\}$ and D is a process which represents its internal operations. The Pi-calculus expression of this module receiving and sending packets is:

$$dataLinkLayer(\tilde{d}) \overset{def}{=} D.\overline{a} < sk\_buff > .localPackage(\tilde{l}) | n(sk\_buff).encapPackage(\tilde{e}).D$$

The data link layer module involves two modules: output processing and local packet processing, both of which can be executed in parallel. Sk_buff is a very important structure of the TCP / IP protocol stack in the Linux kernel and all the operations to packets of IPSec protocol in this paper are carried out around the sk_buff. So the messages passed in this model are all sk_buff.

- The channels and messages associated with the local packet module can be represented as $\tilde{l} = \{a, b, c, g, sk\_buff\}$ and process L represents its internal operations. The Pi-calculus expression of this module receiving and sending packets is:

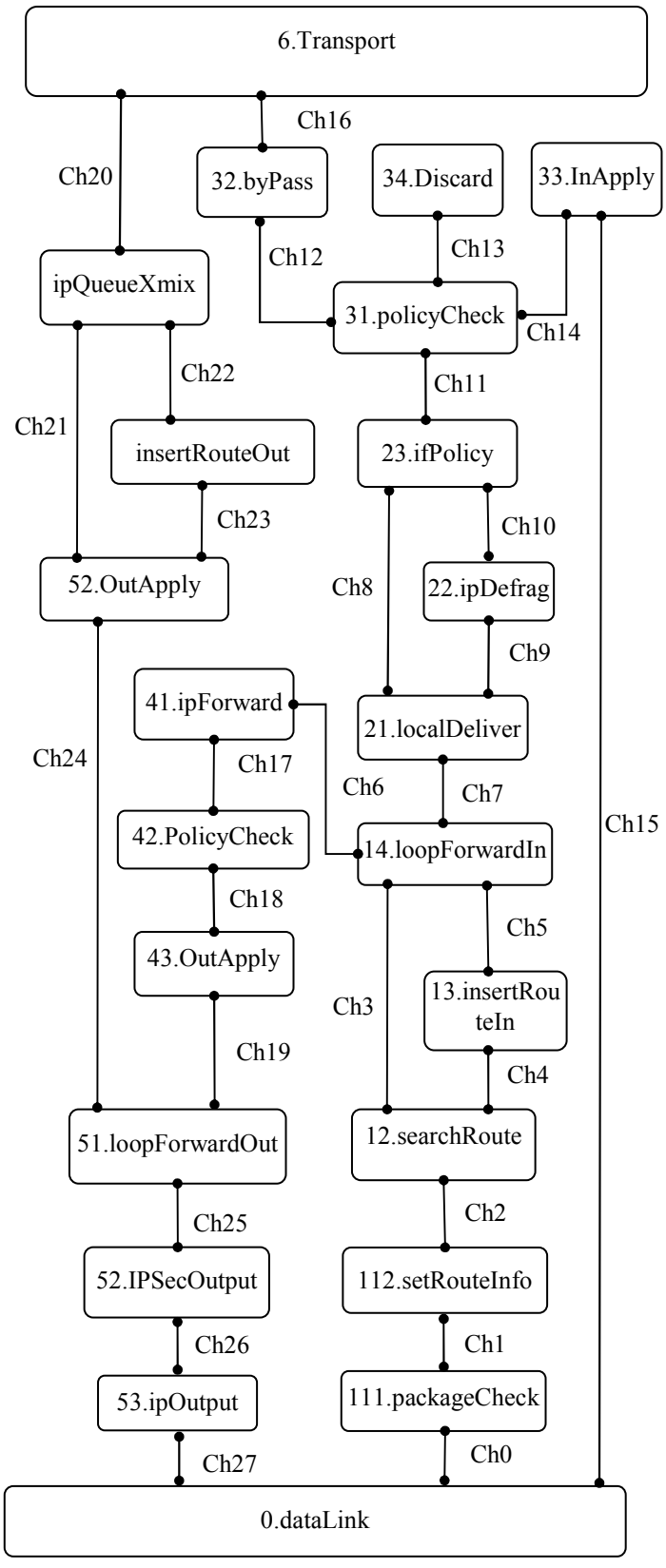

Figure 2. Transition graph of IPSec parallel model

$$localPackage(\tilde{l}) \overset{def}{=} (a(sk\_buff) \mid g(sk\_buff)).L.$$
$$(\overline{b} < sk\_buff > .inputPackage(\tilde{i})$$
$$+ \overline{c} < sk\_buff > .forwardPackage(\tilde{f}))$$

- The channels and messages associated with the input processing module can be represented as $i = \{b, g, h, sk\_buff, selector, bypass, apply, discard\}$ and process I represents its internal operations.The Pi-calculus expression of this module receiving and sending packets is:

$$inputPackage(\tilde{i}) \overset{def}{=} b(sk\_buff, selector).I.$$
$$([selector = bypass]\overline{d} < sk\_buff > .$$
$$transportLayer(\tilde{t})) + [selector = apply]$$
$$\overline{g} < sk\_buff > .dataLinkLayer(\tilde{d}))$$

Input processing module receives sk_buff and selector from the process local Package through channel b and then reach the process I.If the selector is bypass,sk_ buff is sent to the process transportLayer along channel d;if selector is apply, sk_buff is sent to the process dataLinkLayer along channel g.

- The channels and messages associated with the input processing module can be represented as $\tilde{f} = \{c, j, sk\_buff\}$ and process F represents its internal operations.The Pi-calculus expression of this module receiving and sending packets is:

$$forwardPackage(\tilde{f}) \overset{def}{=} c(sk\_buff).F.\tilde{j} < sk\_buff >$$
$$.outputPackage(\tilde{o})$$

- The channels and messages associated with the input processing module can be represented as $\tilde{t} = \{h, k, sk\_buff\}$ and process T represents its internal operations.The Pi-calculus expression of this module receiving and sending packets is:

$$transportLayer(\tilde{t}) \overset{def}{=} h(sk\_buff).T \mid$$
$$\overline{k} < sk\_buff > .outputPackage(\tilde{o})$$

- The channels and messages associated with the input processing module can be represented as $\tilde{o} = \{k, m, sk\_buff\}$ and process O represents its internal operations.The Pi-calculus expression of this module receiving and sending packets is:

$$outputPackage(\tilde{o}) \overset{def}{=} k(sk\_buff).O.\overline{m} < sk\_buff > .$$
$$encapPackage(\tilde{e})$$

- The channels and messages associated with the input processing module can be represented as $\tilde{e} = \{j, m, n, sk\_buff\}$ and process E represents its internal operations.The Pi-calculus expression of this module receiving and sending packets is:

$$encapPackage(\tilde{e}) \overset{def}{=} (j(sk\_buff) \mid m(sk\_buff).E$$
$$.\overline{n} < sk\_buff > .dataLinkLayer(\tilde{d}))$$

The internal operation of each module is analyzed and modeled in detail as follows:

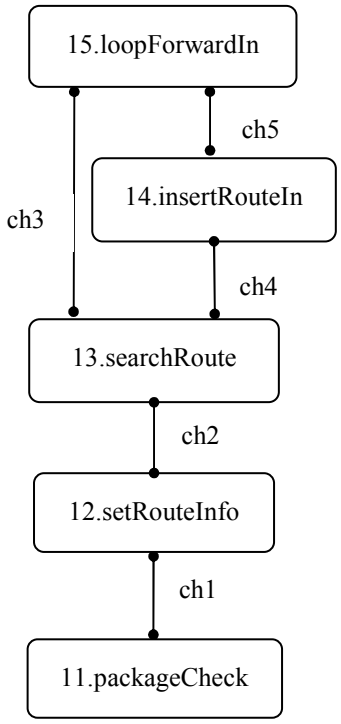

Figure 3. Channel graph of atomic operation on the local package processing module

In Figure 3, for the internal operation of the process localPackage L, it can be refined to policyCheck, setRouteInfo, searchRoute, insertRouteIn, loopForwardIn and other atomic operations.

- The channels and messages associated with the process L can be represented as $\tilde{inl} = \{ch1, ch2, ch3, ch4, sk\_buff, skb\_dst, yes, no\}$ .The Pi-calculus expression of the process L is:

$$L(\tilde{inl}) \overset{def}{=} packageCheck.\overline{ch1} < sk\_buff > .setRouteInfo$$
$$.\overline{ch2} < sk\_buff > .searchRoute.new \quad skb\_dst$$
$$(\overline{ch3} < sk\_buff, skb\_dst > [skb\_dst = yes].$$
$$loopForwardIn + \overline{ch4} < sk\_buff, skb\_dst >$$
$$[skb\_dst = no].insertRouteIn.\overline{ch5} < sk\_buff >$$
$$.loopForwardIn)$$

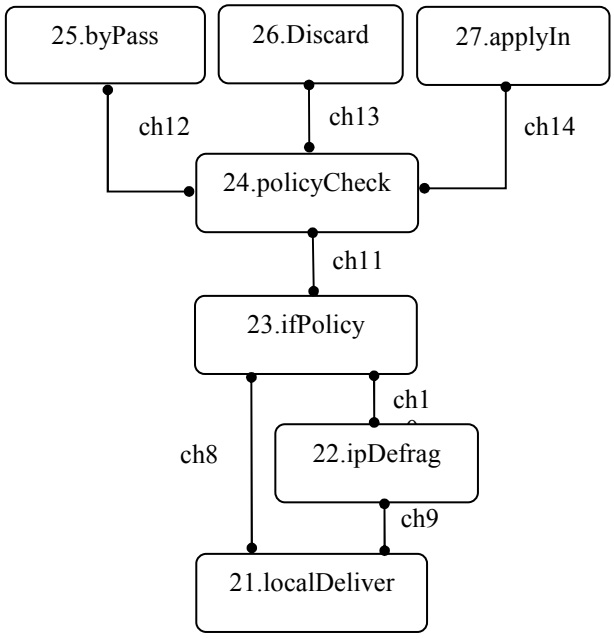

Figure 4. Channel graph of atomic operation on the input processing module

In Figure 4, for the internal operation of the process input Package I, it can be refined to local Deliver,ipDrag,ifPolicy,policyCheck,bypass,DiscardapplyIn and other atomic operations.

- The channels and messages associated with the process I can be represented as $\overline{ini} = \{ch8, ch9, ch10, ch11, ch12, ch13, ch14, sk\_buff, ip\_fm, 1, 0, selector, bypass, discard, apply\}$ The Pi-calculus expression of the process I is:

$$
\begin{aligned}
I(\overline{ini}) \overset{def}{=}\ & localDeliver.(\overline{ch8} < sk\_buff, ip\_fm > .[ip\_fm = 0] \\
& ifPolicy + \overline{ch9} < sk\_buff, ip\_fm > .[ip\_fm = 1] \\
& ipDefrag.\overline{ch10} < sk\_buff > .ifPoilcy).\overline{ch11} < sk\_buff > \\
& .policyCheck.(\overline{ch12} < sk\_buff, selector > \\
& [selector = bypass].byPass + \overline{ch13} < sk\_buff, selector > \\
& [selector = discard].Discard + \overline{ch14} < sk\_buff, selector > \\
& [selector = apply]applyIn)
\end{aligned}
$$

The process localDeliver receives packets to the local.Ip_mf is a message used to determine whether the IP packet is fragmented. If ip_mf is 1, that means this packet is one of the fragments. Along channel ch9,reach process ipDefrag to restruct IP packets, and then along channel ch10,go into the process ifPolicy.If ip_mf is 0,along channel ch8 get to the process ifPolicy.The message sk_buff is then sent to the process policyCheck along channel ch11.Selector is a policy selector. There are three types: bypass, discard, and apply whose corresponding policy types are untreated, discarded, and used.They enter into process byPass, Discard and applyIn along channel ch12, ch13, ch14 .

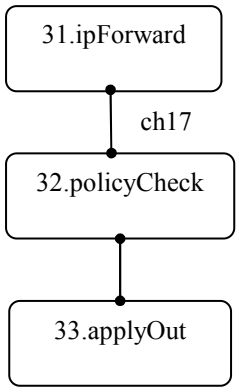

Figure 5. Channel graph of atomic operation on the packet forwarding module

In Figure 5, for the internal operation of the process forwardPackage F, it can be refined to ipForward,policyCheck,applyOut and other atomic operations.

- The channels and messages associated with the process F can be represented as $\overline{inf} = \{ch17, ch18, sk\_buff\}$ .The Pi-calculus expression of the process I is:

$$
\begin{aligned}
F(\overline{inf}) \overset{def}{=}\ & ipForward.\overline{ch17} < sk\_buff > .policyCheck.\overline{ch18} \\
& < sk\_buff > .applyOut
\end{aligned}
$$

The leading role of process F is to receive the IPSec encapsulation of non-local packets from the data link layer.First,the packet is forwarded by the process loopForwardIn and enters the process ipForward to achieve IP packet forwarding. And then enter into the process policyCheck along channel ch17.If using IPSec protocol,enter into the process applyOut along channel ch18 for IPSec outgoing process.

In Figure 6,for the internal operation of the processoutputPackage O, it can be refined to ipQueueXmit,insertRouteOut,applyOut and other atomic operations.

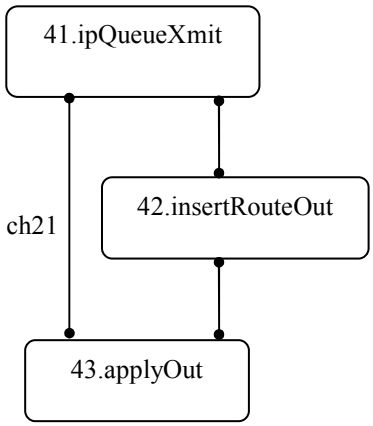

Figure 6. Channel graph of atomic operation on the output process module

- The channels and messages associated with the process O can be represented as $\tilde{ino} = \{ch21, ch22, ch23, sk\_buff, skb\_dst, yes, no\}$ The Pi-calculus expression of the process O is:

$$O(\tilde{ino}) \overset{def}{=} ipQueueXmit.new \quad skb\_dst(\overline{ch21} < sk\_buff, skb\_dst > .[skb\_dst = yes]applyOut + \overline{ch22} < sk\_buff, skb\_dst > .[skb\_dst = no]insertRouteOut.\overline{ch23} < sk\_buff > .applyOut)$$

The role of process O is to operate on the packets passed down from the transport layer,which involves the routing lookup and other operations.

In Figure 7,for the internal operation of the process encapPackage E, it can be refined to loopForwardOut,IPSecOutput,ipOutput and other atomic operations.

- The channels and messages associated with the process E can be represented as $\tilde{ine} = \{ch25, ch26, sk\_buff\}$ .The Pi-calculus expression of the process E is:

$$E(\tilde{ine}) \overset{def}{=} loopForwardOut.\overline{ch25} < sk\_buff > .IPSecOutput.\overline{ch26} < sk\_buff > .ipOutput$$

The main role of process E is to encapsulate the IPSec protocol packets in the forwarding module and IPSec packets from the transport layer,and encapsulate them into ordinary IP packets to the data link layer.

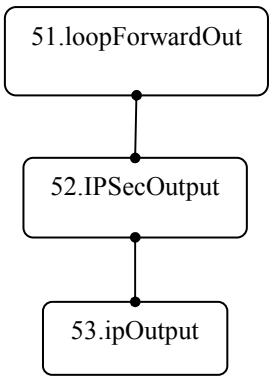

Figure 7. Channel graph of atomic operation on the encapsulation module

IV. IMPLEMENTATION OF PICT AND EXPERIMENT ANALYSIS

The Pict [5] is a programming language based on Pi-calculus,which we can use to implement the model established by Pi-calculus.The compiling environment for Pict language was first given by Benjamin C. Pierce in 1998 and published in 2000 as "Pict: A Programming Language based on the Pi-Calculus".Pict has its own unique way of handling,and it is very close to the idea of Pi-calculus[6].

For example, in the channel graph of IPSec parallel model(Figure 2),we find that function loopForwardIn involves many channels and can receive and transmit messages in parallel,which is the strength of Pict.Pict uses "|" to describe parallel.Specific code is as follows:

```
def loopForwardIn[ch3:^Int ch5:^Int ch6:^Int ch7:^Int]=
        (ch3?i =  print!"receive1"
        | ch5?i = print!"receive2"
        | ch6!1 | print!"send1"
        | ch6!1 | print!"send2")
```

In the model there can be a number of parallel processes and it can be easily expressed by Pict as well. The parallel execution between functions can be expressed as:

```
run (ipQueueXmit![ch20 ch21 ch22] | packageCheck![ch0]| ipForward![ch6 ch17])
```

In this experiment,we use C to describe IPSec protocol serial framework and Pict to describe parallel framework.Through the experiment ,we tested the real time,user time,sys time and memory using size of runing a period of time C and Pict program.The experimental data and results are as follows:

TABLE I. COMPARSION OF REAL TIME（S）

Number	Pict	C
1	0.059	0.124
2	0.058	0.114
3	0.087	0.103
4	0.059	0.100
5	0.086	0.101
6	0.059	0.104
7	0.087	0.101
8	0.059	0.107
9	0.085	0.102
10	0.058	0.118

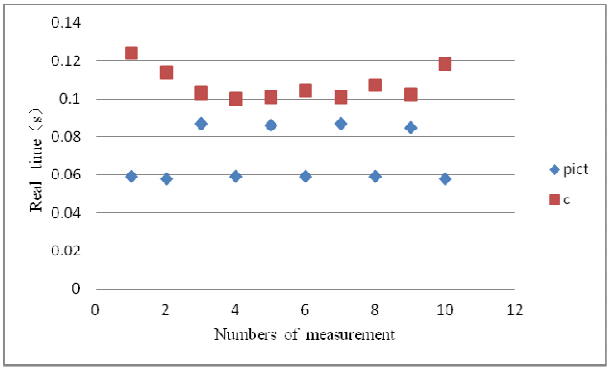

Figure 8. Comparison of real time

TABLE II. COMPARSION OF USER TIME（S）

Number	Pict	C
1	0.040	0.022
2	0.014	0.020
3	0.013	0.018
4	0.018	0.004
5	0.031	0.013
6	0.015	0.006
7	0.021	0.006
8	0.019	0.018
9	0.012	0.008
10	0.019	0.013

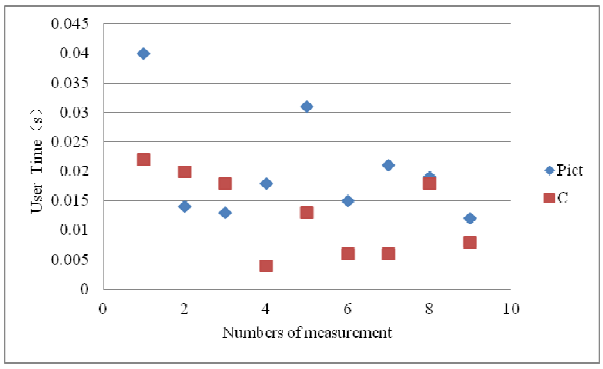

Figure 9. Comparison of user time

TABLE III. COMPARSION OF SYS TIME（S）

Number	Pict	C
1	0.014	0.070
2	0.038	0.053
3	0.042	0.051
4	0.037	0.063
5	0.030	0.054
6	0.038	0.061
7	0.036	0.061
8	0.036	0.053
9	0.043	0.059
10	0.035	0.072

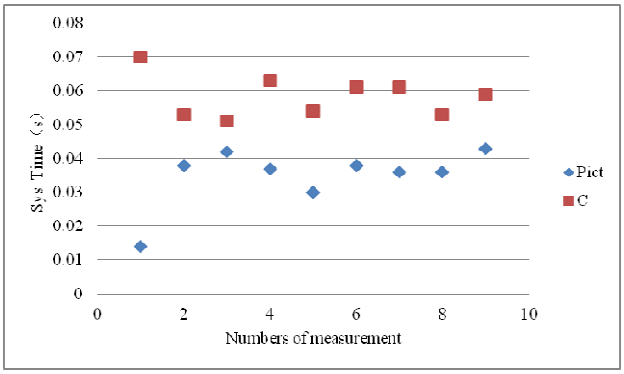

Figure 10. Comparison of sys time

We use performance analysis tools nmon to contrast to the system memory usage when C and Pict program running for a period of time.In Figure 11 which indicates the memory remaining,the first trough is the execution of the C program, and the second trough is the execution of the Pict program.

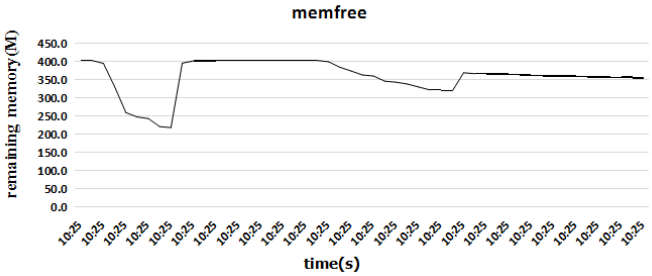

Figure 11. Remaining memory of running C and Pict Program

In the comparison experiment of runtime,combining the above figures and tables,we find that the real time and sys time C program uses are longer than Pict program.As for user time,the two is not obviously superior.This indicates the IPSec parallel framework model based on Pi-calculus is shorter time using and more efficient in the actual running of the program.

In the contrast experiment on memory use,remaining memory(memfree)of C program is lower.That is to say,when having the same memory,the memory C program using is higher than Pict program does.This proves that the IPSec parallel framework model based on Pi-calculus has lower memory usage and lower cost in multiple runs.It should be noted that memory usage of the Pict program done is not fully restored to the level bebore it is executed,because after the execution of a Pict executable,it does not automatically stop,needing a command to terminate.

V. CONCLUSION

In this paper, we combine the field of network information security and Pi- calculus to improve the framework of IPSec protocol and define a parallel model of IPSec protocol framework based on Pi -calculus. It is proved that compared with the traditional IPSec protocol framework,the parallel one not only has an improvement on running time,but less memory in the memory usage.

REFERENCES

[1] Yang Xiaohua.Analysis and Research of IPv6 Security Architecture[J].Computer Knowledge and Technology,2010(8)

[2] Milner, Robin. Functions as Processes.In Research Report 1154, INRIA,SophiaAn-tipolis. Final version. in J. Mathem. Struct.In Computer Science.1990.

[3] Zhu Xiaowei,Zhou Haigang,Liu Jun.Formal analysis and improvement of IKEv2 against man-in-the-middle attack[J].Computer Engineering and Applications,2009(15):126

[4] Milner Robin. Communicating and Mobile Systems: The Pi-Calculus,Cambridge University Press, Cambridge,UK,1999.

[5] Benjamin C.Pierce.Programming in the Pi-Calculus[M].December 9,1994.

[6] Liu Siqi.A New Distributed Communication Model Based on Pi Calculus[D].Jilin University,2015