A new Perspective for e-business with SSL/TLS*

Ibrahim Hajjeh[1], Ahmed Serhrouchni[1], Frédérique Tastet[2]
[1] Ecole Nationale Supérieure des Télécommunications
46, Rue Barrault, 75013 Paris, France
{hajjeh, ahmed}@enst.fr
[2] Thales Communications
66, Rue du Fossé Blanc, 92231 Gennevilliers, France
{Frederique.tastet@fr.thalesgroup.com}

Abstract — **SSL/TLS is a security protocol aimed to secure all data exchange. It was primarily conceived to offer an encrypted channel with application server authentication. The applications for e-business or even education have more specific needs such as identification, authentication or non-repudiation.**
Other solutions have since been born such as IPsec (IP Security) and in more specific way the protocol of session establishment ISAKMP (Internet Security Association and Key Management Protocol). This protocol opens new perspective for secure sessions for all network layers.
We thus integrate this protocol with SSL/TLS in its session establishment phase. This makes it possible to extend the work of SSL/TLS to support new services such as those necessary to e-business applications.

Keywords — **E-business, ISAKMP, SSL/TLS.**

I. Introduction

Actually SSL/TLS[3] (Secure Socket Layer / Transport Layer Secure) [1] is the most deployed security protocol. This is due, mainly, to its native integration in browsers and web servers. In addition, SSL is light and rather easily implemented in applications. SSL is more than a protocol; it is made up of four modular and independent protocols, of which the Handshake protocol has to authenticate the parts concerned, negotiate security algorithms and generate a shared secret necessary to guarantee the services of integrity and confidentiality.
Actually, applications become more and more exigent in security needs and thus an evolution of security services provided by SSL is required. The future extensions of SSL are easy to be integrated by the modular nature of the protocols that composes it. The target of these extensions is mainly the Handshake protocol, therefore we focused on an analysis to extend the SSL protocol rather than to modify it. Among several Key Management protocols, we think that ISAKMP [2] can replace the SSL handshake protocol and this for three reasons:

1. Its capacity to offer a large scale of security services.
2. Its capacity to unify security solutions on different level of the network stack.
3. Its robustness

We then propose a new architecture for SSL with ISAKMP to ensure, amongst other things, the service of authentication and non-repudiation. Our objective is also to keep the SSL handshake protocol for interoperability reasons.
In the following section of this paper we will describe several key management protocol including ISAKMP and SSL Handshake, comparing their different mechanism in security communications and clarifying their advantages and disadvantages. In section three, in order to put our contribution in context we will explain what motivated us to integrate ISAKMP in SSL with the proposed new architecture. We also present the process of integrating new services within our architecture. To conclude we propose an analysis of this solution and its prospects, in particular in experimentation and deployment.

II. Key Management Protocols

The first basic functionality of a Key Exchange protocol is to provide two communicating parties with a shared secret that is known only to them. Key exchange protocol differ in their scenarios of exchanging keys. (e.g. public key method, key distribution center,…). Some of them use a combination of different methods. The result of a key exchange is a master key or session key depending on the life span or the semantics of applications. We will often refer to the exchanged key as a "session key".
Even though the Internet Architecture Board (IAB) has not yet agree on a key management architecture among several existing alternatives, the current work in this area is likely to converge toward a combination of two protocols: the Internet Security Association and Key Management Protocol (ISAKMP) and Oakley key exchange protocol. This is because ISAKMP guarantees several required characteristic like Perfect Forward Secrecy (PFS), Direct authentication, back traffic protection and identity protection [3]. In this

* Work Partially supported by the French program RNRT (Réseau National de Recherche en Télécommunications) under the Icare project (Infrastructure de Confiance sur des architectures de Réseaux Internet & Mobile)
[3] TLS protocol is based on SSL 3.0 Protocol published by Netscape. "No dramatic" difference exist between these 2 protocols. In this paper we mean by SSL, SSL/TLS.

section we will describe some key exchange protocol accepted in different IETF working groups. We will present their main characteristics in a comparison table (table 1).

A. Photuris

1. Introduction

Photuris is a session key management protocol designed by P. Karn from Qualcomm and William Simpson from DayDreamer [8]. Photuris was described in different drafts and RFCs, and is intended for use with the IP Security architecture such as AH (Authentication Header) [7][4] and ESP (Encapsulating Security Payload) [5].

2. Photuris Architecture

Photuris is based on a shared secret configured with Photuris implementation. This secret is replaced automatically with a short-lived session-key necessary to protect data communication.

Photuris consists of several simple exchanges (Figure 1).

1 – A "cookie exchange" to guarantee a weak authentication against DOS attacks.

2 – After, a "value exchange" to establish a shared secret between the two communicators

3 – Lastly, an "identification exchange" to identify the 2 parties with each others. Photuris is independent of any particular party's identification method or certificate format.

In addition, other types of security attributes could be exchanged, for example to exchange session keys or to establish new security parameters.

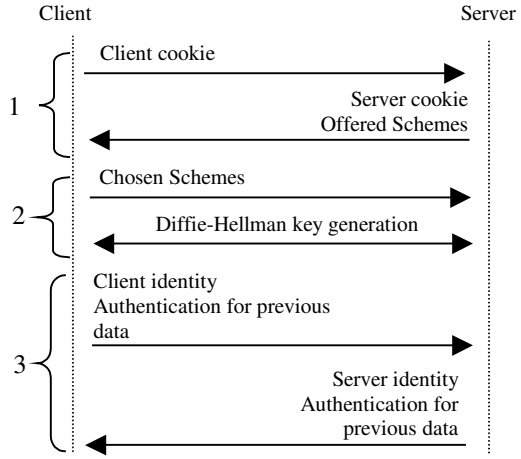

Figure 1. **Photuris key Exchange**

In summary, the primary use of Photuris is for mobile clients, multi-cast applications and network operating over limited bandwidth. This is due mainly to the different types of Photuris exchange related to separate Security Parameters Index SPI lifetimes, speed of links and fast cookie generation and verification.

B. SKEME

1. Introduction

SKEME (Secure Key Exchange Mechanism) [9] is a key exchange protocol developed in the IPSec Working Group WG of the IETF but still not suggested as an internet draft.

SKEME has many similarities to Photuris, which is being developed through the same working group. SKEME provides the basic functionality of Photuris like performing a Diffie-Hellman exchange based on public keys.

In addition SKEME provides various modes of key exchange e.g. with KDC (Key Distribution Center) or manual distribution.

2. Architecture

In short SKEME provides four distinct modes:

- basic mode, which provides key exchange based on public keys and PFS thanks to Diffie-Hellman.
- A key exchange based on public keys, but without Diffie-Hellman.
- A key exchange based on the use of a pre-shared key and on Diffie-Hellman.
- Fast re-keying mechanism based on symmetrical algorithms.

SKEME is based on three basic phases: SHARE,EXCH and AUTH.

1 - In the SHARE phase, the two entities exchange "half-keys", encrypted with their respective public keys. The shared secret is the combination of the two half-keys via a hash function. If anonymity is wanted the identities can also be encrypted (the entities can include the public key certificate of each entity). In short this phase guarantees that only the two entities know the shared secret.

2 - The second phase, EXCH, is used to exchange Diffie-Hellman parameters or even a nonce.

3 - In the AUTH phase, the Diffie-Hellman values exchanged are authenticated using the shared key from the SHARE phase.

We should add that the messages in the three different phases could be combined and the order could be changed, for example the ordering of messages in EXCH can be reversed, as is the case for messages in AUTH. The combined messages in SKEME can be used for a fast and full exchange with a uniform and compact format.

Finally, SKEME has many advantages. It is totally independent of DH key exchange, it uses public key encryption to exchange a one time key and then uses shared-key techniques to authenticate the DH exchange. With this method, one can skip the DH phase and still have a key exchanged between the parties. This is specially relevant to the application where the protocol is used mainly for authentication (not encryption).

C. SKIP

1. Introduction

SKIP (Simple Key Management for Internet Protocols) has been created by A. Aziz, T. Markson and H. Prafullchandra [12] and accepted as an internet draft in the IP Security WG of the IETF in 1996. SKIP unlike most other key management protocol situated in the application layer, is a network layer protocol based on the generation of a shared secret using authenticated Diffie-Hellman public value.

2. Architecture

SKIP is a network layer protocol created to achieve private and authenticated communication between the two communicators in a large network without security applications being aware. Because all network layer protocol are integrated into operating systems, SKIP provides an easy "plug and play" capabilities to upgrade all security mechanisms.

To begin a SKIP connection, the two end entities should have an authenticated public key (ex. Public key signed by a CA (Certificate Authority), or trusted entities such as a PGP certificate). After the two entities should exchange their public key stocked in a Database repository like LDAP directory. A long life shared key is created using Diffie-Hellman scheme. This shared key is not used for data encryption but a short lived random key encrypted with the shared secret is used.

Because encryption and authentication are not mandatory in a public network, SKIP allows a host administrator to specify a security policy that describe the type of connections. In SKIP there are three types of connections: clear, secure and optionally secure.

Finally, even though SKIP was accepted as an internet draft in 1996, the IPsec WG has adopted ISAKMP/Oakley as a key exchange protocol with all IPsec implementation. This has stopped all future large deployment of this protocol and make application layer key management protocol more significant in communications needs in term of dynamic and configurable approaches.

D. Oakley

1. Introduction

Oakley is a key Exchange protocol which is suggested by Hillarie Orman from University of Arizona [10]. Oakley was described in several internet drafts within the IPSec WG of the IETF. As supposed to SKIP, Oakley is an application layer protocol that together with ISAKMP represents the essential mechanism adopted for key exchange with IPSec.

2. Architecture

Oakley is a key determination protocol by which the two entities can first be authenticated with a cookie exchange (using SKEME protocol), then agree on a shared secret with the Diffie-Hellman key exchange algorithm.

Oakley Support also PFS, identity protection, non-repudiation and user-defined group structures for the DH algorithm. Oakley allows also the management of security association for ISAKMP. The compatibility with ISAKMP requires Oakley to be used over the ISAKMP UDP port (port 500). In [10] Oakley in theory can be used directly over IP protocol if no combination with ISAKMP is used . In addition Oakley has the following characteristics:

- The addition of weak authentication for the two entities with a cookie exchange.
- Permit the two peers to agree on key derivation method, encryption algorithms and authentication method.
- Authentication in Oakley does not depend on encryption using the Diffie-Hellman exponentials.
- It is not necessary for the two parties to compute the shared exponentials.
- Oakley can derive a new key from an old key or distribute an external encrypted derived key. This means that Oakley lets the two entities to use all or some of the anti-clogging and PFS features.
- Oakley support different types of certificates such as PKCS#7, PGP certificate, Kerberos Tokens, X509 certificate,…

E. SSL

1. Introduction

SSL (Secure Socket Layer) was originally designed by Netscape Company to meet the occurring needs of Internet Security at that time. In March 1996 TLS (Transport Layer Security) was approved by the IETF as the standard internet secure protocol. SSL and TLS provide a generic channel security mechanism on top of TCP.

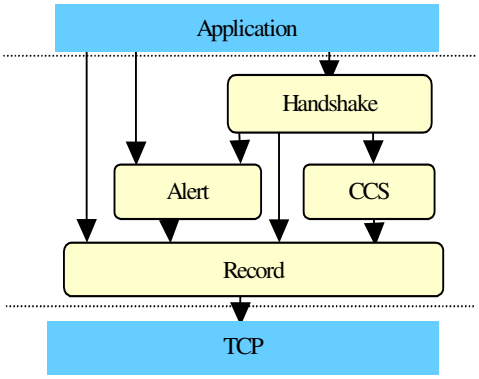

Figure 2. The SSL architecture

2. Architecture

The main goal of SSL is to provide server authentication, encryption and message integrity. Optionally the client can also be authenticated. The internal architecture of TLS is shown in [Figure 2]. SSL was designed in four modular protocols.

a) SSL handshake

The SSL handshake authenticates the server and optionally the client, negotiates cryptographic algorithms, exchanges keys and/or initialisation vectors.

Once this phase is finished, a protected connection is established.

b) SSL Change Cipher Spec (CCS)

The Change Cipher Spec consists of only one message which can be sent either by the client or the server to notify the other party that subsequent records will be protected under the newly negotiated cipher spec and keys.

c) SSL Record

The Record Protocol takes messages to be transmitted, optionally compresses the data, applies a MAC, encrypts and transmits the result. Received data is decrypted, verified, decompressed and then delivered to higher level clients. Moreover, the Record Protocol takes care of data integrity and authentication.

d) SSL Alert

If an error is detected the SSL Alert protocol in the detecting party sends an alert message containing the occurred error. The peer decides further procedure depending on the content type of alert messages. Alert messages convey the severity of the message and a description of the alert. There are three types of alert messages: warning, critical, and fatal.

The following figure (Figure 3) describes a general SSL handshake exchange.

1) In brief, to establish a secure connection SSL with a server authentication, the client sends its SSL information (SSL version number, cipher settings, random number,…)in a ClientHello message to the server.
2) The server sends its own SSL information(a cipher out of the client list, random number,…) with its digital certificate to the client. The server's certificate contains the server's public key used to authenticate the server with client.
3) The client verifies the server's certificate and extracts the server's public key. The client generates a random key called pre-master-secret, encrypts it using the server public key and sends it to the server. The client and server independently generate a master secret by computing client and server random number with the pre-master-secret.
4) The client's MAC function hashes all exchanged messages with the master secret and send them to the server. The client also send a message that the handshake is finished.
5) The server's MAC function also hashes all the exchanged messages and send it to the client. The server confirm that the handshake is finished.

If the client's certificate is required by the server, the server will send first a message that contains his certificate and then the client will send a *certificateverify* message including its signature on the hash value of the pre-master key combined with all past messages exchanged in the current session.

3. SSL Handshake Overhead

SSL was designed to provide server authentication to clients easily and efficient encryption negotiation for any application layer program. Nevertheless HTTP is the protocol the most frequently used with SSL and so it is so natural to think that HTTP runs inside SSL. SSL shows its advantage when performing many secure and small connections. Unfortunately it is non trivial to predict the exact result of SSL on movies, audio or even signature services. In fact SSL Handshake depends on critical variables such as:

- How many times must the handshake be performed?
- Which cipher suite is being used ?
- Which hashing method is being used ?
- How complex is checking certificate validity?

The organization's response was first to define a list of preferred key management, hash and cipher algorithms beginning with less overhead proposition. (e.g. RC4 encryption with 70 bit key and MD5 MAC) and second by the use of a resumed session to limit the expensive public key encryption to the first session only. However other needs in term of new service (e.g. identity protection, access control based on groups, dynamic policy and attribute certificate…) or other unresolved questions (e.g. X509 certificate and Certificate Revocation List CRL verification) are still ambiguous.

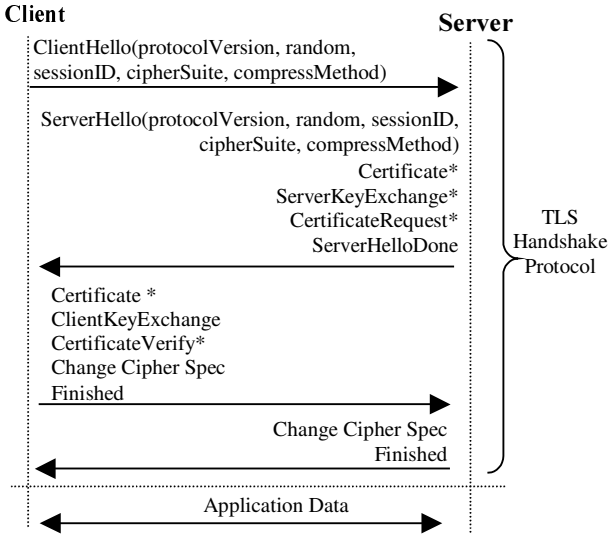

Figure 3. SSL handshake

*** indicates optional or situation-dependent messages that are not always sent.**

F. ISAKMP

1. Overview

The Internet Security Association and Key Management Protocol (ISAKMP) [2], defined in the RFC 2408 of November 1998, is a framework that defines procedures and packet formats for establishing, negotiating, modifying and deleting Security Association (SA). It also allows the two

peers to authenticate one another and to perform key exchange in a protocol and algorithm independent way.
The work on ISAKMP was initiated by the IETF in 1994. In parallel with ISAKMP work, other standards issue such as Simple Key-Management for Internet Protocols (SKIP), Secure Key Exchange Mechanism (SKEME) and Photuris, was being studied. Two years later, ISAKMP outperformed the others in terms of extensibility and generality, and was therefore adopted as the mandatory-to-implement key management protocol for IPv6. For IPv4 it was defined as optional.

2. Architecture

ISAKMP can be implemented over any transport protocol or over IP itself. Implementations must, however, support at least UDP. An ISAKMP Message consists of an ISAKMP header followed by a variable number of payloads. These payloads are the building blocks of ISAKMP.
Because ISAKMP does not impose anything on the parameters that compose the SAs, a document called Domain of Interpretation (DOI) [14] must define the negotiated parameters. The DOI plays an essential role in key management. DOI defines payload formats, exchange types and some security information such as security policies or cryptographic algorithms. There is also a DOI identifier used to interpret the payloads of the ISAKMP messages. For example, the IPsec protocol has number 1 as its DOI identifier. A list of the DOI is defined in [14]. A new DOI Identifier for SSL should be added.

ISAKMP comprises two phases which allow a clear separation of the SA negotiation for a specific protocol and traffic protection for ISAKMP**.**
- During the first phase, all attributes regarding the security are negotiated, the identities of the thirds are authenticated and the keys are generated. These elements constitute a first "Security Association", known as SA ISAKMP.
Phase 1 is concerned only with establishing the protection suite for the ISAKMP messages and does not establish any security associations or keys for protecting user data.
- The second phase makes it possible to negotiate the security parameters related to the SAs to another security protocol (for example IPsec AH, IPsec ESP, TLS, …) to protect user data exchanges. The exchanges of this phase are protected (confidentiality, authenticity,..) by the SA's ISAKMP established in phase 1. Phase 1 negotiations are executed once a day or maybe once a week but phase 2 negotiations are executed once every minute. [9].

ISAKMP allows creation of own exchange sequences for the establishment of security associations and keying material. There are five default exchange types defined in ISAKMP. These exchange types are illustrated in (Table 2).More Key exchange definitions can and often must be defined in DOIs to meet all the needs of security and key management protocols.

The *Base Exchange* is designed to allow the key exchange and authentication related information to be transmitted together. This minimizes the number of exchanges at the expense of not providing identity protection. Initiator and Responder exchange cookies, SA, Nonce and finally user-ID.
- The *Identity Protection* Exchange is designed to separate the key exchange information from the identity and authentication related information. This provides protection of the communicating identities because the user-ID is exchanged only in the last 2 encrypted messages.
- The *Authentication Only* is designed to allow only authentication related information, i.e. Keys for protecting further communication won't be generated. The benefit of this exchange is the computational expense loss when mutual authentication is performed without a key exchange.
- The *Aggressive Exchange* is designed to allow the security association, key exchange and authentication related payloads to be transmitted together. This exchange aims at minimizing the network traffic.
- The *Informational Exchange* is designed as a one-way transmittal of information. It is used for sending information about errors and SA deletion requests to the communicating party.

Table 2. ISAKMP exchange Types.
Notation: I: Initiator, R: Responder, *: payload encryption after the ISAKMP Header.

Base Exchange	**Payloads**
I→ R	Security Association (SA); NONCE
R→ I	SA; NONCE
I → R	Key Exchange KE; Initiator Identity ID(I); Authentication Payload AUTH
R→ I	KE; Identity Receiver ID(R); AUTH
Identity Protection Exchange	
I→ R	SA
R→ I	SA
I → R	KE; NONCE
R→ I	KE; NONCE
* I → R	ID(I); AUTH
* R→ I	ID(R); AUTH
Authentication Only Exchange	
I→ R	SA; NONCE
R→ I	SA; NONCE ; ID(R); AUTH
I → R	ID(I); AUTH
Aggressive Exchange	
I→ R	SA; KE; NONCE; ID(I)
R→ I	SA; KE; NONCE; ID(R); AUTH
* I → R	AUTH
I → R	N/D (Notification/Delete payloads)

3. Initialising SAs with ISAKMP

This section outlines how ISAKMP establish security associations and exchange keys for phase 1 negotiation. To provide a concrete example, we will describe the *identity protection exchange* (Figure 4) that will be used later in a business scenario describing SSL- ISAKMP exchange.
- ISAKMP messages themselves will be carried as UDP payload
- All ISAKMP messages begin with an ISAKMP header that contain the type of exchange, the cookie of the initiator, the cookie of the responder and the message Identifier.

Feature	Photuris	SKEME	SKIP	Oakley	ISAKMP	SSL Handshake
Layer	Application layer (over UDP)	Application layer	Network layer (over IP)	(Over UDP or over IP)	Application layer (over UDP)	Session layer (over TCP)
Workgroup	IPSec WG - IETF	IPSec WG – IETF	IPSec WG – IETF	IPSec WG – IETF	IPSec WG – IETF	TLS WG - IETF
Plate form & Environment	VPN, mobile users, limited bandwidth network.	Supported by any systems, routers.	Supported by any systems, routers.	Supported by any systems, routers.	Supported by any systems, routers.	Ideal for Client (web browser) / Server (web server) application
Performance	Fast cookie exchange adopted for many other protocol.	Similar to Photuris with various key exchange methods	Based on source and destination addresses, suitable for routers	Faster due to one handshake for the entire connections between the two entities		Handshake for each connection between applications
Security services	Weak authentication, encryption, anonymity,	Authentication, confidentiality, integrity, anonymity, anti-replay message	Authentication, encryption	Non-repudiation, identity protection, user-defined group, authentication	Authentication, confidentiality, integrity, non repudiation, traffic analyse, anti-replay	Authentication (server & optionally client), confidentiality, integrity, anti-replay
Authentication methods	Cookies Authentication, DH public keys exchange.	Public key with/without DH, Pre-shared key	Authenticated DH values with a Third Party (e.g. Certificate Authority)	Public key signature, public key encryption, shared key	Shared secret, public key encryption, public key signature Kerberos,	Public key, Diffie-Hellman, Fortezza
Identity Protection	Supported (with a shared key)	Not supported	Not supported	Supported	Supported (in phase 1 & 2)	Not Supported
Key management methods						
Diffie-Hellman	Supported	Supported	Supported	Supported	Supported	Supported
Key Distribution Center (KDC)	Not supported	Supported	Not supported	Supported	Supported	Not supported
Public key	Supported	Supported	Supported	Supported	Supported	Supported
Manuel distribution	Not supported	Supported		Not supported	Supported	Not supported
(PFS) Perfect Forward Secrecy	Supported	Supported	Supported	Supported	Supported	Not supported
Key Refreshment	Not supported	Supported	Not supported	Supported	Supported	Supported
Certificate type	Not defined	Not defined	X509, PGP	X509, PGP	X509, PGP, CRL, Attribute Certificate, …; 7formats	X509
Current state	Current work with IPSec protocol (used in Oakley protocol)	Remain without a draft description	Not widely used since 1996	The two protocols are used in IKE, the Internet Key Exchange Protocol adopted with IPSec.		Remain the large deployed security protocol in use today.

Table 1 - A comparison of Key Management Protocol features.

Legend:

Real advantage

Disadvantage

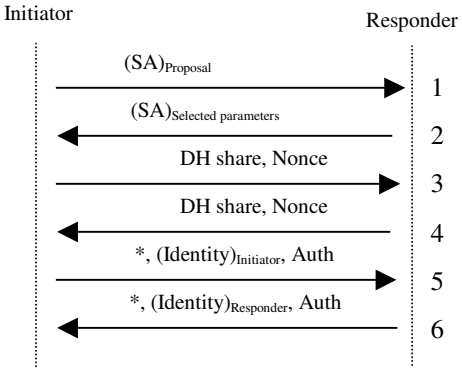

Figure 4. ISAKMP Identity Protection exchange.

Notation: *: all payloads after the ISAKMP Header are encrypted.

- In the first exchange, the initiator sends a DOI identifier (e.g., SSL DOI) and a list of proposed protocol (e.g. ISAKMP, AH…) with an associated identifier named *Security Parameters Index* SPI. Each protocol can name several acceptable transforms carried in a transform payload (e.g. RSA-HMAC-MD5, RSA-DES). The responder will indicate which one he will support. At this point the ISAKMP SA has been agreed by the two entities, the identity of the ISAKMP SA has been set to <$Cookie_{Initiator}$, $Cookie_{Responder}$> but it is not yet verified.
- In the second exchange, the two entities will exchange a two NONCE ($N_{initiator}$, $N_{responder}$) (a random value that is considered to be random according to some very strict mathematical guidelines) in a nonce payload and their respective DH public value: g^x for the initiator and g^y for the responder. Now both initiator and responder create the master secret key (SKEYID). It is obtained by applying the agreed-to pseudorandom function to the known inputs:

(e.g. SKEYID = HMAC-MD5(N_i, N_r, g^{xy})

in addition, the two side will generate:
- SKEYID_d, used in phase 2 to protect user traffic.
SKEYID_d = HMAC-MD5(SKEYID, g^{xy}, $Cookie_{Initiator}$, $Cookie_{Responder}$, 0)
- SKEYID_a, used to authenticate ISAKMP message.
SKEYID_a = HMAC-MD5(SKEYID, SKEYID_d, g^{xy}, $Cookie_{Initiator}$, $Cookie_{Responder}$, 1).
- SKEYID_e, used to encrypt ISAKMP message.
SKEYID_e = HMAC-MD5(SKEYID, SKEYID_a, g^{xy}, $Cookie_{Initiator}$, $Cookie_{Responder}$, 2)
- At this point, all payloads will be encrypted with the derived key. The two entities can exchange identity information using a digital signature algorithm to authenticate themselves. The digital signature is not applied to the ISAKMP message. Instead it is applied to a hash ($hash_{initiator}$) all information available to both entities like SSL. All this information is carried in an identity payload, signature payload and optionally a certificate payload. After, the responder will verify the signed data and the trusted certificate authority, then repeat the same technique of authentication and data hash combination ($hash_{responder}$) and send then in an ISAKMP message to the initiator.
$HASH_{initiator}$ = HMAC-MD5(SKEYID, g^x, g^y, $Cookie_{Initiator}$, $Cookie_{Responder}$, SA_p, $ID_{initiator}$)
$HASH_{responder}$ = HMAC-MD5(SKEYID, g^y, g^x, $Cookie_{Responder}$,$Cookie_{Initiator}$, SA_p, $ID_{responder}$).
Where SA_p is the entire body of the SA payload that was sent by the initiator in the first message. $ID_{initiator}$, $ID_{responder}$ are respectively the initiator and responder identity information. [15]
At this point the two hosts are authenticated and phase 1 exchanges are complete.

III. Integrating ISAKMP in SSL

A. Motivations

There is a clear need for a standard on how to apply security services in public networks. ISAKMP is intended to support the negotiation of Security Associations (SA) for security protocols at all layers of the network stack (e.g., IPsec, TLS, …).
At the present time, each protocol brings its own rules. Contraring to the existing solutions, ISAKMP was considered to be generic and able to support all types of protocols. A manner to unify all these proposals is a real implementation of ISAKMP in SSL. It would be useful to experiment this protocol at different layers.
Our main motivations to adapt ISAKMP in SSL are as follows:

- ISAKMP is at present the only protocol that supports Identity Protection against sniffer attack. Identity Protection is primarily useful where one host has multiple identities and wishes to mask who is behind a specific handshake (unlike other protocol, identity in ISAKMP does not necessarily bear any relationship with IP address, but it can be related to various information's existing in certificates).
- Separating the functionality of key exchange from security association management is critical for interoperability between systems with differing security requirements. It also simplifies the analysis of further evaluation of an ISAKMP server.
- Since ISAKMP is an application layer protocol, then it benefits all the advantages of this layer; to include data comprehension, user authentication related to application and also the advantage of non-repudiation, which can cause an overhead in other sub layers.
- ISAKMP is very useful when establishing different security services for different applications. Under the same SA-ISAKMP phase 1 exchange, it's very possible that different types of traffic will need different sorts of protection. For example, *Telnet* sessions would probably need encryption whereas *DNS* service needs only authentication.
- Several protocols (e.g. TLS, IPsec…) could share the same key management code. This simplifies migration from one protocol to another and reduces the amount of duplicated functionality within each security protocol.
- Multiple certificate exchange in the same SA session connection. This can be useful for application that uses one certificate for authentication and another for authorization.
- New research is done for the distribution of the SPI in multicast for a more economical use of bandwidth. A Work group within the IETF defines new extensions to extend ISAKMP towards a multicast situation based on SA groups.
- Support for Attribute Certificate passing. This type is viewed as the new certificate's generation for web service, access control & authentication. ISAKMP support also seven known certificate format.
- In ISAKMP, sending a certificate to a responder is optional. The sender can transmit a URL towards an LDAP(Lightweight Directory Access Protocol)[17] pointing to his certificate. This is significant to reduce the time of ISAKMP exchange in the case where several certificates and authority certificates are transmitted between the two entities.
- Finally, ISAKMP is largely deployed due to its integration in operating systems and in IPv6 protocol, this make it absolutely necessary in the next Internet generation.

B. ISAKMP and SSL architecture

1. Introduction

The integration of ISAKMP in SSL [figure 5,6] can begin after an ISAKMP phase one negotiation. ISAKMP phase two is intended to create a Security Association SA for any protocol specified and defined in the DOI database. Unlike ISAKMP SA, the SA phase 2 is unidirectional. This give the two communicated entities different authentication methods. Any ISAKMP integration with SSL should respect the two main definitions.

1) the exchanged message should respect the ISAKMP message format. In our case, *Oakley quick mode* and other possible ISAKMP message combination are enough to assemble all SSL handshake scenarios with four different authentication methods (Server Authentication, client and server auth., RSA or DH auth. and resumed handshake) even though a normal phase two exchange is compared to an SSL session resumption.

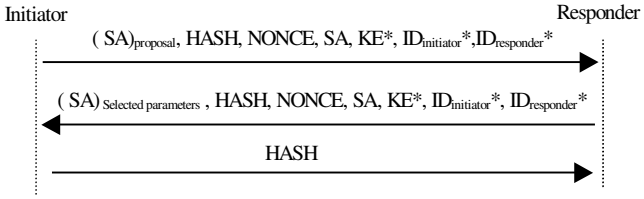

Figure 5. SSL ISAKMP exchange

2) The result in derived keys and exchanged algorithm names should be transparent to the SSL Record protocol. This is automatically provided because any phase 2 exchange will be based on Diffie-Hellman public keys, SKEYID and the NONCEs values.

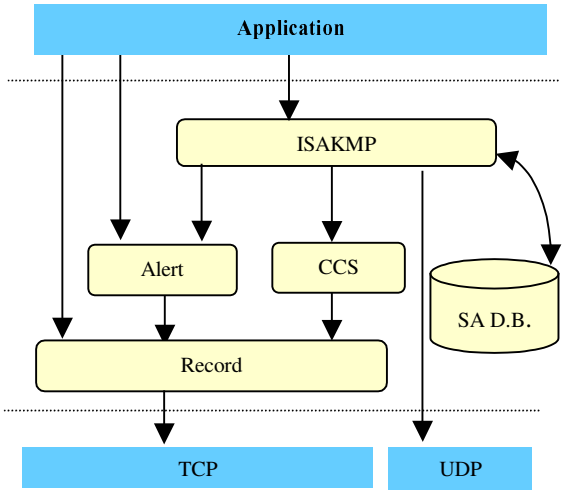

Figure 6. SSL ISAKMP Architecture

For SSL, ISAKMP will be used for bringing new authentication methods, non-repudiation, identity protection, and fast algorithm negotiations. These points will be detailed in this section.

a) New authentication methods

ISAKMP makes no distinct between client and server because the first use of ISAKMP was with IPsec and this distinction does not exist at the IP layer. Instead, we call the sender of the first packet the initiator and the second the responder. To preserve the interoperability with SSL handshake we will explain how ISAKMP can re-generate the four SSL authentication methods.

- First, all SSL authentication need a Diffie-Hellman key Exchange assured by ISAKMP with the KE payload.
- the random client and server in SSL are replaced with $NONCE_{initiator}$ and $NONCE_{responder}$ in SSL/ISAKMP.
- the list of negotiated algorithms in SSL/ISAKMP are assured with *proposal* and *transform* payloads.
- Now, the server authentication in SSL can be replaced with an X509 certificate for the responder and a NONE (defined as one of seven certificate format in rfc. 2408) certificate for the initiator. The certificates are transmitted in the Certificates payload. A HASH of all exchanged message encrypted with their public keys are exchanged in the *authentication payload*. A *signature payload* can also be added to sign to hash data.
- Like server authentication, client and server authentication is a bi-directional authentication with X509 certificates and signature payload.
- In SSL, the RSA or DH authentication require only KE payload for Diffie-Hellman or RSA exchange and a HASH Payload;

the last one in SSL is the session resumption This is the fastest key exchange in SSL/ISAKMP where the two entities generate new keying material from the original SA without a new key exchange.

In addition, other authentication methods can be gained by ISAKMP like authentication based on delegated attribute certificate, user information, manual shared secret, and distributed secret by a key distribution center.

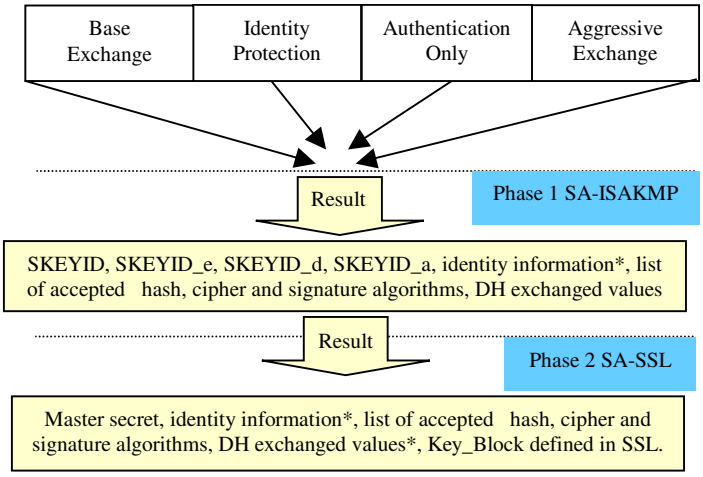

Figure 7. Phase 1 & 2 exchanged results.

Notation: *: for optional values.

b) Non-repudiation

The non-repudiation service is assured by the digital signature. Digital signature allows messages to be authentically sent by the original user or software. Unfortunately, SSL did not include provisions for signing messages because of the overhead that can be caused with digital signature in a transport layer and because that SSL was developed to address the problem of confidentiality and server authentication. Non-repudiation is an important requirement, specially, in Business-to-Business applications. Non-repudiation guarantees that no malicious sender can later disavow having created and sent a specific message. This

implies that non-repudiation guarantees that the sender of a message is the same as the creator of the message .

c) Identity protection

Identity protection is primarily useful for multi-users using a shared station or IP address. An ISAKMP long life phase 1 exchange can be established with authentication based on the IP address or station certificate. Every user can afterwards use his proper authentication methods that can even be a combination of IP address and certificates.

d) Fast algorithm negotiations

Unlike SSL handshake, the SSL/ISAKMP exchange is based on shared and authenticated keys. A new issue can be added to SSL for a fast handshake. If both hosts have defined the authentication method with X509 certificate, the initiator can send his certificate in the first message exchanged, with a signature payload (the signed data can be, all the exchanged messages in phase 1).

In SSL/ISAKMP three is the minimum number of necessary exchanged message.

IV. Implementation

For a real SSL/ISAKMP implementation, we have studied different open source API realized for SSL and ISAKMP. To conserve interoperability between SSL and ISAKMP we found KAME[4] API the most complete API in cryptographic algorithms, authentication methods, message format and configurable parameters.

The KAME Project was initiated in April 1998 in Japan by seven Japanese companies. It aimed to provide free, working, and "specification conformant" code based on BSD variants.

Because the cryptographic API of KAME is based on OpenSSL[5], the exchange between ISAKMP phase 1 and phase 2 will be transparent in term of API. Other work should be done in defining SSL parameters, SSL DOI, SSL negotiation methods and error control schemes.

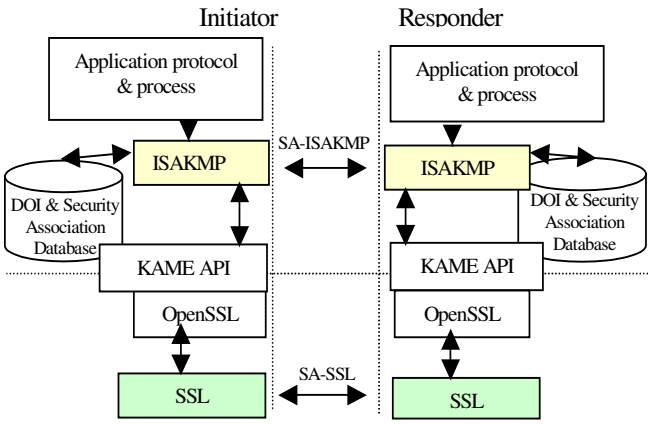

Figure 8. SSL ISAKMP modules and relationships

[4] Available site: www.kame.net

[5] OpenSSL is an Open Source toolkit first developed by Eric A. Young and Tim J. Hudson. OpenSSL implements the SSL and TLS protocols as well as a general purpose cryptography library. Available Site: www.openssl.org

V. Conclusion

The Internet Security Association and Key Management Protocol (ISAKMP), defines a framework for managing security associations between different entities in the network. Many protocols described in this paper are based or defined to work with ISAKMP but there remain a certain number of protocols, such as SSL that proposes their own protocol of key management. Adopting ISAKMP as a unique protocol for key negotiations can be possible to carry out significant optimisations. We showed that ISAKMP has the capability to support different types of protocols and also the possibility of sharing the same secure channel established by several protocols or various sessions of the same protocol. Beyond optimisations, ISAKMP is definitely more flexible than Handshake and thus has the capacity to meet needs of security services necessary for certain applications. This flexibility is what is needed now for the future growth of Internet. The idea of integration with SSL remains always experimental and requires a true implementation. This will be our next objective.

REFERENCES

[1] T. Dierks, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[2] D. maughan, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

[3] R. Molva, "Internet Security Architecture", Institut Eurécom, France.

[4] S. Kent, BBN Corp, R. Atkinson, "IP Authentication Header (AH)", RFC 2402, November 1998.

[5] S. Kent, BBN Corp, R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998

[6] D. Harkins, D. Carrel, "The Internet Key Exchange (IKE)", Cisco Systems, RFC 2409, November 1998

[7] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998

[8] P. Karn, W. Simpson, "Photuris : Session-Key Management Protocol", IETF RFC 2522, March 1999.

[9] Krawczyk, H., "SKEME: A Versatile Secure Key Exchange Mechanism for Internet", from IEEE Proceedings of the 1996 Symposium on Network and Distributed Systems Security.

[10] H. Orman, University of Arizona , "The OAKLEY Key Determination Protocol" , RFC 2402, November 1998.

[11] Jong-Hyeon Lee, "A survey on IPsec Key Management Protocols", University of Cambridge, England

[12] G. Caronni, A. Aziz, R. Skrenta, "SKIP – Securing the Internet", Sun Micro systems, CA

[13] D. Wagner, B. Schneier, "Analyse of the SSL 3.0 protocol", University of California, Couterspane Systems.

[14] Mark Baugher, Thomas Hardjono, "Group Domain of Interpretation for ISAKMP", Draft-ietf-smug-gdoi-01, November 2000

[15] "Using IPsec to Construct Secure Virtual Private Networks", IBM Corporation.

[16] Ghislaine Labouret, "IPSEC : Présentation Technique" , Hervé Schauer Consultants (HSC), Available Site :http://www.hsc.fr.

[17] W. Yeong, T. Howes, S. Kille, "Lightweight Directory Access Protocol" , IETF RFC 1777, March 1995.