SEcube™: Data at Rest and Data in Motion Protection

Antonio VARRIALE [1], **Paolo PRINETTO** [2], **Alberto CARELLI** [2], **Pascal TROTTA** [3]
[1]Blu5 Labs Ltd, Blu5 Group, Ta Xbiex, Malta
[2]Cyber Security National Lab, CINI & Politecnico di Torino, Italy
[3]Lero (The Irish Software Research Center), University of Limerick, Limerick, Ireland

Abstract – *Current trends for ubiquitous data usage have made information security as a mandatory component of any system. The availability of suitable levels of protection for data is required to secure any kind of content throughout its lifecycle and independently from the media, which allows the data to be used. In this paper we present a methodology to provide data protection through a simple and effective security abstraction layer based on the SEcube™ (Secure Environment cube) single chip, a new security-oriented open hardware and software platform. After analyzing the most critical information states, we introduce a set of easy-to-use APIs that provide an open-source, multi-paradigm security layer, suitable to protect both data at rest and data in motion. Being the SEcube™ made up of three hardware elements (a highly powerful processor, a Common Criteria certified smartcard and a flexible FPGA), all the functions are implemented and executed in a fully controlled secure environment. All the complexities related to key management and algorithms are handled within the secure environment, leaving the developers free to focus on the final applications and services.*

Keywords: Security, Data Protection, Hardware Platforms, Open-Source

1 Introduction

Information and data exposure during its entire lifecycle is growing continuously. The constant increase of connectivity, bandwidth, and mobility allows hackers and malicious users from inside and outside organizations to steal and monetize valuable information such as medical records, intellectual properties, bank transactions, and national secrets.

Security is critical, but comes at a cost. Such cost is not to be seen just as a money matter, but as the effort required to integrate security into suitable vertical solutions and the additional time and effort spent by the users to learn and implement the new processes. In this paper we propose a solution aimed to reduce these three aspects to a minimum. The acquisition cost is lowered by resorting to the SEcube™ integrated platform; the integration cost is drastically reduced by a low-impact deployment of a layered security technology, and the complexity of security processes in daily tasks is conceived by an abstraction level which guarantees that user's habits remain unchanged.

Information lifecycle is conveyed through various technologies, each having one (or more) dedicated security solution(s). Some examples are HTTPS for web surfing, S/MIME for e-mailing, VPN for private networking, and many encryption software solutions for HDD storage.

In such a heterogeneous universe of technologies, this paper introduces a new methodology. More abstractly, we distinguish information as being, at any time, either *at rest* or *in motion* and we provide for each a solution that is independent from the underlying technologies: SE*file* protects data at rest and SE*link* protects data in motion.

These solutions, based on the SE (Secure Environment) technology, are implemented according to a common strategy that eases the above-mentioned costs [2].

In order to minimize the actual cost of the product, a multi-paradigm approach is provided. The clients can tailor the solution according to their needs, from a full standard and software-only implementation to a fully customized hardware implementation, from open-source to commercial IPs.

Organizations can adopt the SE based solutions on top of all the pre-existing data storage, management and manipulation systems, such as clouds, e-mails and web based services. SE-based implementations are portable, independent of the Operating Systems and multi-level. How to do this in the software and hardware development platform is detailed in [3] and [4].

Finally, the cryptography complexity is concealed by common abstractions like groups, scopes and policies, in such a way that even the concept of encryption key is invisible [4, 5]. This approach allows users to enjoy security with no impact on their current habits.

The following sections will explain the main features of these solutions.

2 Data at rest and data in motion

Before Internet and the wideband mobile connectivity diffusion, the information lifecycle was much more controlled. Nowadays, as shown in Fig. 1, every piece of data may run complex and unpredictable paths from its creation to its final archiving or destruction.

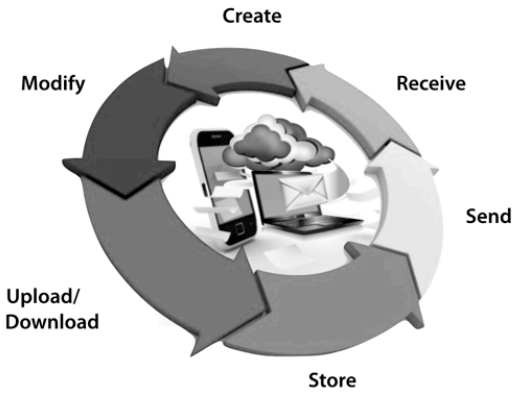

Fig. 1 – Information life cycle

In this scenario several kinds of contents (e.g., documents, messages, and voice/video streams) may require different protection techniques according to their state. For example, cryptography was created to protect communications defined as the transfer of sensitive information between two or more separate entities. This is the case of *data in motion*. However, when information is static, as for stored documents, we talk about *data at rest*.

Cryptography alone is not enough to guarantee a proper data protection, especially when the system weakness may vary according to the information state. It is important to identify a scalable and easy to use methodology able to protect both data at rest and data in motion. In the sequel we introduce such a methodology based on the SEcube™ security platform.

3 SEcube™ multi-device, multi-level, multi-paradigm libraries

The SEcube™ is an open source security-oriented platform in a single chip package produced by Blu5 Group. As described in [1] and [2], the SEcube™ chip provides several communication interfaces, which allow to embed it in any kind of device. In order to reduce the time to market, Blu5 Group also provides ready-to-use devices such as the USE*cube*, a powerful secure USB device which embeds the SEcube™ chip. Even starting from the development board (Fig. 2), OEMs and integrators can easily create diverse form-factor devices according to the final operational environment (e.g. USB tokens for PC/Laptop usage, PCI express boards for servers usage, etc.).

Fig. 2 – SEcube™ development board

On the software side, the SEcube™ platform comes with multi-level, hierarchical libraries, which provide three main levels of APIs:

- Level-0, Communication functionalities (e.g., driver-less USB communication channel) explained in [1]
- Level-1, Security common functionalities (e.g., login, logout, key inject/remove, basic cryptographic functions, etc.) explained in [1]
- Level-2, Service level functionalities (e.g., secure file system functions, negotiation) explained in the next sections.

Due to the open source nature of the project, cryptographic experts are free to modify the software implementations at any level, creating new libraries to their taste. A shown in Fig. 3, each functionality can be implemented in several paradigms according to the required security, performances, and compliance targets.

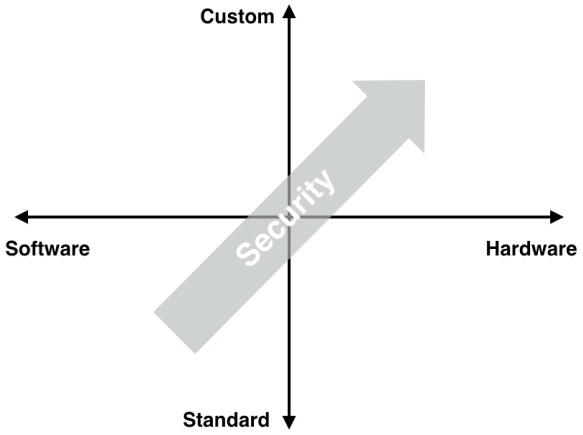

Fig. 3 – Multi-paradigm implementation diagram

For example, the basic cryptographic functions can be easily implemented as a firmware running in the SEcube™ embedded CPU. Nevertheless, an implementation based on the SEcube™ embedded FPGA can be considered when the performances in terms of speed are crucial. Alternatively, the use of the SEcube™ embedded smart card becomes mandatory when the final solution has to achieve specific certifications and compliances.

A software implementation is sometimes enough to guarantee the required security comfort. In this case a full software implementation of the SEcube™ (virtual SEcube™) can be used instead of the chip. In any case, whatever the final implementation is, the SEcube™ libraries are written in ANSI C in order to maximize their portability. Possible Operating System dependencies are isolated in a few modules and the whole code can be wrapped (e.g., JNI, PHP, JavaScript, etc.) to fit any development environment.

Providing a multi-device, multi-level, multi-paradigm approach, the SEcube™ platform is a flexible candidate to deploy security solutions in several scenarios, perfectly matching the final requirements even in complex and heterogeneous systems.

4 SEcube™ easy keys management

A security system becomes appealing when both developers and users are not aware of its complexity. For this purpose, the SEcube™ platform is based on concepts like *closed communication groups* and *security policies* other than keys and cryptographic parameters.

In both data at rest and data in motion based applications and services (e.g., local storage, cloud, email, messaging and voice calls) the sensitive information should be accessed and managed by a specific group of users, only. In the simplest case, the group is made up of one user (e.g., myself, for personal purposes). In other cases, many people can be involved (e.g., file transfer and chat rooms).

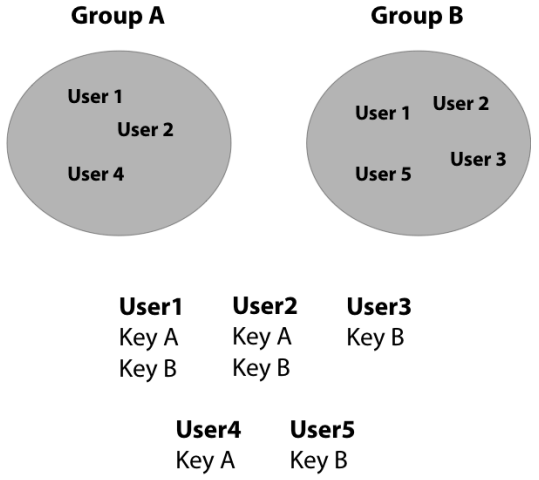

Fig. 4 – Groups and Keys

A group is a pool of one or many users. It is featured by a group communication key, which will be used to generate session communication keys for that group, and a set of security policies (e.g., cryptographic algorithm used to protect the information related to that group and mechanism to generate the session keys) which will be used to decide if the users are entitled to belong to that group.

As shown in Fig. 4, Group A is made up of three users (User 1, User 2, and User 4) whilst Group B is made up of four users (User 1, User 2, User 3 and User 5). The groups are usually created by a security administrator (managed groups). Nevertheless, groups can also be created manually by the users (manual groups), according to the specific organization security policies. In both cases the security architecture is the same.

Whenever a group is created, a group communication key is automatically generated by the SEcube™ and every user receives all the communication keys related to the groups which it belongs to. The SEcube™ platform provides APIs for the secure distribution of group keys to the entitled users.

Referring to Fig. 4, it is easy to understand that two or more users can access the information only if they share at least one group communication key. For example, User 1 and User 3 can access the same information, since they belong the same group (Group B) and, accordingly, they share the same key (Key B). When users share more than one key (they have more than one group in common) the key related to the smallest group is selected, since it is known by less users (more secure). As detailed in paragraph 6, this mechanism is part of the negotiation process, which is executed every time a secure communication link is established.

On the programming side, every group is identified by a unique number (GroupID). Each SEcube™ can manage up to 256 communication groups. In addition, there are two special groups: *personal* and *family*. The *personal group* is associated to the user, which owns the SEcube™ based device. It is used to manage personal, non-shareable information and its GroupID is made up of all zeroes. The *family group* is associated with all of the users inside an organization. It is used to manage information which can be shared with everybody and its GroupID is made up of all ones.

This approach allows developers and users to focus on the final secure service, since they are not required to be familiar with keys and algorithms (e.g., kind of key, key size and algorithm type). The cryptographic complexity is easily and transparently managed by the SEcube™ platform according to the security policies set at provisioning time by a security administrator. On the other side, cryptographic experts and developers are free to customize any part of the system, thanks to the open source nature of the platform.

5 SE*file*, the SEcube™ based Secure File System

Data at rest can be easily protected through the SE*file* technology, a Secure File System level-2 library which allows standard applications to access standard file systems through the SEcube™ cryptographic layer, performing file encryption, signature, and name remapping.

There are several technologies to protect file systems. For example, the ORI File System [6] aims to provide a security solution for distribute file systems, replicating mechanisms like multi-user cloud like drop box and version control like Git. Another example is the Secure File System module [7], which is implemented at Linux kernel level to encrypt any folder with encrypt prefix. All these solutions depend on specific implementations (e.g. operating systems, file systems, etc.) and sometimes are more focused on extra functionalities (e.g. cloud, versioning, etc.) than security. For these reasons they may be very invasive, forcing the users to change their habits, and at the same time they are not so portable. On the contrary the SE*file* technology is independent of the other layers (e.g. operating systems, file systems, etc.). It can be easily integrated in any pre-existing systems (e.g. drop box) and allows developers to create very low-impact and portable security solutions focusing on the content protection.

As shown in Fig. 5, SE*file* operates as a transparent virtual layer on top of the regular file system, providing a set of multi-paradigm, Level-2 APIs (SFS_Open, SFS_Read, SFS_Write, SFS_Seek and SFS_Close). Since the SE*file* is implemented on the top of the standard file systems functions, it is independent of the Operating System.

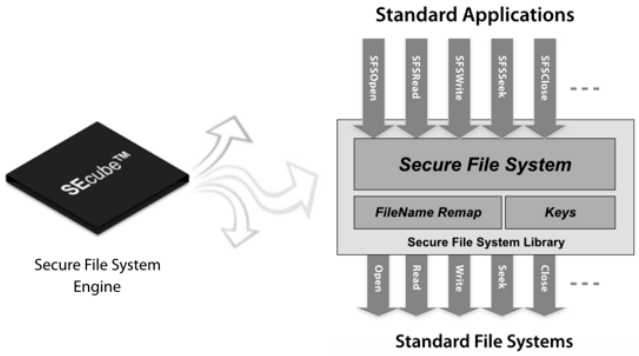

Fig. 5 – SE*file* library concept

Being open source, the SEcube™ implementation can be verified or improved at any time. Nevertheless, the developers that do not feel comfortable with cryptography, or simply trust the open source community system validation, can benefit from the security abstraction level, which simplifies all the cryptographic mechanisms.

For example, when a secure file is created through the SFS_Open function, it is just required to specify the protection scope: for me only (personal), for a group of two or more (group), for everybody (family). When the protection scope is a group, a group identifier must be provided as described in section 4. By doing so, users and developers do not need to deal with keys, algorithms and other low level complex cryptographic features, which are transparently managed inside the SEcube™.

As shown in Fig. 6, a secure file is made up of many encrypted and signed sectors. The first sector is partially encrypted, since it contains the secure file header, which includes some clear and coded fields (e.g. initial vector and security scope). All other sectors are fully encrypted. Each sector is 512 Bytes long in order to be read/written atomically and prevent possible data corruption issues, especially on mobile devices (e.g., low battery).

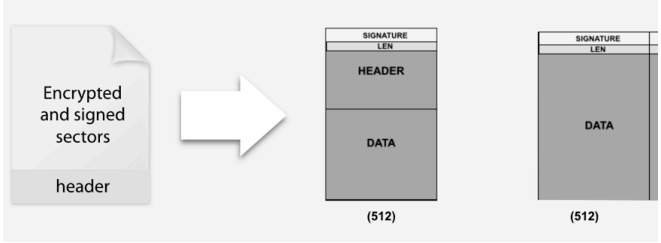

Fig. 6 – Secure File structure

As shown in Fig. 7, when a secure file is created, the file name is coded in a way that nobody can recognize it looking directly at the physical file system. For example, the coded file name can be calculated inside the SEcube™ by a one-way digest function (e.g., SHA256) of the real file name in combination with an SEcube™ common information (family secret), whilst the real file name is encrypted in the secure file header. In this case a folder containing secure files looks like:

…/B5B8D0F121F9596A…4BBC4829615B968EDA.se
…/B5A6135EF9B1783C…F981BC7AA9F541D93F.se
…/B5E93F7BC1DD18D…8AF3C6BA6135EF9B17.se

According to the library paradigm, SE*file* can be provided with different implementations, such as hardware cryptography (accelerated implementation on the embedded FPGA), software cryptography (firmware implementation on the embedded CPU), or extra features (multi-level secure cache, anti reply attack, auto rescue, etc.).

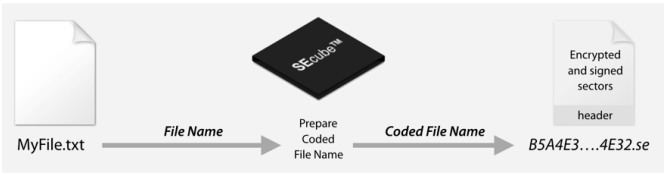

Fig. 7 – Coded file name

Beside some basic posix style APIs, the SE*file* library provides file management functions like SFS_GetFileList (return the list of secure files in a specific folder showing their real file names) and SFS_DeleteFile (delete a specific secure file).

The simple set of posix-like APIs makes SElink easy to be integrated in third-party libraries or applications. As shown in Fig. 8, an interesting library integration example is the combination of SE*file* and SQLite.

SQLite offers a virtual file system interface to interact with the underlying operating systems, thus integrating SE*file* it can be provided with a strong, fast and customizable security layer to the host environment. This approach is very efficient: it keeps the SQLite interface unaltered and no changes are required at the application level.

Fig. 8 – SE*file* and SQLite integration

SE*file* can be also combined with third-party solutions to generate secure application and services. For example, SE*file* can be used in a Dropbox™ folder to deploy a zero-impact secure cloud solution. In a similar way, SE*file* may be used in combination with an email client to store mails and attachments. Although the same technology may be used to send and receive files, the next section presents a more effective and flexible library to protect data in motion.

6 SE*link*, the SEcube™ based Secure Link

Whenever an information item is transferred among entities, the connection links offer an attractive opportunity for attackers to catch sensitive contents.

There are many solutions which provide security for the most standard communication channels, like TLS [8] for IP based data transfers, Virtual Private Networks (VPNs), etc. Nevertheless, all these solutions are implemented for specific communication channels, sometimes they are complex, their overhead is not acceptable and usually provide point-to-point only security.

In line with our general approach, we propose a solution which is very light, easy to be integrated in any environment (over any protocol), open source, portable, multi-device and multi-paradigm: SE*link*.

The SE*link* technology allows protecting data in motion on both point-to-point and point-to-multipoint links through the SEcube™ cryptographic layers, which performs negotiation, encryption, and signature operations. It is a security layer running on top of any transport technology (e.g. IP, TCP, HTTP/S, SMTP, XMMP, CoAP and custom protocols) and featured with a very low overhead in order to be fast and easily integrated.

When two or more entities need to set up a secure communication channel to exchange data, the SE*link* library negotiates the session keys and performs encryption, signature, decryption and verification operations on the exchanged data.

As shown in Fig. 9, in the first phase the SE*link* performs a negotiation process checking the security policies (e.g., methods and master keys to derive session communication keys), agrees the common parameters (e.g., cryptographic algorithms, predefined communication groups/ keys, etc.) and derives the final session communication keys. After performing a successful negotiation, the secure link is ready for data transfer and the negotiated keys can be used to encrypt, sign, verify, and decrypt the information.

Fig. 9 – SE*link* point-to-point example

Negotiation can be duplex or simplex. The duplex negotiation must be executed between two entities (peer-to-peer) that will be co-responsible in generating the final session communication keys. Other entities can join the link at a later stage (peer-to-multi-peer) performing a simplex negotiation, which requires a master entity, already in the link, and one or more slave entities willing to join the link. The master entity pushes negotiation parameters that allows slave entities to generate the communication session keys already negotiated. The generation is possible only if the slaves are entitled to join the link (e.g., security policies match and same communication group).

The negotiation process is performed in two messages, only. For example, in the case of HTTP/S links, the negotiation is performed in one standard GET method.

SE*link* provides a small and easy way to use set of multi-paradigm, Level-2 APIs to manages both binary data, suitable for protocols like IP, and text based data, suitable for protocols like HTTP. In order to simplify the integration on stateless systems (e.g., web servers) SE*link* functions are multi-sessions and the internal state can be saved outside the SEcube™ platform in a secure (it is encrypted and signed) and private (it can be reused by the originator SEcube™ only) container automatically generated by the library. As shown in Fig. 10, each entity is able to concurrently manage several SElink channels. In this case, A and C work on two secure links at the same time and three total sessions are negotiated: A-D, A-C, and B-C.

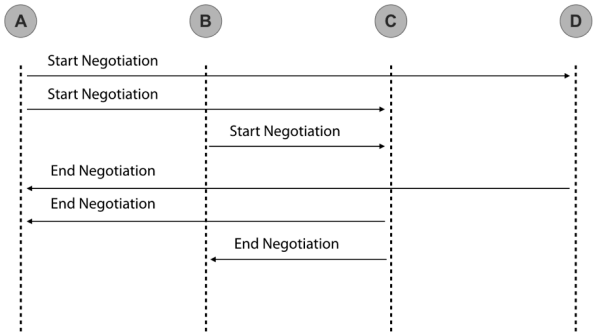

Fig. 10 – SE*link* multi-session negotiation

Similar to the SE*file* technology, SE*link* can be easily combined with third-party solutions to generate secure application and services. For example, SE*link* can be used in combination with XMPP based messaging applications in order to protect text and attachments.

The SE*link* stateless nature makes it suitable for operations on two or many channels even when they are related to the same service. For example, in case of VoIP services, SE*link* can start the negotiation procedure on the signaling channel. After successfully completing the process, the communication stream (e.g., voice) can be encrypted, signed, decrypted, and verified on the data channel, which is logically (and sometime physically) different from the signaling one.

7 Conclusions

This paper introduced some of the SEcube™ platform based methodologies to protect both data at rest and data in motion, which covers any security services and solutions.

It is possible to combine flexibility in the choices of technology and in the level of security to be attained with a well-organized and modular protection of data at rest and data in motion. As described in [2] and detailed in [3] for the software development platform, in [4] for the hardware and cryptography, and in [5] for the knowledge management via properties, also in security, the overall platform is developed to be robust and versatile, espousing the principle of models for encapsulation and model assembly for application development.

This approach delivers both the desired cost containment and the change friendliness.

Technologies like SE*file* and SE*link* provide a flexible security layer which can be deployed through several devices (multi-device) according to the target environment. The SEcube™ platform can be easily integrated in pre-existing systems thanks to a simple set of APIs that provide several entry points (multi-level). The internal implementation can be tuned (multi-paradigm) in order to match requirements of security, costs and user experience choosing among several options: from a standard and software only implementation to a full-custom hardware solution.

In the previous sections we described how to manage several use cases by means of the SE*file* and SE*link* technologies individually. Nevertheless, their combination is even more effective, since it is possible to address complex scenarios like remote machines and web based services, providing a large protection layer against server, client and communication side attacks.

All the above features make the SEcube™, and its associated abstraction layer and libraries, a unique security platform, which provides a multi-level, multi-device, multi-paradigm solution to realize high-grade security services and applications minimizing the development effort and reducing drastically the time-to-market to secure products.

8 Acknowledgment

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie)

9 References

[1] A. Varriale, E. I. Vatajelu, G. Di Natale, P. Prinetto, P. Trotta, and Tiziana Margaria, “SEcube: An Open-Source Security Platform in a Single SoC,” Proc. 11th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS 2016), April 2016, Istanbul, Turkey

[2] A. Varriale, G. Di Natale, P. Prinetto, B. Steffen, and T. Margaria, “SEcubeTM: An open security platform: General approach and strategies,” Proceedings of the International

Conference on Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), July 2016, in press.

[3] S. Boßelmann, J. Neubauer, S. Naujokat, and B. Steffen, "Model driven design of secure high assurance systems: an introduction to the open platform from the user perspective," Proceedings of the International Conference on Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), July 2016, in press.

[4] G. Di Natale, A. Carelli, P. Trotta, and T. Margaria, "Model driven design of crypto primitives and processes," Proceedings of the International Conference on Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), July 2016, in press.

[5] G. Airò Farulla, M. Indaco, A. Legay, and T. Margaria, "Model driven design of secure properties for vision-based applications: A case study," Proceedings of the International Conference on Security and Management (SAM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), July 2016, in press.

[6] Ali Josè Mashtizadeh, Andrea Bittau, Yifeng Frank Huang, David Mazières, Stanford University, Ori File System Web Site, http://ori.scs.stanford.edu/

[7] Rajesh Kumar Pal, Indian Institute of Technology, Secure File System Thesis

[8] T. Dierks, E. Rescorla, Network Working Group, The Transport Layer Security (TLS) Protocol Version 1.2, https://tools.ietf.org/html/rfc5246