Security agility solution independent of the underlaying protocol architecture*

Valter Vasić and Miljenko Mikuc

University of Zagreb, Faculty of Electrical Engineering and Computing,
Unska 3, 10000 Zagreb, Croatia
{valter.vasic, miljenko.mikuc}@fer.hr

Abstract. Cryptographic protocols are constantly exposed to new attack methods. When some cryptographic protocol gets exposed there is a need to replace it. This is hard because most cryptographic protocols are hard coded in applications. Applications should implement a way of negotiating cryptographic protocols used. In that way old and vulnerable protocols could be easily replaced with new ones. The agile cryptographic negotiation protocol (ACNP) proposed in this paper represents a layer-agnostic, robust solution that can be deployed for providing cryptographic agility and greatly improve security. It provides minimal communication overhead and represents a universal and secure solution independent of the communication layer and application that uses it.

1 Introduction

Cryptographic protocols represent the main tools to achieve a secure environment in network communications. Symmetric cryptography is used to ensure secrecy, asymmetric cryptography provides for signing and authentication whereas hash algorithms ensure data integrity. A combination of all these types of algorithms is needed for securing network communication. Digital signatures and HMACs (Hash Message Authentication Code) are examples of these combinations. Digital signatures are a combination of hash functions and public key cryptography. HMACs represent a tool for providing integrity, where the message is concatenated with a common secret, and then hashed. HMACs exploit the possibility of two nodes to generate a common secret to protect future communication, such as Diffie-Hellman.[1]

A wide range of cryptographic protocols is available for each cryptographic method. Newer and better algorithms are introduced constantly. At the same time new and more efficient attacks against the existing algorithms are found and thus become less secure. This problem can be solved by using cryptographic protocol agility which would enable the introduction of newer protocols without changing specifications and implementations of communication protocols.

As cryptographic protocols become less and less secure during time so do the protocols that use them. A fair number of protocols (SEND [2], GSM, CDMA)

* AT2012, 15-16 October 2012, Dubrovnik, Croatia.

use only one hard coded version of cryptographic protocols that has been specified when the protocol was designed. This could automatically make the protocol vulnerable when the used version of the cryptographic protocol has an efficient attack against it. Similar problems are happening to the GSM protocol because the cipher, A5, has successful attacks against it. The static cryptographic protocol definition problem is already recognized and solved in specific protocols like the SSL(Secure Sockets Layer)/TLS(Transport Layer Security)[3][4], SSH(Secure Shell)[5] protocol and IPsec protocol [6]. However these protocols can assure security just for a specific application (SSH for a remote shell and data transfer) or on a specific layer (SSL/TLS only works on the application layer while IPsec works only on the IP layer).

One of the possible pitfalls of SSL/TLS and IPsec is the aim to provide a whole security solution and implement both the negotiation algorithm and the needed cryptographic algorithms. This greatly complicates the whole implementation and possible deployment.

This paper recommends a new negotiation protocol that enables the possibility to negotiate cryptographic protocols instead of deciding upon and using only one version of the needed cryptographic protocols. This would enable the usage of cryptographic protocols just from the aspect of what they provide. Lets assume that a designer wants to assure integrity for the messages that are transferred through the network. For assuring integrity a cryptographic hash algorithm should be used. The negotiation protocol would negotiate the preferred version of the needed protocol and forward the agreed protocol to the application. The negotiation would be integrated in the application and would greatly simplify and secure the protocol that is being designed as it prevents the usage of an outdated protocol. It creates an abstract layer for the programmer who could use cryptographic protocols as a tool without thinking of the specific implementations and versions.

2 Background

Signature algorithm agility is introduced in [7]. [8] argues the benefits of hash agility in the SEND protocol, a security extension for the ND protocol in IPv6. It proposes a negotiation approach which could be used for determining which hash algorithms should be used for securing communication using the SEND protocol. An interesting notion is also the usage of predefined suites which could simplify negotiation and achieve a complete set of cryptographic protocols with less communication overhead.

The SSH specification [9] introduces a way to negotiate algorithms for which we can assume that works well because it is thoroughly tested. A list of acceptable algorithms is exchanged in order of preference. The chosen algorithm must be the first algorithm on the client's list that is also on the server's list. If there is no such algorithm, both sides must disconnect.

On the other hand [5] addresses the need for every SSH server host to have it's own key so that each server can be unambiguously identified. The client must

have a priori knowledge of the servers key. Two different trust models can be used:

- The client has a local database that associates each host name with the corresponding public host key. This method requires no centrally administered infrastructure, and no third-party coordination. The downside is that the database of name-to-key associations may become burdensome to maintain.
- The hosts name-to-key association is certified by a trusted certification authority (CA). The client only knows the CA root key, and can verify the validity of all host keys certified by accepted CAs.

The usage of public key for host communication enables a secure environment for communicating and negotiating cryptographic protocols.

Cryptographic and security agility solutions have been provided for managing security protocols in a private (corporate) network. [10][11] The main obstacles of deploying such solution in a distributed manner is the centralized architecture that depends on a single system that distributes and manages security preferences throughout the network. A similar system that attempts to distribute symmetric cryptographic keys in an Active Directory managed network is also dependent on the domain controllers. [12]

3 Cryptographic agility

The notion of cryptographic agility enables the improvement and adaptation of new cryptographic algorithms opposed to the currently specified fixed selection of cryptographic algorithms. Cryptographic agility would mitigate the problems of transitioning to a newer version of a cryptographic protocol which is thoroughly analyzed in [13]. Agility coupled with a well tested negotiation protocol enables data protection through cryptographic protocols by mitigating the interoperability issues between communicating network nodes. The importance of agility has been acknowledged even in older protocols such as ATM [14]. Cryptographic agility has also been proposed to be implemented on boards based on FPGAs [15].

3.1 Negotiation principles

Negotiation methods are the same as in the SSH protocol. The SSH protocol has a server side and a client side. In this protocol the client is the side that initiates the negotiation, whereas the server is the other host. The basic idea of the negotiation protocol can be found in [16]. An example will be shown to illustrate the behavior.

Before the algorithm lists can be exchanged the hosts need to pass through the initial leap of faith [17] and accept their respective public keys. When the keys are acquired they can digitally sign the messages that are needed in the negotiation process. This is done only once for a specific host or until its IP

address or public key change. The same procedure is needed in the SSH protocol when the hosts communicate for the first time.

The initiating host (client) supports the following hash algorithms and prefers to use them in the following order:

1. SHA-256
2. SHA-512
3. SHA-1
4. MD5

The other host (SSH server) supports hash algorithms in this order:

1. SHA-1
2. SHA-256
3. MD5

Although it's unconventional to place the less secure SHA-1 protocol before the SHA-256 protocol such ordering may be enforced by a security policy that is commonly present in an enterprise environment.

An illustration of the message exchange is shown in figure 1.

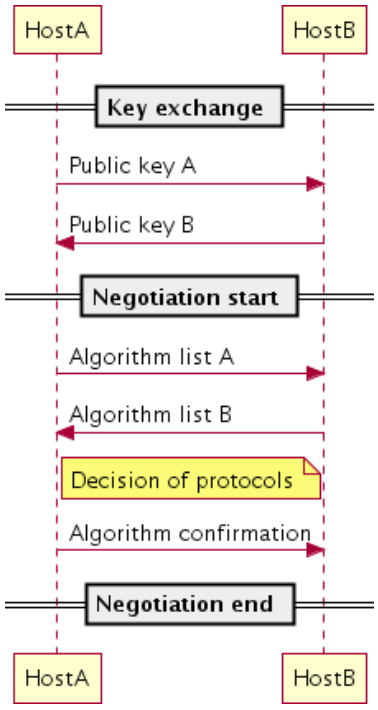

Fig. 1. Negotiation message exchange

After the ordered list of protocols is exchanged between the hosts the decision is made as follows. The first item that is on the clients list and is also present on the server list is the SHA-1 protocol, which will be chosen for usage in the application that initiated the negotiation. Since negotiation in the SSH protocol is just the beginning part there is no need for additional confirmation. In the negotiation protocol it is necessary to send a confirmation message to confirm the decision or an abort message if the nodes couldn't agree on an algorithm.

The list could be further enhanced by introducing weight values to the algorithms in the algorithm list. This can increase granularity and facilitate creating custom lists that abide by a specific security policy.

4 Negotiation protocol

The placement of the Agile cryptographic negotiation protocol (ACNP) in the TCP/IP stack could be anywhere in the IP protocol stack. It can be deployed directly above the data link layer, above the IP layer, and above the transport layer (TCP, UDP, SCTP, ...). UDP is a good choice since it removes the handshake overhead caused by TCP and simplifies message delimiting. Lower layers are similar to UDP but remove the additional header data (UDP, IP header) On the other hand TCP provides a guaranteed delivery of messages. The choice of the layer is left to the programmer integrating the Agile Cryptographic Negotiation Protocol (ACNP) in an another protocol. The ACNP implementation will provide the possibility to use IP and Ethernet layers and the most widely used transport protocols (TCP, UDP, SCTP). An illustration of the protocol stack can be seen in figure 2.

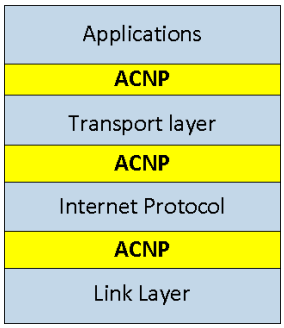

Fig. 2. Agile cryptographic negotiation protocol stack

The first message that is sent from the initiating host contains the following components:

– Random Sequence Number (RSN) - Random value computed by the client before sending the initial message. This number will be a primitive session

identifier. It will be present only in the first message, but it will be hashed for the digital signature in the other messages following the initial message.
- Host Identifier (HID) - A simple ID used for host identification. It is generated as a hash from the host public key and the application specific host discriminator. In the future the HID could be generated from more data to give a higher level of security.
- Algorithm List (AL) - List of algorithms in the preferred order as a delimited string. The algorithm list is divided by algorithm types.
- Digital Signature (DS) - message hash encrypted with the senders private key so that the message could be authenticated on the receiving side and could not be changed without detection.

The other host then replies with its own algorithm list, host identifier and digital signature. After this step both sides know which protocol they will use. To confirm this the initiating host sends an OK message to end the negotiation.

4.1 Data structures

Each ACNP side must have a data structure to hold the list of public keys of other communicating hosts. The structure would be similar to the `known_hosts` file used in the SSH protocols for storing hosts with whom the client previously communicated. The structure is filled on the first communication with a host. Two additional messages are exchanged as a part of the first negotiation. This two messages contain the initiating host public key and the other host public key respectively. After this initial exchange the hosts can safely negotiate cryptographic protocols.

```
Host_ID1:Pub_key1: Alg_type1:Alg_choice1:Timeout
                   Alg_type2:Alg_choice2:Timeout
                   .
                   .
                   .
Host_ID2:Pub_key2: Alg_type3:Alg_choice3:Timeout
                   Alg_type2:Alg_choice4:Timeout
                   .
                   .
                   .
```

Fig. 3. Agile Cryptographic Negotiation Protocol hosts data structure

This structure could also have the previously negotiated preferred protocols to minimize the need to renegotiate protocols through time. The previously negotiated algorithms could also be stored by the application using ACNP. An example of the ACNP hosts data structure, with the record of previously negotiated protocols, is shown in figure 3. The structure also has a `Timeout` value

which can be configured in the implementation. The aim of the timeout value is to enable renegotiation after a certain amount of time to enhance security and change to a new cryptographic algorithm.

4.2 Hash and public key algorithms used to secure negotiation

To secure the negotiation the messages need to be secured by using at least two algorithms:

- A hash algorithm to ensure integrity and generate host IDs
- A public key algorithm that will be used for message signing

For hashing messages the Secure Hash Algorithm that has a fixed output of 256 bits is used (SHA-256). Host IDs will also be generated using the SHA-256 algorithm.

For digital signatures the RSA algorithm is used and a minimal 2048 bit key length must be used. Also the public host keys that are exchanged need to be at least 2048 bit long. Host private and public key must be generated before the first negotiation. It is important to safely store the host private key to avoid security breaches within the ACNP.

In the future implementations the proposed hash and public key protocols will be changed to more secure and up to date versions of the same protocols, i.e. SHA3-256 for generating hashes and ECDSA-256 for public key encryption.

Although the algorithms need to be changed in the future the negotiation mechanism exchanges only three messages. This significantly prolongs packet collection for a potential brute force attack against the cryptographic algorithms used for negotiation.

4.3 Message formats

Figure 1 shows the three messages used for negotiation. This messages have the following formats:

- Acronym explanations
  - *RSN* - Random sequence number
  - *HID* - Host identifier
  - *AL* - Algorithm list
  - E_{pubC} - Public key encryption with public key from C (client)
  - E_{pubS} - Public key encryption with public key from S (server)
- *Message 1*:

$$RSN, HID_C, AL_C, E_{pubS}(Hash(RSN, HID_C, AL_C))$$

- *Message 2*:

$$HID_S, AL_S, E_{pubC}(Hash(RSN + 1, HID_S, AL_S))$$

– *Message 3*:

$$HID_C, OK, E_{pubS}(Hash(RSN + 2, HID_C, OK))$$

If the communicating sides can't agree on a cryptographic protocol two abort messages are provided to replace the second and third message respectively:

– *Abort 1*:

$$HID_S, ABORT, E_{pubC}(Hash(RSN + 1, HID_S, ABORT))$$

– *Abort 2*:

$$HID_C, ABORT, E_{pubS}(Hash(RSN + 2, HID_C, ABORT))$$

Figure 4 shows the message exchange with the abort messages.

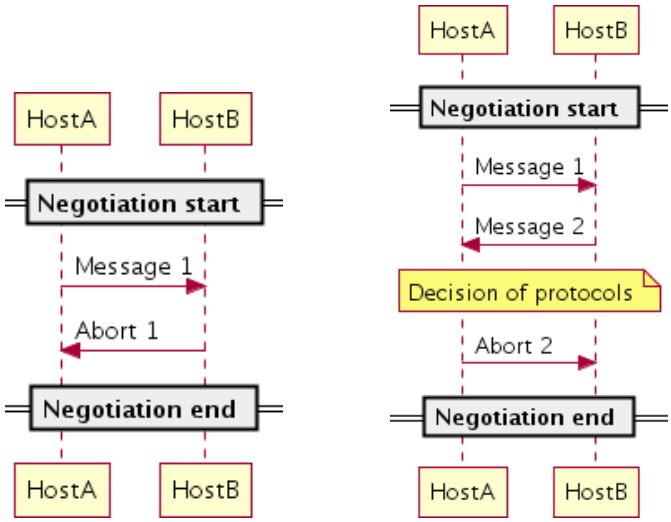

Fig. 4. Negotiation message exchange with abort messages

5 Protocol and security considerations

For the protocol to function properly the code regarding cryptographic functions must be agile. Agile code only defines a required cryptographic function like hash, symmetric encryption or asymmetric encryption and enables dynamic specification of algorithms that are to be used. This is the main prerequisite for the ACNP to be efficient and deployable.

An efficient way for fetching supported algorithms list is needed. The supported algorithms list may depend on the operating system, programming language and their currently used versions. Fetching mechanisms will differ based on the OS and programming language. It is essential to create a fetching procedure that enables the usage of new and enhanced versions of protocols after system updates that affect a systems ability to use security algorithms. ACNP is by design independent of the operating system on which it is used.

Depending on the transport protocol that is used the specific implementations need different approaches. A TCP session failure is easy to detect, whereas an UDP communication needs to have timeouts to detect communication problems. An UDP packet is easier to spoof than a message in a TCP session.

Timeouts and connection limits should also be present to mitigate resource attacks as system resources are needed to keep track of current negotiations. Lower layer mechanisms could be used to prevent attack but that depends only on the deployment environment. Replay attacks should be mitigated by treating the random sequence number as a nonce field and drop all packets that have the same RSN field as messages before. For that purpose all initial RSN values should be stored.

While the negotiation process is protected the initial public key exchange still remains vulnerable to denial of service attacks. The attacker could change the key along the way since the key exchange isn't protected.

6 Conclusion

While creating an application a designer must take into account the security risks of exposing data that is exchanged through the network. To mitigate these risks cryptographic algorithms can be used. Most applications end up having fixed cryptographic algorithms or a badly implemented negotiation protocol. This problem is solved by introducing the Agile cryptographic negotiation protocol (ACNP) which enables the negotiation and introduction of new and updated versions of cryptographic protocols. The ACNP uses the principles from the SSH protocol which is widely deployed and tested. It translates those principles into a peer to peer environment so that it can be used between any two nodes for negotiating and introducing newly supported cryptographic algorithms.

References

1. Schneier, B.: Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2nd edn. John Wiley & Sons, Inc., New York, NY, USA (1995)
2. Arkko, J., Kempf, J., Zill, B., Nikander, P.: SEcure Neighbor Discovery (SEND). RFC 3971, Internet Engineering Task Force (March 2005)
3. A. Freier, P., Kocher, P.: The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101 (Proposed Standard) (August 2011)
4. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol version 1.2. RFC 5246 (Proposed Standard) (August 2008)

5. Ylonen, T., Lonvick, C.: The Secure Shell (SSH) Protocol Architecture. RFC 4251 (Proposed Standard) (January 2006)
6. Kent, S., Seo, K.: Security Architecture for the Internet Protocol. RFC 4301, Internet Engineering Task Force (December 2005)
7. Cheneau, T., Laurent, M., Shen, S., Vanderveen, M.: Signature Algorithm Agility in the Secure Neighbor Discovery (SEND) Protocol. draft-cheneau-csi-send-sig-agility-02 (June 2010)
8. Vasic, V., Kukec, A., Mikuc, M.: Deploying new hash algorithms in secure neighbor discovery. In: Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on. (September 2011)
9. Ylonen, T., Lonvick, C.: The Secure Shell (SSH) Transport Layer Protocol. RFC 4253 (Proposed Standard) (January 2006)
10. Petkac, M., Badger, L., Morrison, W.: Security agility for dynamic execution environments. DARPA Information Survivability Conference and Exposition, **1** (2000) 0377
11. Petkac, M., Badger, L.: Security agility in response to intrusion detection. In: Proceedings of the 16th Annual Computer Security Applications Conference. ACSAC '00, Washington, DC, USA, IEEE Computer Society (2000) 11–
12. T. Acar, M. Belenkiy, L. Nguyen, and C. Ellison: Key Management In Distributed Systems (2010)
13. Bellovin, S.M., Rescorla, E.K.: Deploying a new hash algorithm. (2005)
14. Tarman, T., Hutchinson, R., Pierson, L., Sholander, P., Wirzke, E.: Algorithm-agile encryption in atm networks. Computer **31**(9) (sep 1998) 57 –64
15. Paar, C., Chetwynd, B., Connor, T., Deng, S.Y., Marchant, S.: An algorithm-agile cryptographic co-processor board on fpgas. PROCEEDINGS- SPIE THE INTERNATIONAL SOCIETY FOR OPTICAL ENGINEERING (1999)
16. Mikuc, M., Vasić, V., Brčina, S., Kovačević, L., Marić, M., Marini, T., Pokornić, V., Vežnaver, R.: NgP agreement protocol (croatian). (May 2012)
17. Arkko, J., Nikander, P.: Weak authentication: How to authenticate unknown principals without trusted parties. In Christianson, B., Crispo, B., Malcolm, J., Roe, M., eds.: Security Protocols. Volume 2845 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2004) 57–66