Design and Implementation of an IPsec VPN Gateway Base on OpenWRT

Hua Jiang¹, Qingrui Wang¹,*, Gang Zhang¹ and Jinpo Fan¹

¹Beijing Electronics Science and Technology Institute, Beijing, China

*Corresponding author e-mail: wqr_sdlc@163.com

Abstract. IPSec VPN is a virtual private network technology implemented using the IPSec protocol. Aiming at the problem that the national secret algorithm is relatively less applied to network security products, a gateway based on OpenWRT system equipment is designed. The OpenWRT system includes the complete strongswan software, making it easy to set up a VPN. The design replaces the aes algorithm, the sha256 algorithm and the ECDSA algorithm used in the IPSec VPN in the StrongSwan source code with the SM4, SM3 and SM2 algorithms respectively, and replaces the ECDSA-SHA256 certificate with the SM2-SM3 certificate according to the X509 certificate standard to achieve the national secret standard. The SM4, SM3 and SM2 algorithms are provided by the SSX0912 encryption chip, and the certificate is placed in the SSX0912 encryption chip for dense storage. It is proved that the scheme is completely feasible, and the IPsec VPN that can be realized can run stably for a long time and has high practical value. By setting up the development environment test, the gateway runs stably, has small delay, and has higher security, and the device has high communication speed, small occupation volume, high flexibility, and strong specificity.

1. Introduction

With the increasing performance and security requirements of IPsecVPN, in view of the redundancy and lack of consistency of the IKEv1 protocol [1], in October 2005, the IETF working group released IKEv2 [2] (IKE protocol, Second edition). IKEv2 simplifies the complex functions of IKEv1 and enhances security. It is embodied in a security protocol that defends against man-in-the-middle attacks, denial of service attacks, and perfect forward protection. There are commonly used IPsec protocols openswan and strongSwan in the FreeSWAN project on Linux. OpenSwan only supports the IKEv1 protocol, and the shortcomings such as the redundancy of the IKEv1 protocol impose great limitations on its future development prospects. IKEv2 will also be favored by people for its high efficiency and security. strongSwan supports both IKEv1 and IKEv2 protocols and can be widely used in different terminals, including PCs, Android phones, and IOS phones. VPN data encryption and decryption technology, tunnel technology, key management technology, and user identity authentication technology have been widely used to protect all aspects of Internet information security [3]. The cryptographic algorithms used in the data encryption and decryption technology are all standards developed by foreign organizations or institutions, and commonly used are DES, 3DES, AES, SHA-1, ECC, and the like. In order to adapt to China's own security needs, China's National Cryptography Authority has approved a series of national secret cryptographic algorithms, such as SM1, SM2, SM3, SM4, and Zu Chong's algorithm. Based on the national secret IPsec protocol standard, the realization
of autonomously controllable IPSec VPN gateway is a national security and economic development. This paper replaces the international algorithm in strongSwan, the support of the national secret algorithm, and implements strongSwan in the OpenWRT environment, builds the IPSec VPN development environment, and tests and analyzes the performance of the gateway.

2. Gateway structure
The gateway structure is shown in Figure 1, including the encryption card module and the OpenWRT development board.

![Gateway structure diagram](image)

2.1. Introduction to OpenWRT Development Board
The gateway uses the OpenWRT development board of the MT7620 model. The development board has interfaces such as GPIO, SPI and USB, which provides convenience for communication with the encryption card. The MT7620 development board is based on the OpenWRT system. The OpenWRT system is a full-featured, easy-to-modify router operating system written from scratch. It is one of the three mainstream firmwares of mainstream routers. It is a highly modular, highly automated embedded Linux system with powerful network components and scalability that retains only the network-related features you want without adding to the burden, which makes devices for network communication greatly improved in communication speed, it is very suitable for gateway development. At the same time, it also provides hundreds of compiled software, and the number of software is still increasing, without the use of frameworks to build applications to make gateway development easy and fast. The system provides support for the strongswan software, and only needs to change the strongswan source code in the firmware to implement the design of the national secret VPN.

Because OpenWRT system is often used in router systems, it embeds common routing protocols, cuts redundant parts, and strengthens network communication functions. Compared with Linux systems, network processing rates are higher, network configuration and network testing are more convenient.

2.2. Encryption card design
The encryption card uses the OMAPL137[3]-core chip that Texas Instruments has withdrawn from the US. The chip contains two CPUs, ARM and DSP. ARM is responsible for the control of the communication mechanism, and the DSP combines the SSX0912 security encryption chip to process the algorithm. The SSX0912 encryption chip integrates the hard algorithms of SM4, SM3 and SM2 and provides a dense storage space for certificate storage. ARM defines the communication mechanism and defines the communication protocol (the communication protocol is shown in Table 1) for externally providing the calling interface. The terminal can call the SSX0912 encryption card hard algorithm or read and write certificate according to the communication protocol.

| Command Name          | 5-Byte Command (0x)                      | Command Description                                                                 |
|-----------------------|------------------------------------------|-------------------------------------------------------------------------------------|
| SM4 Encryption        | 106310TTKK(106410TTKK)                   | Input: 5-byte command + 0x10 byte key + 0xTTKK byte plaintext                      |
| (Decrypt)             |                                          |                                                                                     |
| SM3 Summary           | 106100TTKK                               | Input: 5-byte command + 0xTTKK byte data                                          |
| SM2 digital           | 1071200040                               | Input: 5 byte command + 0x20 byte summary + 0x40 signature value                   |
| signature             |                                          | Input: 5 byte command + 0x40 byte public key + 0x20 byte summary + 0x40 byte signature value |
| SM2 verification      | 1072402040                               |                                                                                     |
In addition to the algorithm call communication protocol in Table 1, there are communication protocol interfaces for generating SM2 key pairs, reading and writing certificates, and verifying passwords. The user call permission is set by verifying the password interface to ensure the security of certificate storage. The algorithm is encapsulated into a function interface for the strongswan call according to the communication protocol. When the function interface is encapsulated, it must conform to the Bulk-Only transport protocol.

3. Gateway software design

3.1. Introduction to StrongSwan

StrongSwan is a complete implementation of IPsec and IKEv1 under the 2.4 and 2.6 Linux kernels. The new IKEv2 protocol and the Linux 2.6 kernel strongSwan are now fully supported for X.509 public key certificate authentication. The X.509 attribute certificate implements an advanced access control scheme based on the members of the group.

3.2. Cryptographic algorithm design

By analyzing the source code structure of strongswan, the framework structure of the libstrongswan module is shown in Table 1. It is found that there are many commonly used plugins in the strongswan source structure. AES and DES are included in the symmetric cryptographic plugin. The summary algorithm plugins used in integrity verification mainly include SHA1, SHA2, md4 and md5, etc., in addition to these, they are also openssl. The library and the sqlite library provide OpenSSL and sqlite plugins for calling interfaces. Among them, the strongswan digital signature algorithm elliptic curve digital signature (ECD-SA) algorithm is provided by OpenSSL.

Table 2. Plugin-based libstrongswan module framework structure

| Libstrongswan | Plugin Loader | Crypt | credentia ls | ...... | database |
|---------------|---------------|-------|--------------|-------|----------|
| aes | des | sha2 | pem | openssl | X509 | ...... | sqlite |

The aes algorithm replacement location is located in the aes plugin, the sha256 algorithm is located in the sha2 plugin, the ECDSA signature check algorithm interface is located in the OpenSSL plugin, and the ecdsa algorithm is located in the OpenSSL library.

3.2.1. Sm4 cryptographic algorithm design. By configuring the configuration file ipsec.conf in strongswan, the symmetric algorithm can be set to aes-128, that is, the key length is 16 bytes, which is the same as the SM4 byte length of the national secret algorithm, which simplifies the replacement of aes with sm4 jobs. By studying the strongswan source code and data printing test, I found that the aes algorithm is located in aes_crypter.c. To replace the aes algorithm, just modify aes_crypter.c. Insert the print information into the aes_crypter.c source code, and test the encryption and decryption work of aes by the two functions of encrypt() and decrypt() respectively. The data of the two functions is passed in by the chunk_t data structure, and the data in the encryption function. The encrypted ciphertext is returned by the structure pointer chunk_t *encrypted, and the decrypted function returns the decrypted plaintext through the structure pointer chunk_t *decrypted. The corresponding function sm4_crypt_ecb() in the sm4 program, the parameter int type is the encryption and decryption control bit, SM4_ENCRYPT represents encryption, and SM4_DECRYPT represents decryption. The two function names of encrypt() and decrypt() are unchanged, and the internal source code is replaced by sm4_crypt_ecb() and the input and output are kept corresponding.

3.2.2. Sm3 cryptographic algorithm design. The hash algorithm that is loaded by default in Strongswan is sha1, the length of sha1 is 96 or 160 bits, and the summary length of sm3 algorithm is 256 bits. By configuring the configuration file ipsec.conf in strongswan, the digest algorithm can be set to sha256, which is the length of the digest. It is 32 bytes, which is the same as the SM3 byte length of the national secret algorithm, which simplifies the hash algorithm replacement work. The
sha256 algorithm in the strongswan source code is quite special, providing two sha256 functions: one in the sha2 plugin, the sha2 plug provides three hash functions of sha256, sha384, sha512; the other is the openssl plugin, which is the algorithm provided by the OpenSSL plugin. The interface calls the sha256 algorithm function in the OpenSSL library. During the IPsec VPN work process, the two locations of the black 256 are crossed. The process is more complicated. In order to simplify the algorithm replacement, the most rooted interface of the hash algorithm call is found. The difference between the system call hash algorithm and the system call hash algorithm is not changed. The sm3 algorithm, which wants to do a national secret product, even if all other hash algorithms are masked, it does not affect the design's support for the national secret.

The root interface of the hash algorithm is located in the mac_signer.c source code, similar to the aes algorithm. The summary and checksum work in the source code is done by the get_signature() and verify_signature() functions. The get_signature() data is passed in by the chunk_t data structure, and the digest value is passed back to the system by uint8_t *buffer. The verify_signature() data is passed in by the structure chunk_t data and chunk_t signature. If the system returns TRUE, it means the verification succeeds. Sm3 has a corresponding hash function and a test function. The change method is the same as the sm4 change method, and is not described here.

3.2.3. X509 certificate design. Through the analysis of the negotiation process of IKEv2, it is found that the Sm2 signature and verification are mainly used in the issuance and verification of certificates, the generation and verification of AUTH values. To replace ecdsa with sm2, the certificate issued by ecdsa-sha256 must be replaced. Issued as sm2-sm3. Strongswan can generate a set of certificates issued by ecdsa-sha256, and the certificate conforms to the x509 format specification. You can use the certificate issued by ecdsa-sha256 as a template to replace the information that needs to be replaced in the certificate with the information required by the system.

The X509 certificate structure can be roughly divided into a version number, a certificate serial number, a signature algorithm ID, an issuer name, an expiration date, a subject name, a subject public key, and a signature value. Only the public key and the signature value are affected by the sm2 signature check. The part of the public key needs to be replaced with the public key of sm2. The signature value is generated by the sm2 algorithm, and the signature body is sm2 to obtain the signature value. Each SSX0912 encryption card can randomly generate a pair of public and private key pairs. The public key can be read out to issue a certificate to the user. The private key is saved in the encryption card centimeter, and the sm2 check function is called when the signature is verified. The name, expiration date, etc. of the issuer in the certificate can be changed as needed.

After replacing the Ca certificate, the server certificate, and the client certificate, the certificate is written to the encryption card by the interface of the SSX0912 encryption card to be stored in a secret state, and the source code portion of the certificate read from the path in the strongswan is read from the encryption card. take. The certificate loading function is located in the pem plugin of the source code. The ca certificate, the terminal certificate and the private key are all loaded by the load_from_file() function in pem_builder.c. After the data is loaded, it is passed to the function internal non-local variable chunk_t *chunk; structure Continue to do block processing, code conversion and other work. The function parameter char *file is used to distinguish the loading path. Different paths can be used to distinguish which one of the ca certificate, the terminal certificate and the private key is to be called, and then call the different encryption card to read the data program to take the certificate or The private key data is loaded. The load_from_file() function calls the load_from_blob() function to perform base64 encoding to hexadecimal encoding. The certificate is stored in the hexadecimal encoding into the encryption card. It does not need to be decoded. Therefore, the part of the overwriting part needs to be annotated. Drop it.

3.2.4. Sm2 cryptographic algorithm design. The private key of the CA sm2 algorithm visa certificate is stored in the encryption card of the ca end, but the public key is included in the certificate for the server and the client to read. The same server and client public key are also in the certificate. To use the public key, you need to extract the public key from the certificate. As mentioned above, the method of loading a certificate from an encryption card can be used when the certificate is loaded, and
the public key in the ca certificate can be extracted and placed in the storage area of the encryption card for use in the sm2 check.

By studying the source code, the signature program is located in the source openssl_ec_private_key.c, and build_der_signature() is the function used to generate the signature value. The checksum program corresponds to the openssl_ec_public_key.c source, and verify_der_signature() is the function used to verify the signature value. The digital signature is generated by first calculating the digest value according to the agreed digest algorithm, and then encrypting the digest value with the private key, that is, generating a digital signature. Therefore, in the build_der_signature() function, the signature algorithm in which the hash algorithm is SM3 and then the ECDSA_sign() algorithm is replaced by SM2 is replaced first, corresponding to the previously designed SM2_sign() function, and SM2_sign() is called according to the encryption card communication protocol. Sm2 signature algorithm. Verify that the signature function is verified_der_signature() in the two parts, one part is the visa verification certificate, and the other part is the AUTH payload of the verification end. The signature value of the certificate is "32 bytes R part +2~3" Byte padding value +32 bytes S part" format is passed in, the byte length is 68 bytes or 70 bytes, and the AUTH payload generated after the replacement of the SM2 signature algorithm is a fixed 64-byte check value. All judged by judging the length of the incoming check value is the check AUTH load or the visa test. When the visa is verified, the pre-stored public key is read from the encryption card, and then the certificate is checked. When the incoming certificate is judged as the peer certificate, the public key of the peer end needs to be extracted and placed on the encryption card for verification of the AUTH payload.

3.3. OpenWRT firmware compilation

The Strongswan source code is placed in the "/openwrt/dl" folder in the firmware in a fixed format. For example, the package name is "strongswan-5.3.5.tar.bz2". After the source code is modified, the same is required. The format is compressed and placed back in the "/openwrt/dl" folder. In OpenWRT firmware, each software has two levels of Makefile: one level is juxtaposed with the whole software, which is read by the firmware system, and mainly specifies some software name, software version, software hash value and software dependency library; The other level is located inside the software, there will be multiple, and the source files are juxtaposed, specifying the compilation rules of the c file.

In the Makefile read to the firmware system, there are two locations that need to be changed, one is the MD5 value of the software, and the other is the software dependency library. The software source code has been changed, and its hash value will inevitably change. In order for the software to be approved by the firmware system, the hash value needs to be changed. The original strongswan does not use the USB interface during the running process. It does not require the usb library. Now it is necessary for strongswan to use the usb protocol to call the encryption card. Therefore, the linusb library needs to be added to the strongswan software dependency library. The way to increase is shown in Figure 2 and Figure 3.

![Figure 2. Modify the strongswan dependency library (1)](image)

![Figure 3. Modify the strongswan dependency library (2)](image)

On the basis of the original Makefile, Figure 2 adds the name of the usb library "+libusb-1.0", and the addition of "+libusb-1.0" in Figure 3 is to increase the rules for compiling the source code of the second-level Makefile. After the Makefile is modified, you also need to add the strongswan software to the firmware. In the firmware directory "openwrt" enter the command "make menuconfig" to enter the configuration page, select the components of strongswan under the network vpn interface, as shown in Figure 4, the selected components are strongswan, sqlite, gmp and so on.
In addition to selecting the strongswan component, add the components of the libusb library, and then type the command "make V=99" to compile. After the compilation is complete, burn the firmware into the development board.

4. Gateway performance test

4.1. Build a network environment

After the development board burns the firmware, set up a point-to-point VPN hown in Figure 5.

![VPN network environment](image)

**Figure 5.** VPN network environment

The test environment node configuration is shown in Table 3.

| equipment     | IP           | LAN         |
|---------------|--------------|-------------|
| Router1       | 192.168.51.100 | 192.168.8.1  |
| VPN Server    | 192.168.51.138 | 192.168.32.1 |
| VPN Client    | 192.168.8.103  | 192.168.36.1 |
| PC1           | 192.168.38.101 | --          |
| PC2           | 192.168.36.120 | --          |
| PC3           | 192.168.51.110 | --          |

The VPN server adopts the Ubuntu system. The system has two network ports, one network port is connected to the external network, and the other network port is connected to the local area network. The LAN IP segment is 192.168.32.0-192.168.32.255.

4.2. Verification sm4 & sm3 algorithm

According to the test environment built in Figure 5, after the VPN tunnel is started, the data is transmitted to the PC2 on the PC1 through the UDP protocol, and the ESP[4] packet can be captured by the wireshark on the PC3. Wireshark does not support the decryption of the sm4 algorithm, so the captured RSP packet can only see ciphertext. By setting the sensitivity level of the strongswan log to 4, you can view the plaintext, ciphertext, key, and digest values of sm4 in the log. By comparison, it is found that the ciphertext data captured by wireshark is consistent with the ciphertext in the log, and the summary data is also the same. Then, the plaintext and the key are encrypted into the standard national secret algorithm and sm3 is encrypted, and the ciphertext and summary data in the log are compared with the standard national secret software, and the data is consistent and the results are consistent.

It is argued that the sm4 and sm3 algorithms are successfully replaced, and Strongswan can already support the sm4 and sm3 algorithms.
4.3. Sm2 algorithm verification

According to the test environment built in Figure 5, after the VPN tunnel is started, the data is transmitted to the PC2 on the PC1 through the UDP protocol, and the ESP[4] packet can be captured by the wireshark on the PC3. Wireshark does not support the decryption of the sm4 algorithm, so the captured RSP packet can only see ciphertext. By setting the sensitivity level of the strongswan log to 4, you can view the plaintext, ciphertext, key, and digest values of sm4 in the log. By comparison, it is found that the ciphertext data captured by wireshark is consistent with the ciphertext in the log, and the summary data is also the same. Then, the plaintext and the key are encrypted into the standard national secret algorithm and sm3 is encrypted, and the ciphertext and summary data in the log are compared with the standard national secret software, and the data is consistent and the results are consistent.

4.4. Gateway performance test

The network transmission parameters under the VPN channel are measured for performance analysis. To test the performance of the gateway, the most important performance index is RTT (Router-Trip Time)[6], the unit is ms. The ping command is used to test the Router-Trip Time, and the ping is 119 times. The result shows that the average RTT in the VPN channel is 3.611 ms, the minimum is 2.540 ms, the longest is 3.88 ms, and the packet loss rate is 0.

Then, use the iperf tool to compare the TCP throughput before and after the VPN tunnel is established. The VPN server acts as the server of iperf. The iperf startup command is: iperf –s, the VPN client acts as the client of iperf, and the iperf startup command is: iperf –c 192.168.51.138. After the VPN tunnel is established, the VPN client can access the LAN of the VPN server. Therefore, the startup command of the iperf client is: iperf –c 192.168.22.122, the throughput with the VPN tunnel.

The round-trip delay of the VPN tunnel is small, and the average delay is only 3.611ms, which is basically the same as the delay without the VPN tunnel. After adding the VPN channel, the bandwidth is changed from 91.0 Mbit/sec to 9.22 Mbit/sec. The IP video phone is connected to the VPN server and the client. After testing, the video quality is clear and voice. Smooth, it can be seen that the gateway runs stably at a high speed and the rate is 9.22 Mbit/sec.

5. Conclusion

This paper develops a VPN gateway under the OpenWRT platform, and designs and implements the design and implementation of the national IPsec VPN gateway by using the strongswan software framework under OpenWRT system. The open source software strongswan implements IPSec VPN. By replacing the cryptographic algorithm, it supports the national secret standard. The algorithms and certificates are placed on the encryption card, which is more secure and efficient. The simplicity and high network performance of the OpenWRT system improves network stability and communication speed. The gateway runs stably, the delay is small, and basically meets the application requirements, but the next step is to use the national secret algorithm in strongswan instead of replacing the international algorithm.

References

[1] Harkins D, Carrel D. The Internet Key Exchange (IKE)[J]. Rfc, 1998:2017 - 2020 Vol.4.
[2] Kaufman C. Internet Key Exchange (IKEv2) Protocol[J]. Rfc, 2005.
[3] TI. OMAP-L137 C6000 DSP+ARM Processor Technical Reference Manual[Z]. TI: 2013
[4] Kent S, Atkinson R. IP Encapsulating Security Payload (ESP)[M]. RFC Editor, 1995.
[5] Fei C, Wu K, Wei C, et al. The Research and Implementation of the VPN Gateway Based on SSL[C]// International Conference on Computational & Information Sciences. IEEE, 2013:1376-1379.
[6] Du M. Implementation of a host-to-host VPN based on UDP tunnel and OpenVPN Tap interface in Java and its performance analysis[C]// International Conference on Computer Science & Education. IEEE, 2013:940-943.