Published in the proceedings of the Fifth Workshop on Enabling Technologies, (WET ICE '96), IEEE Computer Society Press, 1996.

SKIP — Securing the Internet

Germano Caronni[1], Hannes Lubich[2], Ashar Aziz[3], Tom Markson[3], Rich Skrenta[3]

[1] Computer Eng. & Networks Lab, ETH Zürich, email: gec@acm.org
[2] Swiss Academic and Research Network, SWITCH Zürich, Switzerland
[3] Internet Commerce Group, Sun Micro Systems Mountain View, CA

Abstract

Currently, two different approaches are being pursued for securing the Internet with respect to commercial use on a broad scale. The properties of these two approaches – application coupled security vs. network coupled security – will be discussed and compared. We will then focus on SKIP as an example for network coupled security, and show how it can be used to provide easily upgradable 'plug & play' cryptographic security. Providing upgradable security is an ideal base for the employment of organically growing security infrastructures, whose members still need to communicate with unsecured peers.
Keywords: Network layer security, IP security, SKIP, dynamic security, flexible up- and downgrading.

1 Introduction

The Internet is growing up. Due to strongly increasing commercial usage, new problems have come into the focus of its different user communities. The net is no longer a playfield for academic research and related user groups, but an area where significant amounts of money can be made — and lost.

To satisfy the strong and urgent need for security in the Internet, a multitude of security solutions (such as "ad hoc" packet filtering and firewall methods) are being deployed[1]. However, these solutions are mostly isolated, and no interoperability is foreseen for the conceptually different approaches. Some of the isolated solutions work in close cooperation with applications, some are coupled to the operating system or are even embedded in the link layer. In this paper, we will concentrate on a very generic solution which introduces security on the network layer and tries to give a foundation for an unifying solution. Our main goal is to show how authentication and encryption can be deployed throughout the Internet in a scalable manner. This will be done by expanding the functionality of the key management protocol 'Simple Keymanagement in IP' (SKIP)[2], which is one of the efforts at network layer key management currently pursued in the Internet Engineering Task Force (IETF).

The paper first discusses the merits of providing security coupled to the application or the network. A concise description of the inner workings of SKIP will then lead to an explanation of a method to dynamically up- and downgrade security properties of a communication relation. At the same time advantages and inherent vulnerabilities of this method are presented.

2 Network and Application Coupled Security

Current approaches to provide communication security to the users of the Internet can be separated into two broad classes. One provides authentication and encryption closely coupled to the secured application, the other does this (possibly for a whole end system at once) by being integrated into the operating system, e.g. in the network or transport layer. This section collects the arguments for and against both solutions, and explains why we choose to pursue the approach of network coupled security.

Application coupled security usually resides in the application itself, or in the transport layer of the ISO/ OSI reference model[3] It should not be mistaken as application layer security. Network coupled security usually resides within the network layer and may be placed in the operating system in question, or a higher layer in the system architecture. The main distinctive property of the two forms of security is the place where they are located within an end system. The coupling with the ISO/ OSI layers is weak, as Figure 1 clarifies.

Applications or Libraries **A**	Application	FTP/SMTP/RPC
	Presentation	XDR
	Session	
Core System (Kernel) **N**	Transport	TCP/UDP
	Network	IP
	Link	Ethernet /ATM
External HW	Physical	

A=Appl. coupled **N**=Network coupled

Figure 1 Layering vs. place of integration

Application coupled security may be realized by dedicated applications such as 'secure shell'[4] and encrypting digital telephones such as 'Nautilus' or 'Speak Freely'.

Extensions which fit closely under an application, such as secure RPC, encrypting telnet, or, best known, the 'secure socket layer' (SSL)[5] also provide this type of security. Application coupled security solutions usually demand that the application be security-aware, handles special conditions or errors, and provides proof of authenticity (certificates) and keying material.

In contrast to application coupled security, as presented above, network coupled security is placed within the operating system itself, usually within the network layer, and thus does not depend on reliable transport mechanisms. In certain scenarios, it may also be placed on intermediate systems. The most popular network layer is the Internet Protocol (IP), which is undergoing a major redesign[6] under the supervision of the Internet Engineering Task Force (IETF). The IETF forum is pursuing the goal to provide secure networking infrastructure in the IP security (IPSEC) working group that has been founded in late 1994. Some months ago, IPSEC has turned out proposed standards as to what the security framework in IPv4 and IPv6 will look like (RFC 1825, see [17]) and how authentication and encryption are to be done (RFC 1826 – 1829). There are first implementations offering this service, both under IPv4 and IPv6.

The currently pursued approaches on the network layer of the Internet Protocol (IP) suite extend IP packets by the new building blocks 'Authentication Header' (AH) and 'Encapsulating Security Payload' (ESP) which are used to secure payload and headers. These mechanisms assume that a shared secret is available to the two parties that want to communicate. How the shared secret is established is a problem of key management, which will be addressed in section 3.

2.1 A Discussion of Merits

Application coupled security is very sophisticated and certainly achieves all needs for private and reliable communication of applications. Nevertheless, there are substantial incentives not to bind security tightly to the application environment, but have it reside in the operating system, e.g. on the network layer itself. Advantages and disadvantages of these two classes of solutions will now be discussed.

Advantages of Application Coupled Security:

- The application is most able to decide what data falls under what security constraints. In a classical system this requires the application to be specifically designed security aware, in dynamic and configurable approaches (e.g. see [7]) this can be achieved by specifying the security relevant application requirements. Depending on the type of data that is to be transmitted, the appropriate actions can be taken.
- Assuming that the application carries out all security-relevant operations within its own environment, there is no need to invest any trust in a lower layer or independent components.
- No keys have to be transferred to lower layers, the visibility of this critical information can thus be most limited.

Disadvantages:

- By introducing security infrastructure closely coupled to the application, usually in the application itself, or in a library that is linked to it (eventually at run-time), the behaviour of this infrastructure is susceptible to influences from the application. Thus, components of the application have to be re-implemented, or verified, to assure that the system offers the promised security. An application that wishes to circumvent security relevant requirements of the environment can do so, which may be a risk in certain scenarios. Assuring the integrity of software components also becomes an issue here.
- Keeping the employed algorithms, certificates and revocations synchronized in a growing infrastructure of different security architectures can lead to wrong configurations. Additionally the issue of interoperability for independently developed but interoperating applications arises.
- If no external services at all shall be used, key storage (and management) has to be supplied by the application. This may cause problems. e.g. concerning scaleability.

Advantages of Network Layer coupled Security:

- Mechanisms and interfaces are equal for all applications on one machine. After correctness and security of cryptographic elements have been shown, no additional work (conceptual/ implementation) is needed to use them in other applications.
- Applications and users utilizing the network layer may access distributed key databases transparently, ideally without the application even notifying it. This also allows for a wholly transparent encapsulation of security and key management mechanisms.
- Providing security on the network layer, allows for intermediate systems such as routers and firewalls to perform (additional) operations on the communicated data. This allows an intelligent implementation of virtual networks and 'soft' link encryption.

Disadvantages:

- If security is done in the network layer, the application has to trust this layer unconditionally.
- The encryption of data always is always done the same way for one specific communication channel, not depending on the actual meaning of the data. A part of the flexibility available on the application layer is lost.
- A per-user key is more difficult to realize, and adds to the complexity of the communication infrastructure. When network layer security is employed, the layering principle is being violated implicitly when user identities are taken into account on this layer.

One has to remember, that on today's widely deployed machine architectures, the components of the system that control the network layer are also able to dominate the application. This means that no application is able to hide information from another application or user with special privileges. The arguments for keeping security near to the application to reduce the amount of required trust are considerably weakened by these facts. In the end it is just a question of expense an enemy with systems privileges on a machine has to go to to access the data that is to be kept secret.

To achieve a centralisation of the infrastructure and a unification of services (interoperability) it pays to integrate the security mechanisms with the operating system. This provides authentication and encryption to existing applications in a transparent manner, and without incurring the risk that future applications circumvent the system wide security requirements. On the one hand we observe a system which is structurally more secure, and allows for transparency, but on the other hand by coupling security to the application, either in the application layer or in the network layer, better control over what actions have to be performed on which data is granted to the security-aware application itself.

2.2 Proposed Architecture

We believe future solutions that provide communication security will reside partly in the application space, and partly near the network layer. This is not an 'either/or' decision, but rather an 'and', actually traversing and encompassing multiple layers. Bulk data cryptographic operations are to be performed on the network layer, which is usually tightly coupled with the rest of the operating system and thus allows for efficient employment and scheduling of potentially available dedicated hardware, random number sources and so on. At the same time, system wide policies that have been defined by a site administrator can be enforced without individual users being able to circumvent them. This has to be supplemented by an API and other utilities residing in the application space which allow to define the policy which the communicating application itself wants to enforce, and which provide the application with information about the level of authenticity a connection actually has, and what algorithms for data encryption and authentication are being used. Figure 2 depicts possible segmentation of and the relations in such an architecture. In the foreseeable future, applications on widely deployed platforms will have to hope for system integrity, as no provably secure environments exist for the general user.

While a part of this architecture can be realized by employing AH and ESP in the IP protocol stack on the network layer, the abstract database which provides keys and policies is realized by key management approaches.

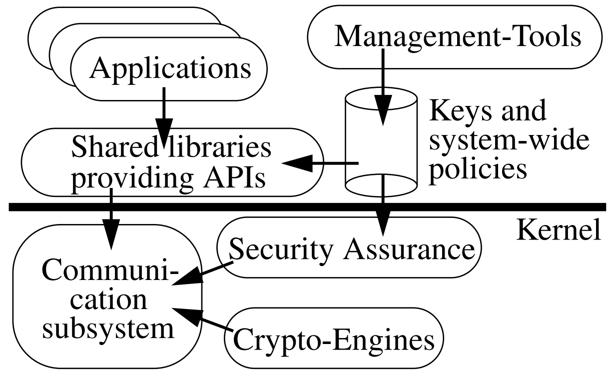

Figure 2 Possible placement of components

3 Key Management using SKIP

One problem not depending on the placement of the security mechanisms is key management. Somehow, an infrastructure has to be available that allows communicating peers to verify their respective identities, and to establish a shared secret used for bulk data encryption.

In this section, we will discuss one key management approach dedicated to the network layer mechanisms. The primary goal of security on the IP layer is to achieve private and authenticated communication between end systems, or alternatively between complete networks via firewalls. The employed system is structured such that the existing (and appreciated) properties of IP like the ability for e.g. re-routing, load balancing and crash recovery are maintained, and only a minimal overhead is being introduced in order to secure the network traffic. At the same time, the system is flexible and extensible enough to allow for future extensions like per-user or per-port keying, support of smartcards, secure multicasting, migration to IP v6, addition of new algorithms, etc.

Currently, two different approaches in key management (Oakley[12] and SKIP) are being pursued by the IETF working group on IP security (IPSEC). Although they offer a common subset of functionality, their fundamental approaches of solving the key management problem are vastly different. As SKIP is used to realize the mechanisms proposed in the following section, we will now elaborate on the key management and certificate discovery properties of it. For further discussion and comparison please refer to [8].

'Simple Key Management in the Internet Protocol' (SKIP) relies on the fact that a public certificate (signed e.g. by a certificate authority, or by oneself and other trusted entities using the PGP[10] infrastructure), is somehow provided to the communicating peer. No context is needed during the lifetime of a communication association (other than for efficiency reasons), thus switching from one host to another (as long as both hold the same secret part of the certificate) can be done easily. As SKIP does not reintroduce the notion of hard states below of the stateless IP layer, the establishment of a shared secrets is designed to happen implicitly. Each participant accesses a certificate, which contains the public value of the peer, by fetching it from a database or retrieving it via the built-in certificate discovery protocol (see below).

The acquired public value of the communication peer is combined with the system's own secret value using the Diffie-Hellman scheme[9], thus resulting in the same, shared secret on both sides. As the shared secret should be usable for a long time (it might be expensive to create a new correctly signed certificate) it is not used to encrypt data directly. Instead, a random traffic key is generated and used to encrypt data, and this random traffic key is encrypted with the long-lived shared secret, using a symmetric algorithm. In-band communication is used to transfer algorithm choices and the bulk traffic keying material.

SKIP certificate discovery is the most scalable way in which communication peers can retrieve the certificates of each other. The certificate discovery protocol (CDP)[13] allows for requests to be made using out-of-band communication, and is most efficient if a simple request-response protocol can be devised to provide sufficient data to the peer to establish a shared secret. The protocol actually allows for independence from servers of any kind, as long as the received certificate contains a signature from a trusted entity. After briefly covering some limitations of the key management aspect in SKIP, we will elaborate on how the CDP can be used to provide for a large deployment of secure communication infrastructure in the Internet.

It is the duty of the key management process that provides keying material to the network layer to allow the user to discover the actual 'quality' of the security that has been achieved in talking to the peer. The keying material used by the hosts that follow the scheme presented in section 4 may be authenticated. But new hosts, that are not embedded in a larger organisation, and just acquired the capability to run in secure mode usually do not own such authenticated keys. They will use dynamically generated and unsigned public values instead. The dynamically generated DH public values should not be authenticated by on-line certification authorities, and are thus not bound cryptographically to an identity. They offer weak security only, and are similar to the public values that are presented in [16].

4 Large Deployment

By using network layer security mechanisms, security can be deployed throughout the Internet without making applications security-aware. Users of different communities may use SKIP to fulfil their needs, as long as the configuration of SKIP represents their policy.

A high degree of importance should be placed on 'plug & play' capabilities for security in the network layer. An average user who upgrades his machine to secure networking wants to incur as small an amount as possible of overhead and 'new' problems. Thus his machine should be able to switch into 'privacy' mode without having to own a certificate, and the user or administrator should be allowed to easily define (and control) the behaviour of his machine.

In view of increasing emphasis on providing private and authenticated communications between hosts, the easiest solution would be to require that hosts secure all traffic, and also expect all received communications to be secured. However, most hosts do not yet have the capabilities to perform secure communications, thus a host enforcing this policy would loose its ability to communicate with most other hosts in the Internet.

It is desirable to have a scheme to allow the dynamic detection of security capabilities of a host, and instigation of the use of encryption and authentication if such capabilities are detected. The following scheme allows a host administrator to specify a security policy that dictates: "Require encryption to host A. Always speak in the clear to host B. Other hosts may connect in the clear, but use security mechanisms if both sides provide them."

This leads to an immediate consequence: To allow for a widespread adoption of encryption and authentication on the network layer, key management protocols and 'policy engines' need to allow three types of connections, and need to have the ability to upgrade, and eventually downgrade, the current level of security. The three types of connections are 'clear', 'secure', and 'optionally secure'. Each host assigns its peers to one of the classes.

As users indulging in secure communications want to incur minimal overhead while a clear connection is upgraded to the optionally secure mode, data should (optionally) be exchangeable right from the start. An active Internet host exchanging traffic with many other sites will not be willing to endure delay for initial connections and noticeable initial delay would hamper its performance and thus the use of security protocols such as SKIP. Even though the discovery of whether encryption and authentication may be used takes some time, the scheme must not cause noticeable delays, before the users data is sent.

The scheme also has to correctly recover from states where a host ceases to handle encryption (e.g. the host looses its secret key or boots another operating system that does not provide security). In such an event, optionally secure communication must be able to fall back to clear communication.

```
record is
    remote IP address
    remote public value
    own public value
    own secret value
    auxillary data
    idle time
    max. idle time
    age , usage  , mode
end record
```

Figure 3 Cache record

The behaviour described in above paragraphs can be easily obtained by letting packets flow in the clear initially (this should be optional, depending on site configuration), while simultaneously attempting to complete a certificate exchange using the CDP with one or more target hosts. If a certificate exchange is successful, both sides can begin to encrypt and/ or authenticate the packets that are sent. Capabilities of peer hosts are cached,

Figure 3 represents possible contents of such a record. Here ‘age’ is the age of the record, ‘usage’ represents the intensity of the traffic, and ‘mode’ is in this case either ‘clear’, or ‘opt. secure’. ‘Max. idle time’ is variable (see below) and if the actual ‘idle time’ reaches this limit, the record is discarded.

During normal operation, a ’clear traffic’ record is created, allowing the remote hosts to send and receive clear packets. At the same time, one of the hosts will initiate a certificate discovery for the other side (see Figure 4). Only one certificate discovery is initiated between two hosts, as long as no remapping of IP addresses takes place by intermediate systems.

```
initialize is
    find record
    if (no record) then
        create ‘clear’ record
        if (own IP < remote IP) then
            initiate CDP PUSH/GET
        end if
    end if
    return record
end initialize
```

Figure 4 Create record

Figure 5 and Figure 6 represent this behaviour in algorithmic fashion for the receiving and the sending side. If the certificate discovery is successful, both parties will upgrade their communications to the default optional security settings and change the cache entries to require encrypted and authenticated traffic.

```
clear IP received is
    record := initialize()
    if (record is ‘clear’) deliver
    if (record is ‘opt. secure’) then
        if (age of record < e.g. 20”) then
            deliver
        else
            discard
        end if
    end if
end clear IP received
```

Figure 5 Handle clear input

While secured traffic flows, this setting stays valid. If traffic ceases to flow, and thus the cached record expires after a while, it is removed. It has become questionable whether the former peer is still able to run in encrypted mode. The same holds true if ICMP messages of type ’Protocol unreachable’ are received from the host, or the host begins to send data in the clear. It should be left to the administrator of the site which of these indications he wants to trust that the former peer lost its capabilities for secure communications. We suggest using a maximum idle time in the order of 5 minutes, and honoring the ICMP message unless secure traffic has been received within the last minute, but logging all downgradings of security properties in a visible manner. We also suggest to ignore clear data that is received, as in that case the cache expiry mechanism will provide for a delayed fallback in any case. Figure 7 represents an algorithmic representation of the cache aging process, which is invoked e.g. every 5 seconds. In this embodiment, a separate maximum idle time can be dynamically calculated for each host, based on factors such as the location of the host, analysis of past communication history with this host, etc. By updating these values on the fly, more efficient behaviour is achieved.

```
outgoing IP is
    record := initialize()
    if (record is ‘clear’ ) deliver
    if (record is ‘opt. secure’) then
        process and deliver
    end if
    if (record is ‘CDP in progress’) then
        queue packet for 10”
        OR until CDP complete
        process and/or deliver
    end if
end outgoing IP
```

Figure 6 Handle output

It should also be possible for the administrator to specify a behaviour where no downgrading occurs, and cache entries that are created for secure communications are kept alive indefinitely. This would allow clear sites to upgrade, but disable the automatic fall-back. Once a host used encryption, manual reconfiguration is needed to allow this host to communicate in the clear.

```
cache aging is
    for each record do
        increment idle time and age
        if (idle time > max. idle time) then
            remove record
        end if
    max. idle time of record :=
            f(idle time , usage , age , ...)
    done
end cache aging
```

Figure 7 Cache aging

If a host receives a secured IP packet from a peer for which it does not hold a cached entry specifying its capability to do so, the host should initiate a certificate discovery, and upgrade to optionally secure mode. Such a condition arises form e.g. a natural time-out of a cached entry when no communication took place for a longer time. Figure 8 describes in detail how to proceed if a secure IP packet is received.

```
secure IP received is
    find record
    if (no record) create ‘clear’ record
    if (record is ‘clear’) then
        initiate CDP PUSH/GET
        queue packet for 10”
        OR until CDP complete
    end if
    if (record is ‘opt. secure’) then
        if (own public value
                as used by peer is
                obsolete or unknown) then
            CDP PUSH new value
            discard packet
        else
            process and deliver
            idle time := 0
            increment usage
        end if
    end if
end secure IP received
```

Figure 8 Handle secure input

Although the security in the ‘large deployment’ scheme is susceptible to ’man in the middle’ attacks, and several active ’denial of service’ attacks, the scheme allows hosts to encrypt and authenticate traffic which would otherwise travel in the clear. In this design, malicious interactions cause a wrong behaviour only on a per session basis, and not for all time. As much clear traffic as possible is encrypted to thwart the far more widespread passive attacks. As the traffic may not be upgraded to a secure connection until after some packets have been sent in the clear, an active attack may force the hosts to speak in the clear when they could be encrypting. If an active enemy is stubborn enough to pursue this at all times, he will be detectable quite easily.

Since the communication would otherwise be completely unsecured it is acceptable to provide a weak degree of security, as long as the users are aware of its limitations. And as soon as the user upgrades to an authenticated certificate, and makes it available to a broad community, the degree of security available to him rises dramatically. This flexible upgrading, we hope, will allow to make the Internet secure for everybody who cares.

5 Outlook & Summary

After giving a rationale for placing security mechanisms in the network layer, we have explained how SKIP can be used to rapidly deploy security throughout the Internet. Depending on the policies of the participants, and on the availability of operating systems that are capable of secure communications, ever larger parts of the Internet may thus become secure in an organic fashion. This is achieved by allowing the dynamic up- and downgrading of the security between two communicating peers. SKIP provides most of these capabilities.

Assuming that a PGP like infrastructure for public keys used in the network layer security packages will be available soon, the user can then employ the well known techniques of web-of-trust to grow into a secure environment. Users already embedded in an hierarchical environment can just add a certificate as it is provided by their CA. This flexibility is possible by three new messages (appertaining to different namespaces) to establish a shared secret in SKIP. The three messages are the traditional certificate, a message providing unauthenticated ephemeral key material, and, most valuable, a message providing authenticated ephemeral keying material and perfect forward secrecy (see [11] and [14] for details). The last message can only be used if the identity (and certificate) of the potential peer is already known, but is the most powerful of those taken into consideration, as it provides both authenticity for the potential peer, and anonymity against passive attackers in one single message. These three types of messages can be combined into two-way exchanges, providing a series of small protocols, each with valuable properties.

Other issues for further study are how to provide keying material in an efficient and scalable manner to the members of a large multicast group, and how to valuate the use of key rings as they are provided by the PGP infrastructure. [15] addresses the theoretical aspect of ‘web of trust’, and a related efficient and practical solution has to be found.

Availability

SKIP is readily available in source form for multiple platforms in the whole world, and performs well in pilot installations, either using manual keying, a local X.509 certificate infrastructure, or supporting a chipcard offering securely authenticated and ephemeral keying.

Source is available: (international source release) ’http://www.tik.ee.ethz.ch/~skip’ and (in the U.S. of A. only) ’http://skip.incog.com’.

Acknowledgments

The authors express their gratitude to Bernhard Plattner for discussions which strongly influenced our work, Burkhard Stiller for his patient and persistent reviewing and valuable feedback, Michael Hauber and Christian Schneider for actually implementing what we were just writing about, and to the anonymous reviewers whose comments we value highly and which helped us to create a better final version of this document.

References

[1] W. R. Cheswick, S. M. Bellovin, “Firewalls and and Internet Security”, Addison-Wesley, 1994.

[2] A. Aziz, T. Markson, H. Prafullchandra, “Simple Key-Management For Internet Protocols”, Internet draft, work in progress, December 1995.

[3] “The IS/OSI Reference Model”, ISO standard 7489 and CCITT standard X.200.

[4] T. Ylonen, “SSH Transport Layer Protocol”, Internet draft, work in progress, June 1996.

[5] A. O. Freier, P. Karlton, P. C. Kocher, “The Secure Socket Layer Protocol v3.0”, Internet draft, work in progress, March 1996.

[6] S. Deering, R. Hinden, “Internet Protocol, Version 6”, Request for Comments (Standard) RFC 1883, Internet Engineering Task Force, December 1995.

[7] D. Bauer, G. Caronni, C. Class, C. Conrad, B. Stiller, M. Waldvogel, “The Da CaPO++ Project”, internal report, Swiss Federal Institute of Technology Zürich, April 1996.

[8] G. Caronni, H. P. Lubich, “Proposed Security Mechanisms in the ‘New’ Internet”, SWITCH Journal nr. 1/96, May 1996.

[9] W. Diffie, M. E. Hellman, “New Directions in Cryptography”, IEEE Transactions on Information Theory, vol. IT-22, pp. 644-654, November 1976.

[10] P. R. Zimmermann, “The Official PGP User’s Guide”, MIT Press, 1995.

[11] W. Diffie, “Authenticated Key Exchange and Secure Interactive Communication”, Proceedings of SECURICOM ‘90, 1990.

[12] H. K. Orman, “The OAKLEY Key Determination Protocol”, Internet draft, work in progress, May 1996.

[13] A. Aziz, T. Markson, H. Prafullchandra, R. Skrenta, G. Caronni, “Certificate Discovery Protocol”, Internet draft, work in progress, June 1996.

[14] A. Aziz, “SKIP extension for Perfect Forward Secrecy”, Internet draft, work in progress, Februar 1996.

[15] U. Maurer, “Modelling a public-key infrastructure“, Proceedings of ESORICS ‘96, 1996.

[16] A. Aziz, T. Markson, H. Prafullchandra, “Encoding of an Unsigned Diffie-Hellman Public Value”, Internet draft, work in progress, December 1995.

[17] R. Atkinson, “Security Architecture for the Internet Protocol”, Request for Comments (Standard) RFC 1883, Internet Engineering Task Force, August 1995.