Skype (v2.5) Protocol Data Structures (French)

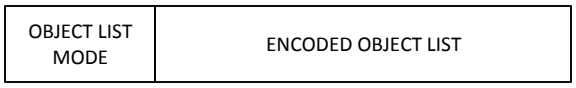
Author : Ouanilo MEDEGAN

http://www.oklabs.net

SKWRITTEN() désigne une valeur numérique encodée suivant l'algorithme d'encodage des valeurs numériques propre a Skype

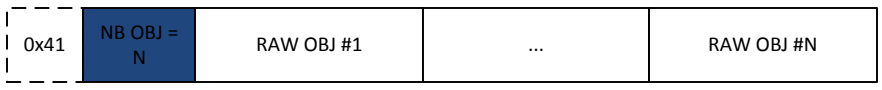
: Champ Encodé SKWRITTEN() : Champ Variable défini Précédemment & définissant l'état des champs à suivre

OBJECT LIST



- OBJECT LIST MODE : Mode d'encodage de l'ObjectList (1 Octet)
- ENCODED OBJECT LIST : Lise d'objets encodé suivant le mode spécifié (Taille Variable)

* MODE « RAW_OBJS »



- 0x41 : Valeur spécifiant le type d'ObjectList = RAW_OBJS (1 octet)
- NB OBJ : Nombre d'objets dans l'ObjectList (SizeOf SKWRITTEN())
- RAW OBJ : Objet d'une ObjectList de type RAW_OBJS (Taille Variable)

* MODE « EXT_OBJS »

- 0x42 : Valeur spécifiant le type d'ObjectList = EXT_OBJS (1 Octet)
- ENCODED OBJS : Objets Encodés suivant un algorithme de sérialisation propre a Skype impliquant un algorithme de compression propre a Skype (Taille Variable)

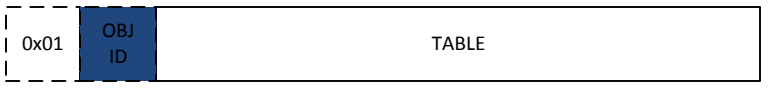
RAW OBJ

- OBJ TYPE : Type de l'objet encodé (1 Octet)
- OBJ ID : Identifiant permettant aux fonctions de reconnaitre le paramètre représenté par l'objet (SizeOf SKWRITTEN())
- ENCODED OBJ : Objet encodé en mode RAW (Taille Variable)

* OBJECT TYPE « NUMBER »

- SKWRITTEN(NUMBER) : Valeur numérique encodé suivant un algorithme propre a Skype (Taille Variable = SizeOf SKWRITTEN())

* OBJECT TYPE « DOUBLE »

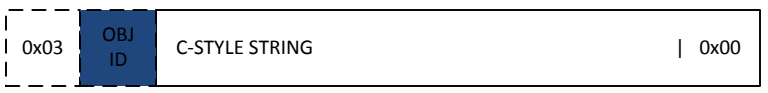


- TABLE : Tableau d'octets (8 Octets)

* OBJECT TYPE « INET_ADDR »

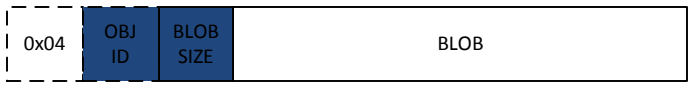
- IP ADDRESS : Adresse IP sur 32 bits (4 Octets)
- PORT : Port (IP) sur 16 bits (2 Octets)

* OBJECT TYPE « STRING »



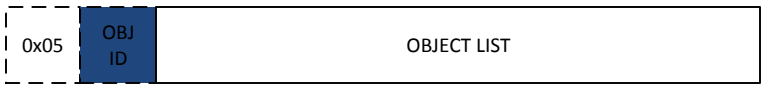
- C-STYLE STRING : Chaine de caractère, chaque caractère étant encodé sur 1 octet (ASCII), et terminée par le caractère NULL (0x00) (Taille Variable = Strlen())

* OBJECT TYPE « BLOB »



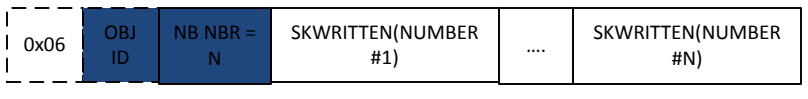
- BLOB SIZE : Taille en nombre d'octets du tableau (Taille Variable = SizeOf SKWRITTEN())
- BLOB : Suite d'octet représentant un tableau de taille variable (Taille Variable = BLOB SIZE)

* OBJECT TYPE « CONTAINER »



- OBJECT LIST : Liste d'objet (ObjectList) de type RAW_OBJS ou EXT_OBJS encapsulée (Taille Variable)

* OBJECT TYPE « NUMBER LIST »



- NB NBR : Nombre de valeurs numériques encodées (Taille Variable = SizeOf SKWRITTEN())
- SKWRITTEN(NUMBER) : Valeur numérique encodée via SKWRITTEN (Taille Variable = SizeOf SKWRITTEN())

SKWRITTEN() désigne une valeur numérique encodée suivant l'algorithme d'encodage des valeurs numériques propre a Skype

: Champ Encodé SKWRITTEN() : Champ Variable défini Précédemment & définissant l'état des champs à suivre

Paquet UDP

TRANSACTION ID	PACKET TYPE	DATAS

- TRANSACTION ID : Identifiant Permettant d'identifier le paquet (2 Octets)
- PACKET TYPE : Valeur permettant de définir le type de Paquet (1 Octet)

* PACKET TYPE & 0x8F = 0x07 : Paquet « NACK »

&= 0x07	PUBLIC IP	CHALLENGE

- PUBLIC IP : Adresse IP Publique du destinataire sur 32 bits (4 Octets)
- CHALLENGE : Valeur numérique servant a « authentifier » de manière basique un paquet rejouer, suite à la réception d'un paquet de type NACK (4 Octets)

* PACKET TYPE & 0x8F = 0x03 : Paquet « UDP/RC4 PACKET REPLAY »

&= 0x03	0x01	CHALLENGE	DESTINATION IP	CRC32	REPLAYED - RC4 (UDP PAYLOAD)

- 0x01 : Valeur Constante (1 Octet)
- CHALLENGE : Valeur « CHALLENGE » spécifiée dans le paquet NACK à l'origine du paquet de type REPLAY (4 Octets)
- DESTINATON IP : Adresse IP Publique de l'hôte auquel est destiné le paquet de type REPLAY, étant aussi celui ayant envoyé du paquet NACK à l'origine du paquet REPLAY (4 Octets)
- CRC32 : Valeur du hash CRC32 appliqué sur la charge utile du paquet en son état décrypté. Cette valeur sert à vérifier le bon décryptage de la charge utile (4 Octets)
- REPLAYED UDP/RC4 PAYLOAD : Charge Utile du paquet rejouée en l'état (contenu et cryptage RC4) (Taille Variable)

* PACKET TYPE & 0x8F = 0x02 : Paquet « UDP/RC4 PACKET »

- EnvPropsBlob :

PUBLIC IP	DESTINATION IP	TRANSACTION ID	\x0x00\x0x00

- PUBLIC IP : Adresse IP Publique de l'expéditeur sur 32 bits (4 Octets)
- DESTINATON IP : Adresse IP Publique du destinataire sur 32 bits (4 Octets)
- TRANSACTION ID : (Transaction ID du paquet) Identifiant Permettant d'identifier le paquet (2 Octets)

- La charge utile d'un paquet UDP est cryptée en utilisant l'algorithme RC4. La clé utilisée pour ce cryptage est grande de 128 bits et est générée à partir d'une valeur numérique pré calculable désignée, un « *Seed* ».. Le Seed est défini par l'opération :

Seed = CRC32(EnvPropsBlob) ^ INITIALIZATION VECTOR

&= 0x02	INITIALIZATION VECTOR	CRC32	RC4 (UDP PAYLOAD)

- INITIALIZATION VECTOR : Valeur numérique utilisée dans la génération du *Seed*, primitive de la clé utilisé pour crypter la charge utile via RC4. Elle est générée aléatoirement via un moteur de génération aléatoire très puissant impliquant de nombreux paramètres aléatoires du système et une forte distribution grâce à l'usage de l'algorithme SHA (4 Octets)
- CRC32 : Valeur du hash CRC32 appliqué sur la charge utile du paquet en son état décrypté. Cette valeur sert à vérifier le bon décryptage de la charge utile (4 Octets)
- RC4 (UDP PAYLOAD) : Charge utile du paquet UDP, chiffrée avec l'algorithme RC4, utilisant une clé de 128 bits générée à partir du *Seed* (Taille Variable)

UDP PAYLOAD

PAYLOAD LENGTH	CMD	REPLYTO ID	OBJECT LIST

- PAYLOAD LENGTH : Taille en octets de la charge utile. Cette valeur n'incluse ni sa propre taille en octet, ni celle du champ CMD (SizeOf SKWRITTEN())
- CMD : Valeur numérique représentant la commande, la fonction réseau, la requête contenue dans le paquet UDP (SizeOf SKWRITTEN())
- REPLYTO ID : Valeur numérique qui sert à identifier la requête, le paquet auquel est associé une réponse provenant du réseau dans la mesure où cette valeur est renvoyée dans la réponse. En générale elle vaut (TRANSACTION ID – 1) (2 Octets)
- OBJECT LIST : Liste d'objets (Object List) représentants les paramètres de la requête réseau (Taille Variable)

SKWRITTEN() désigne une valeur numérique encodée suivant l'algorithme d'encodage des valeurs numériques propre a Skype

: Champ Encodé SKWRITTEN()

: Champ Encrypté avec RC4

: Champ Encrypté avec AES

: Champ Variable défini Précédemment & définissant l'état des champs à suivre

: Champ à existence conditionnée

- Quand une connexion TCP s'effectue avec un pair sur le réseau (Super Node, Relay..), chacune des parties encrypte les paquets qu'il envoie en utilisant un flux RC4, c'est à dire un contexte RC4 qui est initialisé une et une seule fois avec une clé de 128 bits et qui est utilisé tout le long de la communication, sans réinitialisation ultérieure. Les deux parties doivent donc êtres en connaissances des deux clés utilisées pour initialiser chacun des flux RC4. Chacune des parties envoie donc à l'autre avant tout échange de données pertinentes, le *Seed* qui lui servira à générer la clé de son flux RC4. Une fois les clés connues des deux cotés, les échanges de données pertinentes peuvent commencer.

PEERS RC4 KEYS EXCHANGE PACKET

SEED	SEED FINGERPRINT	0x01	0x03	LEN	TYPE	RANDOM DATAS

- SEED : Valeur numérique utilisée pour générer la clé de 128 bits qui va servir à initialiser le flux RC4 qui cryptera tous les paquets sortants à venir de l'expéditeur (4 Octets)
- SEED FINGERPRINT : Valeur numérique servant d'empreinte pour le SEED. Il vaut SEED & 0x81FF (2 Octets)
- 0x01 : Cookie servant à vérifier la bonne initialisation du flux (4 Octets)
- 0x03 : Cookie servant à vérifier la bonne initialisation du flux (4 Octets)
- LEN : Valeur numérique indiquant le nombre d'octets restants du paquet. Supposons qu'il reste N octets. LEN = (2 x N) + 1 (1 Octet)
- TYPE : Valeur numérique servant à identifier le type de données pertinentes. Il vaut ici 0x03 (1 Octet)
- RANDOM DATAS : Suite aléatoire d'octets (Taille Variable = (LEN – 1) / 2)

PEERS CLASSIC TCP PACKET

G. LEN.	TRANSACTION ID	PAY. LEN.	CMD	REPLYTO ID	OBJECT LIST

- G. LEN. : Valeur numérique indiquant le nombre d'octets restants du paquet. Supposons qu'il reste N octets. LEN = N << 1 (SizeOf SKWRITTEN())
- TRANSACTION ID : Identifiant Permettant d'identifier le paquet (2 Octets)
- PAY. LEN. : Taille en octets de la charge utile. Cette valeur n'incluse ni sa propre taille en octet, ni celle du champ CMD (SizeOf SKWRITTEN())
- CMD : Valeur numérique représentant la commande, la fonction réseau, la requête contenue dans le paquet TCP (SizeOf SKWRITTEN())
- REPLYTO ID : Valeur numérique qui sert à identifier la requête, le paquet auquel est associé une réponse provenant du réseau dans la mesure où cette valeur est renvoyée dans la réponse. En générale elle vaut (TRANSACTION ID – 1) (2 Octets)
- OBJECT LIST : Liste d'objets (Object List) représentants les paramètres de la requête réseau (Taille Variable)

- Le champ REPLYTO ID n'existe pas quand la commande ne nécessite pas de réponse (Envoie de statistiques réseau, etc..). Seules les commandes respectant la règle suivante incluent ce champ : CMD & 0x07 = 0x03
- Plusieurs requêtes ou commandes peuvent être contenues dans un seul paquet, concaténées les une après les autres.

PEERS USER TCP PACKET

- Pendant une session texte/voix/vidéo à travers un relais, et allant d'utilisateur à utilisateur via un relais, ce sont des paquets de type User Packet qui sont envoyés. Ceux-ci sont eux aussi passés dans le flux RC4, mais avant cela, hormis quelques octets descriptifs, ils sont d'abord encryptés en utilisant l'algorithme AES 256 en mode Counter avec une clé négociée au fur à mesure de la négociation de la session.

LEN.	0x05	CTRL ID	TRANSACTION ID	CMD	OBJECT LIST	CRC

- LEN. : Valeur numérique indiquant le nombre d'octets restants du paquet. Supposons qu'il reste N octets. LEN = (2 x (N + 1)) + 5 (SizeOf SKWRITTEN())
- 0x05 : Valeur propre aux paquets de type User Packet (1 Octet)
- CTRL ID : Valeur numérique servant à authentifier le paquet par rapport au Relay sur lequel il est envoyé. Il est généré à partir du hash CRC32 de données pertinentes du paquet et d'une clé propre au Relay fournie durant la négociation du paquet (2 Octets)
- TRANSACTION ID : Identifiant Permettant d'identifier le paquet (SizeOf SKWRITTEN() [Max. 2 Octets])
- CMD : Valeur numérique représentant la commande, la fonction réseau, la requête contenue dans le paquet TCP (SizeOf SKWRITTEN())
- OBJECT LIST : Liste d'objets (Object List) représentants les paramètres de la requête réseau (Taille Variable)
- CRC : Deux derniers octets de la valeur du hash CRC32 des données cryptées par AES (2 Octets)

CENTRAL STRUCTURE TCP PACKET

- Lorsque le client Skype se connecte à une architecture centrale via le port 443, les paquets échangés « sont de type HTTPS », cf. le chapitre Handshakes. Le client envoie des paquets de type Server Hello dont le champ *Content Type* vaut *Handshake* (0x16), et reçoit du serveur des paquets du même type dont le champ *Content Type* vaut *Application Data* (0x17). Les données pertinentes de ces paquets sont encryptés en utilisant un contexte AES 256, dont la clé est transmise encryptée avec RSA de manière que seule « Skype S.A. » puissent la déchiffrer.

MAGIC (\x16\x03\x01)	PAYLOAD LENGTH	PAYLOAD

- Paquet envoyé vers une structure centrale. Les commandes et les paramètres sont contenues dans le PAYLOAD sous forme d'ObjectList.

MAGIC (\x17\x03\x01)	PAYLOAD LENGTH	PAYLOAD

- Paquet reçu d'une structure centrale. Les commandes et les paramètres sont contenues dans le PAYLOAD sous forme d'ObjectList.

SKWRITTEN() désigne une valeur numérique encodée suivant l'algorithme d'encodage des valeurs numériques propre a Skype

: Champ Encodé SKWRITTEN() : Champ Variable défini Précédemment & définissant l'état des champs à suivre

- Quand une connexion TCP s'effectue via le port 443, dans le but de « bypasser » les firewall et autres, le client Skype envoie et reçois en premier des paquets de type HTTPS HANDSHAKE

PEER TCP HTTPS HANDSHAKE

- Lorsque la connexion s'effectue vers un pair sur le réseau (SuperNode, Relay..), le paquet HTTPS HANDSHAKE envoyé est un paquet *Client Hello*. Celui utilisé par le client Skype est toujours le même à un champ près , le champ « Challenge », représenté par les 16 derniers octets et qui est généré aléatoirement :

{0x80, 0x46, 0x01, 0x03, 0x01, 0x00, 0x2D, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x05, 0x00, 0x00,
0x04, 0x00, 0x00, 0x0A, 0x00, 0x00, 0x09, 0x00, 0x00, 0x64, 0x00, 0x00, 0x62, 0x00, 0x00, 0x08,
0x00, 0x00, 0x03, 0x00, 0x00, 0x06, 0x01, 0x00, 0x80, 0x07, 0x00, 0xC0, 0x03, 0x00, 0x80, 0x06,
0x00, 0x40, 0x02, 0x00, 0x80, 0x04, 0x00, 0x80, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};

```
SSLv2 Record Layer: Client Hello
    Length: 70
    Handshake Message Type: Client Hello (1)
    Version: TLS 1.0 (0x0301)
    Cipher Spec Length: 45
    Session ID Length: 0
    Challenge Length: 16
  Cipher Specs (15 specs)
      Cipher Spec: TLS_RSA_WITH_RC4_128_SHA (0x000005)
      Cipher Spec: TLS_RSA_WITH_RC4_128_MD5 (0x000004)
      Cipher Spec: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x00000a)
      Cipher Spec: TLS_RSA_WITH_DES_CBC_SHA (0x000009)
      Cipher Spec: TLS_RSA_EXPORT1024_WITH_RC4_56_SHA (0x000064)
      Cipher Spec: TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA (0x000062)
      Cipher Spec: TLS_RSA_EXPORT_WITH_DES40_CBC_SHA (0x000008)
      Cipher Spec: TLS_RSA_EXPORT_WITH_RC4_40_MD5 (0x000003)
      Cipher Spec: TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 (0x000006)
      Cipher Spec: SSL2_RC4_128_WITH_MD5 (0x010080)
      Cipher Spec: SSL2_DES_192_EDE3_CBC_WITH_MD5 (0x0700c0)
      Cipher Spec: SSL2_RC2_CBC_128_CBC_WITH_MD5 (0x030080)
      Cipher Spec: SSL2_DES_64_CBC_WITH_MD5 (0x060040)
      Cipher Spec: SSL2_RC4_128_EXPORT40_WITH_MD5 (0x020080)
      Cipher Spec: SSL2_RC2_CBC_128_CBC_WITH_MD5 (0x040080)
    Challenge
```

PEER TCP HTTPS HANDSHAKE RESPONSE

- Lorsque le client Skype envoie un paquet de type *Client Hello* , il reçoit du pair une réponse HTTPS de type *Server Hello. Ex* :

{0x16, 0x03, 0x01, 0x00, 0x4A, 0x02, 0x00, 0x00, 0x46, 0x03, 0x01, 0x40, 0x1B, 0xE4, 0x86, 0x02,
0xAD, 0xE0, 0x29, 0xE1, 0x77, 0x74, 0xE5, 0x44, 0xB9, 0xC9, 0x9C, 0xB4, 0x31, 0x31, 0x5E, 0x02,
0xDD, 0x77, 0x9D, 0x15, 0x4A, 0x96, 0x09, 0xBA, 0x5D, 0xA8, 0x70, 0x20, 0x1C, 0xA0, 0xE4, 0xF6,
0x4C, 0x63, 0x51, 0xAE, 0x2F, 0x8E, 0x4E, 0xE1, 0xE6, 0x76, 0x6A, 0x0A, 0x88, 0xD5, 0xD8, 0xC5,
0x5C, 0xAE, 0x98, 0xC5, 0xE4, 0x81, 0xF2, 0x2A, 0x69, 0xBF, 0x90, 0x58, 0x00, 0x05, 0x00};

```
TLSv1 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 74
  Handshake Protocol: Server Hello
      Handshake Type: Server Hello (2)
      Length: 70
      Version: TLS 1.0 (0x0301)
    Random
        gmt_unix_time: Jan 31, 2004 18:23:18.000000000
        random_bytes: 02ADE029E17774E544B9C99CB431315E02DD779D154A9609...
      Session ID Length: 32
      Session ID (32 bytes)
      Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
      Compression Method: null (0)
```

- Ces paquets n'ont aucune utilité si ce n'est celle de flouter les firewall et autres dispositifs. Des paquets de type Server Hello nous ne retiendrons deux champs.

MAGIC (\x16\x03\x01)	PAYLOAD LENGTH	PAYLOAD

- MAGIC : Suite d'octets propres aux paquets de type Server Hello (3 Octets)
- PAYLOAD LENGTH : Taille en octets de la charge « utile » du paquet (2 Octets)
- PAYLOAD : Charge « Utile », ici inutile (Taille Variable = PAYLOAD LENGTH)

CENTRAL STRUCTURE TCP HTTPS HANDSHAKE

- Lorsque la connexion s'effectue vers une structure centrale du réseau (LoginServer, EventServer) le paquet HTTPS HANDSHAKE envoyé est de type *Server Hello* et ne contient pas de charge utile:

{0x16, 0x03, 0x01, 0x00, 0x00};

CENTRAL STRUCTURE TCP HTTPS HANDSHAKE RESPONSE

- Lorsque le client envoie un paquet HTTPS HANDSHAKE à une architecture centrale du réseau il reçoit en réponse un paquet de type Server Hello ne contenant pas de charge utile et dont le champ *Content Type* au lieu de valoir *Handshake* (0x16), vaut *Application Data* (0x17) :

{0x17, 0x03, 0x01, 0x00, 0x00};

SKWRITTEN() désigne une valeur numérique encodée suivant l'algorithme d'encodage des valeurs numériques propre a Skype

: Champ Encodé SKWRITTEN()

: Champ Encrypté avec RC4

: Champ Encrypté avec AES

: Champ Encrypté avec RSA

(Précisons que les données cryptées avec RSA sont d'abord paddées ou formatées selon un certaine taille (0x80 / 0xC0), avant d'être cryptées. Elles se voient aussi rajoutées des hash de contrôle (SHA) afin d'assurer leur intégrité)

: Champ Variable défini Précédemment & définissant l'état des champs à suivre

: Champ à existence conditionnée

LOCATION BUFFER

NODE ID	P.N.	INTERFACE IP	INTERFACE PORT	PARENTNODE IP	PARENTNODE PORT

- NODE ID : Le Node ID est un identifiant unique identifiant chaque nœud (client, machine) du réseau (8 Octets)
- P.N. (ParentNode ?) : Ce champ peut valoir 0x01 ou 0x00. Si le champ vaut 0x00 cela veut dire que le nœud dispose d'une interface (adresse) publique, et est donc accessible directement sans forcément passer par un Parent Node. Si le champ vaut 0x01, le nœud n'est joignable que par le biais d'un Super Node lui servant de Parent Node (1 Octet).
- INTERFACE IP : Adresse de l'interface du nœud sur 32 bits. C'est une adresse publique si le nœud est directement joignable sinon c'est son adresse privée (4 Octets)
- INTERFACE PORT : Port de l'interface du nœud sur 16 bits (2 Octets)
- PARENTNODE IP : Champ existant seulement si le nœud n'est pas directement joignable. Il représente l'adresse IP du Super Node qui lui sert de Parent Node, sur 32 bits (4 Octets)
- PARENTNODE PORT : Champ existant seulement si le nœud n'est pas directement joignable. Il représente le port du Super Node qui lui sert de Parent Node, sur 16 bits (2 Octets)

SIGNED CREDENTIALS

- Après l'authentification d'un client Skype, auprès d'un LoginServer, celui-ci reçoit une structure de données, ici baptisée « Signed Credentials ». Cette structure va lui permettre de garantir son authenticité auprès de ses pairs du réseau tout au long de la session, dans la mesure où cette structure de données est signée, c'est à dire « crypté » par le LoginServer auprès duquel le client s'est authentifié avec succès, et cela en utilisant une clé privée connue seulement du LoginServer. Tout pair recevant cette structure de données, si elle arrive à la décrypter en utilisant la clé publique associée à la clé privée utilisée et spécifiée dans cette structure, a donc la garantie que le pair qui lui envoie cette structure est bien authentifiée auprès de « Skype S.A. ».

KEY INDEX	RSA(CREDENTIAL OBJECTS)

- KEY INDEX : Index de la clé publique associée à la clé privée utilisée par le LoginServer pour signer la structure de données Signed Credentials (4 Octets)
- RSA(CREDENTIAL OBJECTS) : Données pertinentes des Signed Credentials cryptées avec RSA par le LoginServer (192 Octets)

* CREDENTIAL OBJECTS

- OBJET « USER »
  Type : STRING
  Id : 0x00
  Value : User Login
- OBJET « X »
  Type : NUMBER
  Id : 0x03
  Value : 0x00
- OBJET « PUBLIC KEY »
  Type : BLOB
  Id : 0x01
  Value : User Public Key
- OBJET « UIC EXPIRY »
  Type : NUMBER
  Id : 0x04
  Value : Durée de validité des Signed Credentials

AUTHENTICATED DATAS

- Lorsque le client Skype doit envoyer des données sensibles (Location Buffer, Contact AuthCert, ..) sur le réseau, afin de garantir l'authenticité de ces données, il forge un blob de données suivant une structure spéciale et qui va être passé comme paramètre. Ce blob en question transporte les données sensibles cryptées ainsi que les preuves de l'authenticité des données, à savoir Signed Credentials et le fait que les données sensibles soient signées c'est à dire cryptées avec la clé privée de l'utilisateur, ce qui implique que seule sa clé publique peut les décrypter, clé publique qui se récupère dans les Signed Credentials qu'il faut commencer par décrypter, simple fait qui authentifie les données.

SIGNED CREDENTIALS SIZE	SIGNED CREDENTIALS	RSA(SENSIBLE DATAS)

- SIGNED CREDENTIALS SIZE : Taille en octets de la structure de données Signed Credentials (4 Octets)
- SIGNED CREDENTIALS : La structure de données Signed Credentials attribuée à l'utilisateur après son authentification réussie auprès d'un LoginServer (Taille Variable = SIGNED CREDENTIALS SIZE)
- RSA(SENSIBLE DATAS) : Données sensibles encryptés avec RSA en utilisant la clé privée de l'utilisateur, ce qui implique que seul la clé publique de l'utilisateur peut décrypter des données cryptées (Taille Variable)