Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 8728424, 11 pages
https://doi.org/10.1155/2018/8728424

Research Article

Cryptographic Algorithm Invocation Based on Software-Defined Everything in IPsec

Ximin Yang,[1] Deqiang Wang,[1] Wei Feng,[1] Jingjing Wu,[2] and Wan Tang [1]

[1]*College of Computer Science, South-Central University for Nationalities, Wuhan 430074, China*
[2]*Tongfang Computer Co., Ltd., Wuxi 214000, China*

Correspondence should be addressed to Wan Tang; tangwan@scuec.edu.cn

Received 26 January 2018; Revised 1 May 2018; Accepted 21 May 2018; Published 2 July 2018

Academic Editor: Sudarshan Guruacharya

IPsec was initially developed for IPv6 to ensure the communication security. With the development of Internet of Things (IoT) and the mounting importance of network security, increasing numbers of applications require IPsec to support the customized definition of cryptographic algorithms and to provide flexible invocation of these algorithms. To address this issue, an invocation mechanism for cryptographic algorithms is proposed in this paper and applied to IPsec, entitled Free to Add (FTA), based on the concept of software-defined everything. Using the idea of interface opening, the addition of a new cryptographic algorithm and updating of the existing algorithms in the algorithm library both can be achieved through the opening interfaces provided by FTA. Switching the cryptographic algorithm to be used in the FTA framework can avoid the unnecessary consumption. Besides, using the subalgorithm interface and algorithm-control interface designed here, FTA provides several software-defined invocation modes (e.g., combination and switching according to the control instruction sent by the control program) to implement hybrid encryptions or change the cryptographic algorithms for communication. Finally, the feasibility and availability of the proposed FTA mechanism are evaluated by StrongSwan.

1. Introduction

With the maturity of technologies like software-defined networking, big data, and cloud computing, the Internet of Things (IoT) affects people's lives in many areas. Software-defined mobile network (SDMN) is presented as a promising solution for IoT and works on improving wireless networking, resource management, performance, and scalability [1, 2]. However, these technologies also raise the risk of privacy leakage when creating more convenient conditions [3], and the IoT security becomes more and more prominent and severe. The development of the IoT needs the support of IPv6 technology [4–6], which provides sufficient IP addresses for IoT devices and guarantees the security of communication links through Internet protocol security (IPsec). On the basis of an IoT security architecture, IPsec can provide data security and authentication during the communication process and enhance the security of IoT [7].

Following the evolution of IPv6 in IoT and its support for IPsec, the number of applications of IPsec is constantly increasing [8–10]. Based on the open framework provided by IPsec, users can choose appropriate cryptographic algorithms for communication security [11]. However, the various public cryptographic algorithms that IPsec supports by default are not applicable in certain specialized fields. Besides, due to the increasing awareness of network security [1, 12], applications are emerging that require the addition and timely switching of customized cryptographic algorithms in IPsec to ensure higher security and confidentiality for the IPv6 networks and especially for the IoT [12].

The optional and flexible use of a cryptographic algorithm can efficiently improve the performance of IPsec and reduce the consumption of the IoT employing IPsec when the key is long enough. In recent years, the flexibility and scalability of IPsec have become more critical than before, following the rise of the concept of software-defined everything [13–15]. Meanwhile, the method of changing the cryptographic algorithms applying in IPsec for different application scenarios is becoming inefficient and complicated. At present, relevant work is mostly focused on adding IPsec cryptographic algorithms,

establishing a security gateway for network communication, and the impacts of IPsec on network applications [16–18]. However, to the best of our knowledge, there is little research on the simplification and flexibility of algorithm invocation in IPsec.

To address this issue, in this paper, we propose an algorithm invocation mechanism applied to IPsec, entitled Free to Add (FTA), by introducing the idea of software-defined everything. The contributions of this paper are as follows:

(i) To employ two opening interfaces that are first designed in FTA, introducing the concept of software-defined everything: the control program can be seen as a part of the controller in the software-defined network (SDN), and the instructions are sent to the FTA module by employing the opening interfaces

(ii) To improve the implementation of adding and switching cryptographic algorithms avoiding the unnecessary consumption caused by the traditional IPsec processes

(iii) To provide a software-defined way of invoking multiple cryptographic algorithms by constructing a combined-policy and performing a hybrid encryption process based on the combined-policy: the policy is combined in a similar manner to an entry of flow table in the OpenFlow switch in SDN.

The remainder of this paper is organized as follows. Section 2 describes the issue addressed in this paper. Section 3 presents the details of the mechanism of FTA. Section 4 explains the implementation of the FTA module and compares the performance of FTA-based IPsec with traditional IPsec. Finally, this paper concludes with Section 5.

2. Description of Problem

IPsec is a compute-intensive work which consumes more resources and requires high-performance IoT devices. Nonetheless, if the security architecture of IoT is sufficiently improved, proper invocations of low-strength cryptographic algorithms can provide sufficient security with lower resource consumption. That is to say, it is crucial and efficient for the IoT security to add appropriate customized cryptographic algorithms in IPsec and reasonably invoke these algorithms.

IPsec encryption consists of two processes: Internet key exchange (IKE) and the subsequent encryption in session (ES) carried out in the kernel [18, 19]. As shown in Figure 1, the IKE process establishes an IKE security association (SA) and negotiates the key and algorithm required for the IPsec SAs of the ES process using the IKE SA, which is performed mostly in the user layer. Then, in the ES process, the kernel establishes IPsec SAs with the negotiated key and algorithm and achieves encrypted end-to-end communication. As the cryptographic algorithms used in the two processes are implemented in different locations and provide security services in different phases, the addition of new cryptographic algorithms to IKE SA and IPsec SAs is also considered in terms of two separate cases.

Because the Linux kernel supports IPsec, mainstream IPsec virtual private networks (VPNs), such as StrongSwan and OpenSwan mainly implement the IKE process, pass the key and algorithm to the kernel, and establish IPsec SAs in the kernel layer. Thus, the addition of a new algorithm for the IKE SA can be implemented efficiently simply by adding it to the IPsec VPN.

Unfortunately, when adding an algorithm to IPsec SA, the identifier of the algorithm is required in both the IPsec VPN and the kernel for negotiation and identification, and a module for performing the algorithm is also needed for the kernel or encrypted card, requiring recompilation of all related kernel modules. Moreover, the IPsec VPN typically adopts a configuration file to invoke cryptographic algorithms and uses them in an inflexible and nonuniversal way, making the addition of cryptographic algorithms more difficult (note that the analysis and design of our work are mainly based on StrongSwan [20] since it is a widely used and mature open-source IPsec VPN with proper maintenance of version updates).

To guarantee the stability of cryptographic communication, the IPsec VPN must add the cryptographic algorithms to be used in the configuration file in advance, and then it recognizes the selected algorithm by reading the file. Nevertheless, the use of algorithms in this way is inflexible; for instance, if attackers can identify the cryptographic algorithm, they can decrypt the communication data quickly and efficiently with the relevant methods, especially when the algorithm is public. That is to say, updating the algorithms more frequently can make attacks harder and more costly, thus increasing the strength of security. A mechanism that implements flexible and extensible addition and invocation of an algorithm is therefore crucial for IPsec.

Furthermore, because IPsec can be a part of the security architecture of IoT, it will help the security architecture of IoT to be in control of the overall system security and energy consumption if the cryptographic algorithms in IPsec can be invoked using a more flexible and low-cost mechanism. The communication efficiency will be improved well while the cryptographic algorithms can be adaptively switched and changed according to the complex environment. However, the asymmetric cryptographic algorithms with high strength are often employed to ensure the security of keys in the IKE process; it means that more resources will be required. Moreover, the frequent execution of IKE process also increases the resource consumption.

It is vital and crucial for wireless networks to provide secure communications with flexible and customized encryption schemes in a wide variety of situations, especially, the secure communication between base stations and the core network. Therefore, our work in this paper is to study and attain an invocation mechanism providing flexible and expandable addition and switching of IPsec cryptographic algorithms with low resource consumption.

3. Invocation Mechanism for Cryptographic Algorithms

At its core, encrypted communication relies on cryptographic keys and algorithms. To enable the addition and switching

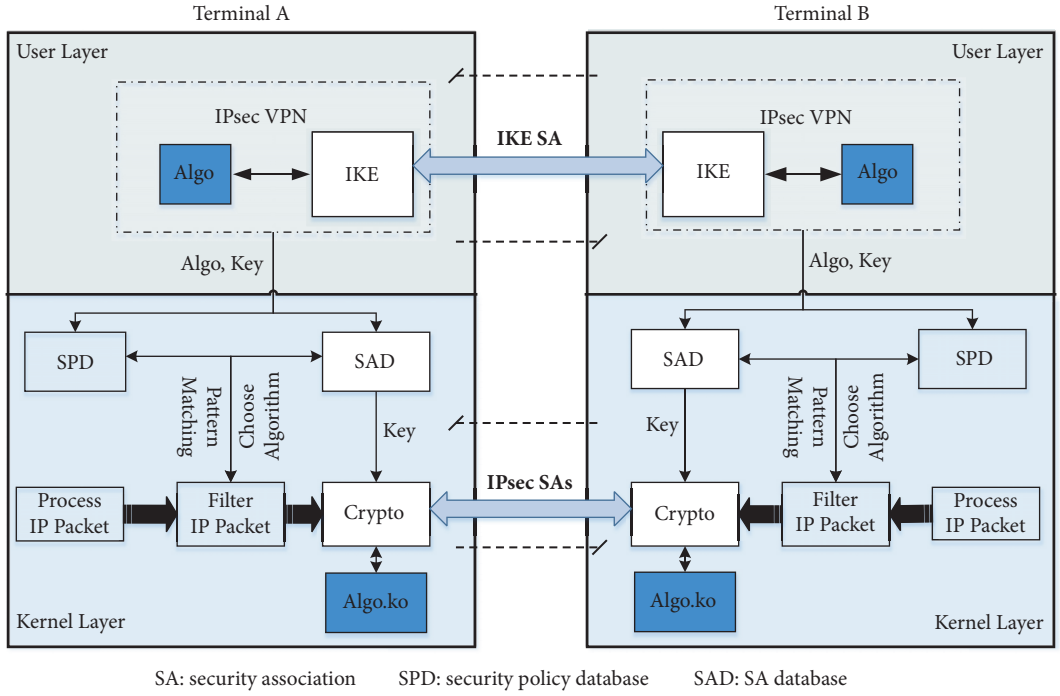


Figure 1: Communication encryption processes in IPsec.

of cryptographic algorithms on demand during an IPsec SA negotiation, an invocation mechanism of cryptographic algorithms called FTA is presented in this section. In FTA, the concepts of *opening interface* and *combined-policy* are introduced, in order to reduce the complexity of adding and switching cryptographic algorithms and the resource consumption on the basis of software-defined everything. The combined cryptographic algorithms are multigrained, and can be flexibly invoked to meet the demand for applications with a higher level of security.

3.1. Invocation Mechanism: FTA

3.1.1. Basic Introduction of the FTA Process. In the traditional IPsec, the process of algorithm identification, module addition, and recompilation of related modules is repeatedly carried out when adding a new algorithm. As shown in Figure 1, the encryption policy is fetched by an IPsec daemon (a program running as a background process) during the ES process from the security policy database (SPD) in the kernel. The IPsec daemon then invokes the algorithm module according to the corresponding algorithm identifier in the policy to perform encryption or decryption.

The process is the same as that of traditional IPsec before invoking the FTA module. The FTA mechanism makes some modifications on the basis of traditional invocation mechanism in IPsec, and the IPsec kernel directly invokes the FTA module rather than the implementing modules of algorithms. In FTA, the algorithms can be switched by the control program, with no need for reading the configuration file and negotiating again on the IPsec layer. Besides, IKE process regularly negotiates long-enough keys and no longer involves the algorithm renegotiation, which can eliminate the consumption of the duplication of IKE process caused by algorithm renegotiation.

Furthermore, the FTA mechanism simplifies the process of algorithm addition using the opening interfaces and combined-policies to enhance the flexibility of IPsec. Based on the inputting policy ID from the control program, FTA invokes a combined-policy which provides the specific encryption mode including the algorithm(s) and the location of encryption. All these are done in the kernel layer and more secure and time-effective than the process in IPsec that implements encryption by invoking Crypto API using af-alg in the user layer. As the encryption module in IPsec, af-alg is invoked in the user layer, but the encryption process of af-alg is executed in the kernel layer. In contrast, in FTA, all the operations are executed in the kernel layer, and the synchronizing signal is sent into the kernel by the control program using the algorithm-control interface without affecting the execution efficiency of the kernel.

Based on the existing IPsec workflow, the FTA mechanism just needs to add an algorithm-control interface and a subalgorithm interface to the last-invoked kernel module of a cryptographic algorithm; this is convenient and flexible for the addition and invocation of cryptographic algorithms.

3.1.2. Features of the FTA Mechanism. Compared with the method of adding algorithms in the convenient IPsec (Figure 2(a)), the FTA mechanism (Figure 2(b)) introduces certain changes, mainly in terms of the addition and invocation of cryptographic algorithms through opening interfaces.

(i) The FTA mechanism changes the invocation mode for the specific cryptographic algorithm; it does not

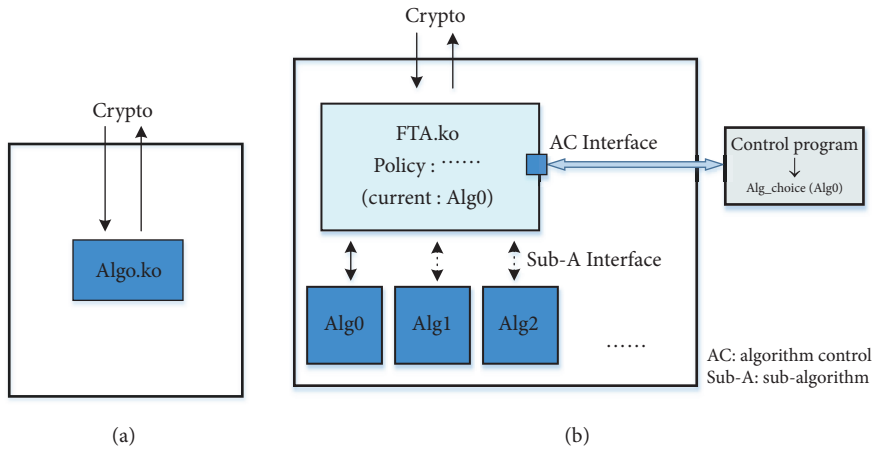


Figure 2: Algorithm addition in (a) traditional IPsec and (b) FTA-based IPsec with additive opening interfaces.

invoke the cryptographic module *Algo.ko* directly but invokes the FTA module *FTA.ko* including a cryptographic instruction and subalgorithm invocation in order to perform encryption. Then, the FTA module invokes the module of a specific cryptographic algorithm to perform encryption or decryption.

(ii) Alg_choice is a control message sent from the control program by employing opening interfaces. The control program can be seen as a part of the SDN controller.

(iii) A cryptographic instruction, given from the external control program through the algorithm-control interface, is either a single specific cryptographic algorithm or an ordered combination of multiple cryptographic algorithms, e.g., Alg1 and Alg2, through the subalgorithm interface of *FTA.ko*.

(iv) Based on the cryptographic instruction, the sender invokes the relevant cryptographic algorithm to encrypt the communication data in order, and the receiver implements the decryption in reverse. The implementation of these algorithms can be seen an encryption device controlled by the control program.

(v) The cryptographic algorithm library may be composed of multiple algorithm modules in the kernel or may be performed using a unified subalgorithm library.

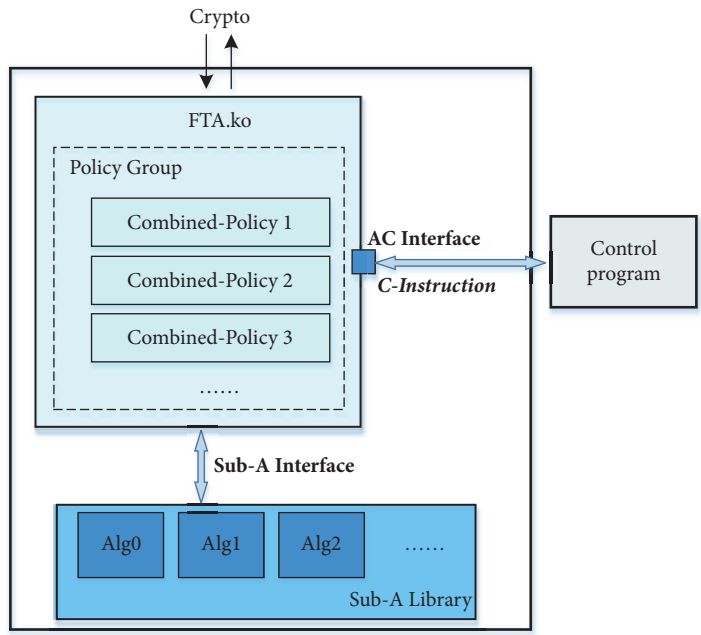
It is possible to substitute and synchronize the cryptographic algorithm through the algorithm-control interface. The combined cryptographic algorithm for an encryption/decryption operation can also be defined by adding a combined-policy of encryption algorithms or more flexible policy interfaces to the FTA module. This software-defined mechanism makes the invocation of cryptographic algorithms in IPsec more scalable and flexible.

3.2. Adding a Cryptographic Algorithm Based on FTA. In the FTA mechanism, after obtaining the encryption information from the SPD, a specific cryptographic algorithm from the algorithm library is not directly invoked by the kernel module but by the FTA module according to predefined encryption instructions. While there are more algorithms to be added to IPsec, only the implement modules of these algorithms need to be inserted into the FTA module and to be invoked through the algorithm-control interface (see Figure 3). In the same way as a plug-in device, a new cryptographic algorithm must conform to a unified interface standard, which is the only requirement for the cryptographic algorithm added or switched in the FTA mechanism. Thus, in subsequent processes, the new algorithm can be invoked through the external control program of FTA via the algorithm-control interface and policy definition. Synchronizing with the algorithm-control interface, FTA can also invoke other cryptographic algorithms without the IKE process.

Compared with the traditional method, the proposed FTA mechanism no longer has to repeatedly add the identifier of a new algorithm to the IPsec VPN, recompile *pfkey*, *xfrm*, or other modules, and modify the configuration file. The addition of the algorithm identifier and the invocation of the algorithm can be performed by using the software development method in the FTA module, thus simplifying the process of adding cryptographic algorithms without the IKE process. This approach also avoids some system-level errors caused by maloperation of the related kernel modules.

3.3. Software-Defined Invocation of Cryptographic Algorithms. The FTA mechanism invokes the combined-policy and the subalgorithms through the algorithm-control and subalgorithm interfaces, respectively; the cryptographic instructions for algorithm invocation can be software-defined using these interfaces.

3.3.1. Combining Cryptographic Algorithms. Since the flexible combination of algorithms (i.e., a combined-policy) and multiple encryptions supported by the subalgorithm and algorithm-control interfaces are possible in FTA, a subalgorithm invoking hybrid encryption can be implemented. First, two or more related algorithms are integrated into a combined-policy which is to be added to the FTA module.



AC: algorithm control, Sub-A: sub-algorithm, C-Instruction: cryptographic instruction

Figure 3: Structure of the scheme for combining and switching cryptographic algorithms.

Then, when the control program defines an alg_choice, one or more of these combined-policies is chosen through the algorithm-control interface, and data is encrypted or decrypted by invoking the corresponding algorithms through the subalgorithm interface. There are various methods to form and invoke a combined-policy. The policy can be predefined and stored in the FTA module and directly invoked based on its ID. It can also be defined using a specific format and sent to the FTA module by the control program. The following is an example of a combined-policy in JavaScript object notation (JSON), which is similar to an entry of the flow table in a OpenFlow switch in SDN:

```
{ “match”: {“src”:“IP”, “dst”:“IP”, “porto”:“http”},
  “actions”:
  [
    { “action”:“DES”,
      “params”:
          { “fragment”: {“start”:0,“end”:“63” }}
    },
    { “action”: “3DES”,
     “params”:
          { “fragment”: {“start”:64,“end”:“127” }}
    },
    { “action”:“AES128”
    “params”:
          { “fragment”: {“start”:0,“end”:“127” }}
    },
          { “action”:“out” }
  ]
}
```

Corresponding to Figure 3, the combined-policy defined above means a process of 128-bit data encryption in IPsec with following steps.

Step 1. Match the information of data (e.g., the IP addresses) to the corresponding field; if there is no match, go to Step 4.

Step 2. Encrypt the first and last 64 bits using DES and 3DES, respectively.

Step 3. Encrypt the 128-bit data again using the AES algorithm.

Step 4. Output the data and exit.

Step 5. Operate the data using the default encryption algorithm.

The process can be implemented in sequence or parallel. The decryption based on the policy is implemented in the reverse process. This method can refine the usage of cryptographic algorithms and create more flexible encryption schemes. Compared with the traditional approach to IPsec, the FTA mechanism can implement the customized invocation more easily without certain convenient customized processes such as rewriting the algorithm module and updating the configuration file.

Although the algorithm combination scheme creates extra resource overheads and decreases the efficiency of the

cryptographic processes, it can enhance the confidentiality and security of cryptographic algorithms, and the cracking and attacks using a single specific decryption algorithm are mitigated.

3.3.2. Switching Cryptographic Algorithms. FTA switches the algorithms to be used in the IPsec SA by resending the cryptographic instruction through the reserved algorithm-control interface. There is no need to design a new cryptographic algorithm or to implement the workflow of traditional IPsec, i.e., to update the configuration file of IPsec and then restart IPsec. Since the FTA module supports the policy group, algorithm switching can be achieved simply by switching the corresponding combined-policies instead of the algorithms themselves, through the algorithm-control interface; that is to say, a combined-policy can be regarded as a specific cryptographic algorithm. For instance, in Figure 3, Combined-Policy 0 can be defined as an encryption method of AES128 in the following pattern, and it will be invoked through the algorithm-control interface:

```
{
  “match”: {“src”:“IP”,“dst”:“IP”},
  “actions”:
    [
     {“action”:“AES128”
       “params”:
            { “fragment”: {“start”:0, “end”:“127” }}
     },
     { “action”:“out” }
    ]
}
```

Algorithm switching is more flexible in FTA-based IPsec than in traditional IPsec and does not require interrupting IPsec session and updating IPsec SAs. If the system should switch the cryptographic algorithm being used for data encryption to be another algorithm according to the encryption demand, the only thing the control program needs to do is update the algorithm name and the corresponding parameters in the combined-policy. Algorithms are reencapsulated in the combined-policy and synchronized as the FTA module synchronizes the policy. The attacker cannot decrypt the cryptographic algorithm to be replaced in the FTA, even if its identifier can be obtained, and the FTA mechanism therefore makes attacks costlier and more difficult.

3.4. Discussion on the Complexity of FTA. The complexity of the FTA mechanism mainly includes the space and time complexity. The space complexity primarily depends on the space requirements of combined-policies and subalgorithm library. Because the execution of the cryptographic algorithm is not implemented in the FTA module, the time complexity associated with FTA is affected by the scheme of policy query and algorithm invocation. In FTA, the invocation process is finished by only three steps and the time complexity of each step is O(1), which means the time complexity of the algorithm invocation can be negligible. Therefore, the policy query is becoming the dominant factor.

Because the data being processed by IPsec is not fine-grained, the FTA mechanism does not take up much storage space for policies and subalgorithm library and the time for searching a policy. Here, the time of policy query is decided by the volume of the policies stored in the FTA module. In the work of this paper, the policies are stored in a simple linear list, and the time complexity is mainly related to the number of policies n. The maximum value is O (n). The query method also affects the time of policy query. Several techniques, such as tree matching and parallel processing, can be used to improve the query efficiency, but we have not considered the issue yet.

As a note, in our work, we assume that the security of the network devices is ensured. In the circumstance, the customization of cryptographic algorithms is implemented by the network administrator in a software-defined manner. That is, the network operating environment applying the FTA mechanism would not be worse than that of traditional IPsec.

4. Experiment and Evaluation

4.1. Experimental Environment. In this work, the experimental environment is built on a virtual machine with an Ubuntu installation, to evaluate the performance of the FTA mechanism. The network topology and corresponding simulation environment are shown in Figure 4 and Table 1, respectively. New cryptographic algorithms and the control module developed here were installed in advance on terminals A and B.

4.2. Adding FTA Module. As mentioned in Section 2, the FTA module is added to IPsec as a new algorithm, using the traditional method in IPsec. The open architectures of StrongSwan and Linux make it straightforward to add new IPsec algorithms. Each functionality is inserted into StrongSwan as a plug-in, and the Linux kernel also provides some operations for new modules, e.g., registration, insertion, and invocation.

Furthermore, the IPsec VPN and the kernel manage to add the identifier of the FTA module. The corresponding modules need to be inserted into the kernel or encrypted card, so that the IPsec SA can negotiate the key and algorithm with the IPsec VPN and implement encryption and decryption in the kernel. Here, we give an example to illustrate this process. A modification was made to the configuration file *ipsec.conf* (as shown in Figure 5(a)) after the new algorithm *fta* (i.e., the FTA module) was added to IPsec. Its corresponding kernel module *FTA.ko* was also inserted and connected to IPsec. The results, as displayed in Figure 5(b), show that the IPsec connection was successfully established and that the FTA module was added to IPsec as a new algorithm.

Since StrongSwan supports the complete implementation of the IKE process without involving the kernel of the operating system, the addition of new algorithms into IKE is easy and convenient. Furthermore, IKE is seldom employed, and algorithm switching is seldom required in IKE. In this

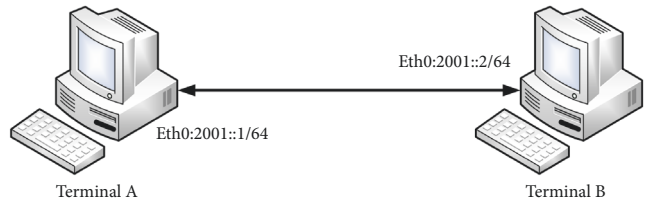


Figure 4: Topology of the testing network.

Table 1: Simulation software and hardware.

Device	Configuration
PC	CPU: Intel(R) Core(TM) i5-4200H @ 2.80 GHz memory: 16 GB, hard disk: SSD 250 GB, OS: Win10 64 bits
Virtual software	VMware Workstation 12.1.0
Virtual machine	OS: Ubuntu 14.04.3, CPU: 1∗2, memory: 2 GB, hard disk: SSD 30 GB
IPsec VPN	StrongSwan 5.4.0

paper, we therefore do not consider the ways of switching the algorithm in IKE.

4.3. Feasibility of the FTA Mechanism. In the FTA mechanism, there are two methods for adding a new cryptographic subalgorithm: one is to add this directly into the subcryptographic library, and the other is to insert the relevant implementation module of the algorithm into the FTA module. For example, algorithms AES128, DES, and 3DES are inserted into the FTA module *FTA.ko* using the second method, and Table 2 lists the combined-policies based on these three algorithms.

Figure 6 shows the two communication terminals (the Receiver and Controller) establishing an IPsec encrypted tunnel based on StrongSwan 5.4.0, in which new algorithms have been successfully added. The Server and Client of the control program communicate based on TCP, and the security of the sessions between them is also ensured, due to the existing encrypted channel. To control the algorithms synchronously, the Server and Client communicate with the FTA module through *netlink*, a mechanism that implements a certain type of datagram socket for communication between the kernel and the user space.

After establishing an IPsec SA using the FTA mechanism, cryptographic algorithm switching can be performed by the control program. This uses 0, 1, and 2 as the policy group IDs for switching the FTA encryption policies, where each number corresponds to a combined-policy group. As shown in Figure 7, these combined-policies perform well, and combinations can easily be switched. The results indicate that (1) it is feasible to add a new algorithm for IPsec via the FTA subalgorithm interface; (2) the encrypted policies can be switched flexibly via the algorithm-control interface; and (3) a hybrid encryption can be implemented when invoking a policy group consisting of multiple cryptographic algorithms in FTA.

4.4. Availability of the FTA Mechanism. Compared with the conventional method, the invocation of the cryptographic algorithm in the FTA mechanism is somewhat complicated and requires greater consumption of resources. The performance of both FTA and the traditional IPsec invocation for cryptographic algorithms AES128, DES, and 3DES is tested on the simple network given in Figure 4. This experiment evaluates the performance in terms of Ping response time and CPU occupancy rate via tools *ping*, *iperf3*, and *top*, and the results are shown in Figures 8 and 9 and Table 3, respectively.

Figure 8 shows the average Ping response time of 20 crypto-operations. From the results, it is evident that the Ping response time of the FTA mechanism is slightly longer than that of the traditional IPsec in the three cases due to the additional operations brought by FTA (e.g., the selection and switching of algorithms), but the fluctuations are all within a reasonable range. For the AES128 case, the difference between the Ping response time of the FTA mechanism and that of the traditional approach is only 0.0072 ms, which is the largest one of the three cases.

Based on the testing network given in Figure 4, we test the impact on the CPU occupancy of communication terminals when translating TCP traffic at a constant rate. Only the CPU occupancy of the sending end A is collected, since the sending end A has to consume more resources for not only generating but also sending data.

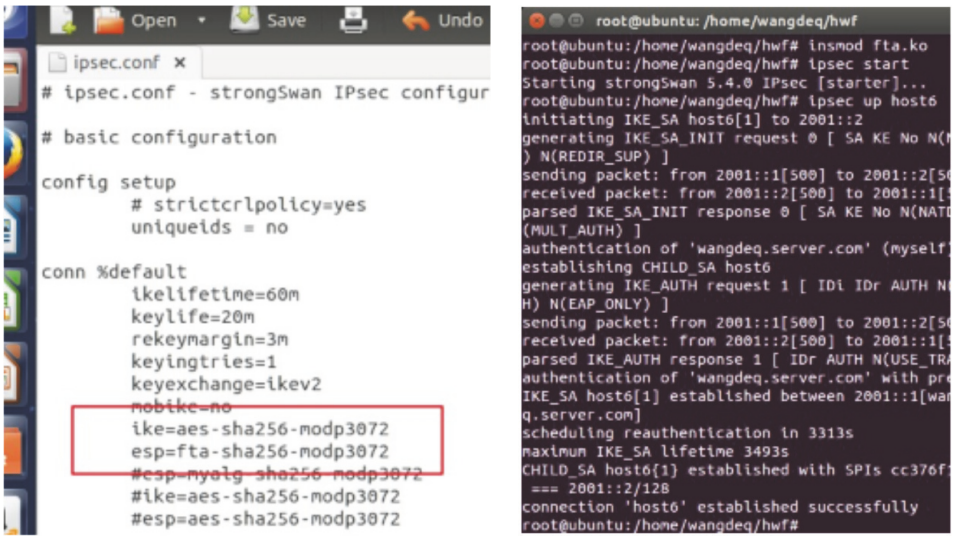
The results depicted in Figure 9 indicate that each mechanism causes the CPU occupancy rate to be increased with increasing data volume, and the trend of change is almost the same in the six cases. Even though different cryptographic algorithms, e.g., AES128, DES, and 3DES, are invoked for encryption, the CPU occupancy rates of the traditional and FTA-based IPsec scenarios are approximated. Integrating the results depicted in the three subfigures of Figure 9, the type of cryptographic algorithm has a primary influence on the CPU occupancy, while the FTA mechanism which modifies and

Table 2: Examples of combined-policies.

Policy ID	Cryptographic algorithm	Combined-policy description
0	AES128	Encrypt 128 bits of data using AES128.
1	DES, 3DES	Encrypt the first 64 bits and the last 64 bits of data using DES and 3DES, respectively.
2	AES128, DES, 3DES	Encrypt the first 64 bits and the last 64 bits of data using DES and 3DES, respectively, and then encrypt the 128 bits of data again using AES128.

Table 3: Average processing rate under full-load CPU (Mbps).

Method	Cryptographic algorithm		
	AES128	DES	3DES
Traditional IPsec	270	182	123
FTA-based IPsec	263	180	120



(a) (b)

Figure 5: Modification to (a) the configuration file for adding a new algorithm and (b) the established IPsec connection.

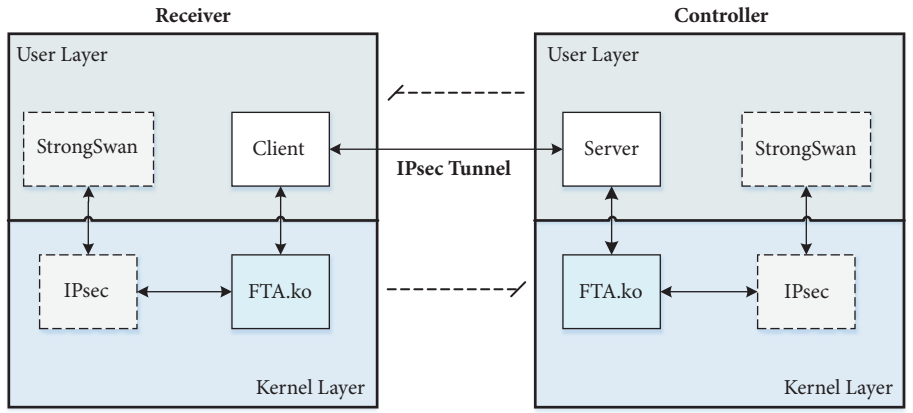
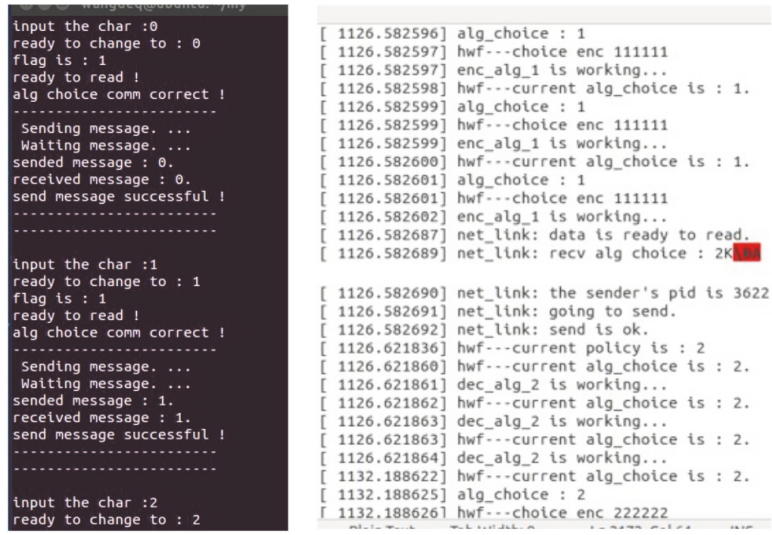


Figure 6: Frame of the designed control program for switching the combined-policies.



(a) (b)

Figure 7: Results of switching the policies of combined algorithms: (a) controlling program for switching; (b) policy switching records in the log file.

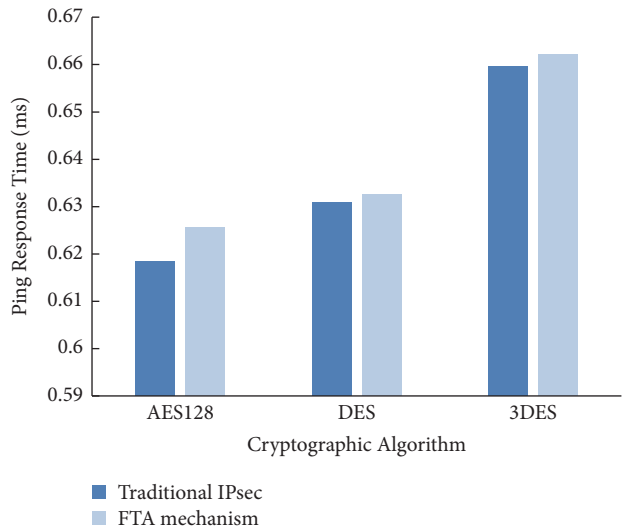


Figure 8: Average Ping response time during 20 seconds.

improves the IPsec algorithm invocation had no noticeable impact on the CPU occupancy.

Moreover, the results listed in Table 3 indicate that the average processing rate when applying the FTA-based IPsec is 97.4% of that in traditional IPsec when the CPU is at full load. Here, the data processing consists of packet generation via *iperf3*, encryption using IPsec, and forwarding to the link, and the average processing rate stands for the amount of data processed per second.

5. Conclusions

To ensure high security and confidentiality for IPv6 networks and especially for the IoT under diverse application environments, in this paper, we proposed an algorithm invocation mechanism FTA with simplified processes of the invocation of customized cryptographic algorithms in IPsec, avoiding the unnecessary consumption caused by traditional IPsec processes. In the verification experiment, FTA made the addition and switching of cryptographic algorithms and hybrid encryption in IPsec more convenient, and the algorithm invocation is more flexible. The results indicated that the definition and invocation of policy were software-defined and configurable. Furthermore, the availability of the FTA mechanism was also proven. The use of the FTA mechanism had no noticeable impact on the system performance regarding Ping response time and CPU occupancy rate.

Based on a quantitative analysis of the candidate cryptographic algorithms and the construction of a parameterized module, further work will examine the methods of

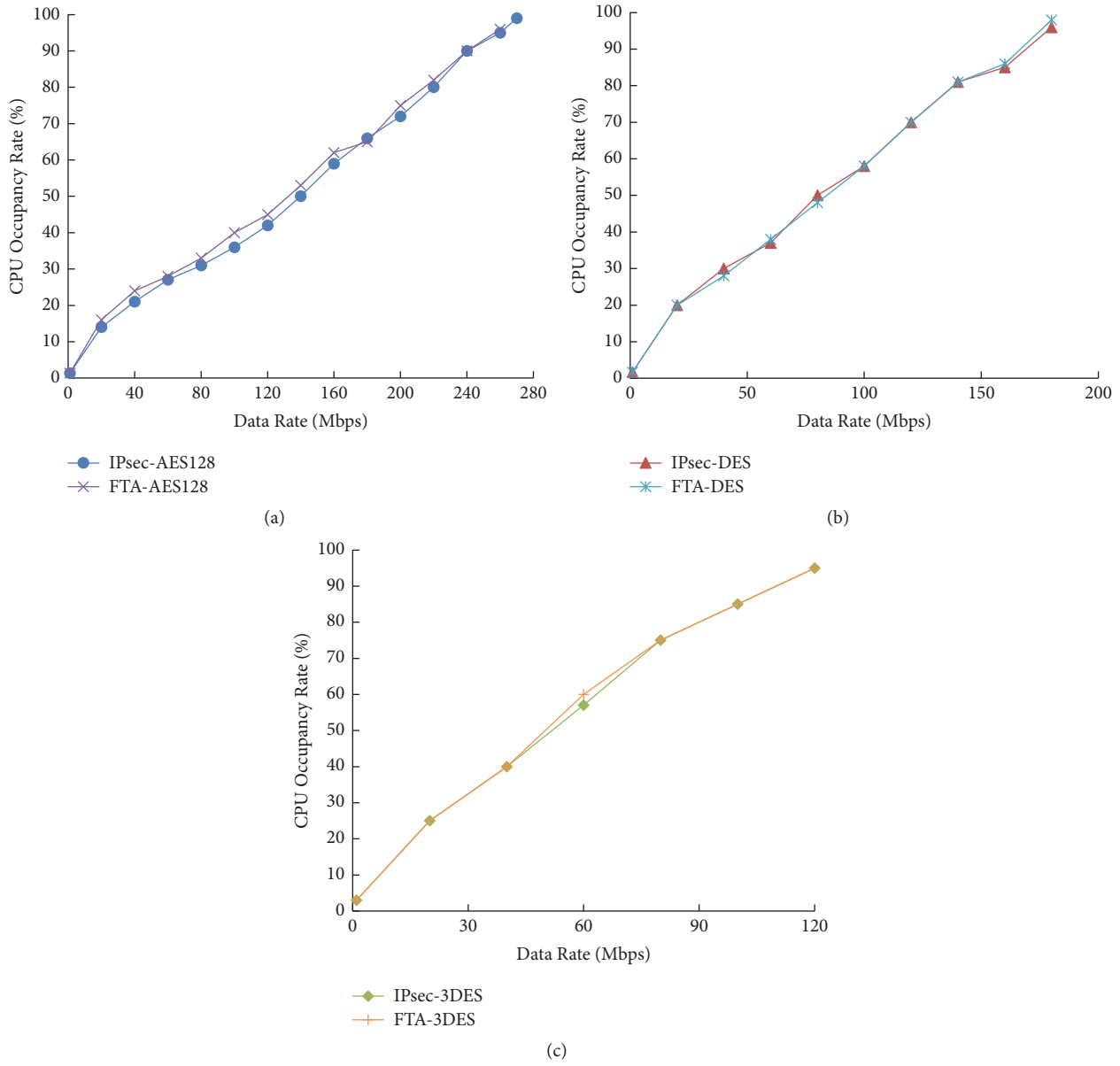


Figure 9: Trend of CPU occupancy rate influenced by data volume of traditional IPsec and FTA-based IPsec using (a) AES128, (b) DES, and (c) 3DES.

automatically generating the policy group after the standard of encryption policy is imported via the algorithm-control interface. We will also study the optimized scheme of policy query and storage in the future.

Data Availability

The data (i.e., simulation results, .xls, and resource code files, .c, .conf, and .sh) used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

The work described in this paper was carried out with the support of the National Natural Science Foundation of China (61772562), China Education and Research Network (CERNT) Innovation Project (NGII20150106), and the Fundamental Research Funds for the Central Universities, South-Central University for Nationalities (CZY18014).

References

[1] M. Chen, Y. Qian, S. Mao, W. Tang, and X. Yang, "Software-defined mobile networks security," *Mobile Networks and Applications*, vol. 21, no. 5, pp. 729–743, 2016.

[2] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[3] M. R. Bashir and A. Q. Gill, "IoT enabled smart buildings: A systematic review," in *Proceedings of the 2017 Intelligent Systems Conference (IntelliSys)*, pp. 151–159, London, UK, September 2017.

[4] M. H. Miraz, M. Ali, P. S. Excell, and R. Picking, "A review on Internet of Things (IoT), Internet of Everything (IoE) and Internet of Nano Things (IoNT)," in *Proceedings of the 6th International Conference on Internet Technologies and Applications, ITA 2015*, pp. 219–224, Glyndŵr University, Wrexham, North Wales, UK, September 2015.

[5] X. w. Wu, E. H. Yang, and J. Wang, "Lightweight security protocols for the Internet of Things," in *Proceedings of the 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–7, Montreal, Canada, October 2017.

[6] A. J. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[7] M. Irshad, "A Systematic Review of Information Security Frameworks in the Internet of Things (IoT)," in *Proceedings of the 2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pp. 1270–1275, Sydney, Australia, December 2016.

[8] M. Rao, J. Coleman, and T. Newe, "An FPGA based reconfigurable IPSec ESP core suitable for IoT applications," in *Proceedings of the 2016 10th International Conference on Sensing Technology (ICST)*, pp. 1–5, Nanjing, China, November 2016.

[9] S. K. Majhi and S. K. Dhal, "Placement of security devices in cloud data centre network: analysis and implementation," *Procedia Computer Science*, vol. 78, pp. 33–39, 2016.

[10] S. Namal, I. Ahmad, A. Gurtov, and M. Ylianttila, "Enabling secure mobility with OpenFlow," in *Proceedings of the 2013 Workshop on Software Defined Networks for Future Networks and Services, SDN4FNS 2013*, pp. 1–5, Trento, Italy, November 2013.

[11] B. C. V. Camilo, R. S. Couto, and L. H. M. K. Costa, "Assessing the impacts of IPsec cryptographic algorithms on a virtual network embedding problem," *Computers & Electrical Engineering*, July 2017.

[12] Y. Zhang, R. Yu, S. Xie, W. Yao, Y. Xiao, and M. Guizani, "Home M2M networks: Architectures, standards, and QoS improvement," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 44–52, 2011.

[13] A. Gupta, E. Katz-Bassett, L. Vanbever et al., "SDX: a software defined internet exchange," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 579-580, 2014.

[14] J. Esch, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 10–13, 2015.

[15] IEEE 5G, "IEEE 5G and beyond technology roadmap white paper," *IEEE*, https://5g.ieee.org/images/files/pdf/ieee-5g-roadmap-white-paper.pdf.

[16] V. H. F. Tafreshi, E. Ghazisaeedi, H. Cruickshank, and Z. Sun, "Integrating IPsec within OpenFlow architecture for secure group communication," *ZTE Communications*, vol. 12, no. 2, pp. 41–49, 2014.

[17] S. Samoui, I. El Bouabidi, M. S. Obaidat, F. Zarai, K. F. Hsiao, and L. Kamoun, "Improved IPSec tunnel establishment for 3GPP-WLAN interworking," *International Journal of Communication Systems*, vol. 28, no. 6, pp. 1180–1199, 2015.

[18] A. A. Al-Khatib and R. Hassan, "Impact of IPSec protocol on the performance of network real-time applications: a review," *International Journal of Network Security*, vol. 19, pp. 800–808, 2017.

[19] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen, "Internet key exchange protocol Version 2 (IKEv2) RFC 7296," 2014, https://tools.ietf.org/html/rfc7296.

[20] StrongSwan, https://www.strongswan.org/.