Internet Engineering Task Force IETF MSEC
Internet Draft Perrig, Canetti, Whillock
draft-ietf-msec-tesla-spec-00.txt CMU / IBM / CMU
27 October 2002
Expires: 27 April 2002

# TESLA: Multicast Source Authentication Transform Specification

STATUS OF THIS MEMO



Abstract

Data authentication is an important component for many applications, for example audio and video Internet broadcasts, or data distribution by satellite. This document specifies TESLA, a secure source authen tication mechanism for multicast or broadcast data streams. The com panion draft draft-msec-tesla-intro-01.txt [1] introduces and describes TESLA in detail, this document specifies the format of the TESLA authentication field as it is used within the MESP header [2].

The main deterrents so far for a data authentication mechanism for multicast were seemingly conflicting requirements: tolerance to packet loss, low per-packet overhead, low computation overhead, scal ability, no per-receiver state at the sender. The problem is particu larly hard in settings with high packet loss rates and where lost packets are not retransmitted, and where the receiver wants to authenticate each packet it receives.

TESLA provides multicast source authentication of individual data packets, regardless of the packet loss rate. In addition, TESLA

features low overhead for both sender and receiver, and does not require per-receiver state at the sender. TESLA is secure as long as the sender and receiver are loosely time synchronized.

Table of Contents



## 1 Introduction

This document specifies the format of the TESLA source authentication data as an external-authentication transform within the MESP protocol [2]. It specifies the cryptographic operations and message formats. The description of the TESLA protocol is in the companion draft draft-msec-tesla-intro-01.txt [1], more details are in our academic publications [3,4,5,6].

## 1.1 Previous Work

A number of schemes for solving data authentication have been sug gested [7,8,9,10,11,6,5,12,13]. An Internet Draft based on [11] was proposed by McCarthy in 1998 but was not updated.

This document is based on the TESLA [3,6,5] and FLAMeS [12] schemes, which have low computation and communication overhead. Similar schemes were suggested by Cheung [14], and by Bergadano et al.

[13,15].

### 1.2 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [16].

## 2 TESLA Specification

TESLA is described in several academic publications: A book on broad cast security [3], a journal paper [4], and two conference papers [6,5]. Please refer to these publications for an in-depth treatment. The TESLA companion Internet Draft gives a basic introduction to the TESLA protocol [1].

### 2.1 Sender Setup

The sender chooses a time interval duration T_int. Practical values for T_int vary from 100 milliseconds to 1 second.

The sender estimates an upper bound on the round-trip-time (RTT). More specifically, this RTT is the propagation delay of a unicast packet from the receiver to the sender, plus the propagation delay of a broadcast packet from the sender to the receiver. Based on the RTT, the sender chooses a key disclosure delay d, where d denotes a number of time intervals, such that d > ceil(RTT/T_int).

The sender picks the starting time of time interval zero, which we denote with T_0. The starting time of time interval i is thus T_0 + i * T_int.

Next, the sender pre-computes the one-way key chain. Each key of the one-way key chain is active in one time interval, for example, key K_i is active in time interval i. The sender always uses the active key to compute the MAC of a packet it sends. The sender discloses the keys with a delay of d time intervals, so it discloses key K_i-d in time interval i.

To generate the one-way key chain, the sender chooses the length of the chain N. In future versions, we will define extensions which allow the sender to switch key chains, but for now we assume that N is large enough such that the key chain is long enough for the dura tion of the broadcast. The sender randomly selects the last key of the one-way key chain K_N. The random choice of K_N is critical, since the security of TESLA relies on the fact that an attacker cannot guess K_N. Simply using a hash of the current time, process id, port num ber, etc. is thus not sufficient, and we suggest using hardware-based

random number generators, or operating system provided random number generators such as /dev/random or /dev/urandom on some UNIX systems. The bit length of K_N is a global parameter, and needs to match the output length of the pseudo-random function F that generates the one-way key chain (see the companion draft for more details on how the PRF F is used to generate the one-way chain and the PRF F' is used to derive the MAC key [1]).

The sender uses a pseudo-random function F with target-collision resistance to generate the previous elements of the chain. We suggest using HMAC-MD5 for this purpose [17,18], where the output is trun cated to the bit length of the key. If the key length is larger than 128 bits, we suggest using HMAC-SHA-1; the 160 bit size should be sufficient for all practical purposes.

Jakobsson [19], and Coppersmith and Jakobsson [20] present a storage and computation efficient mechanism for one-way chains. For a chain of length N, storage is about log(N) elements, and the computation overhead to reconstruct each element is also about log(N).

The companion document draft-msec-tesla-intro-01.txt [1] has more information on the one-way key chain.

### 2.2 Receiver Bootstrapping

TESLA requires that the sender and the receiver be at least loosely time synchronized such that the receiver MUST know an upper bound on the sender's clock. The receiver MUST also receive authentic parame ters for initiating the TESLA session. Authentication is achieved with a digital signature such as RSA or DSA.

#### 2.2.1 Bootstrapping Parameters

In order to bootstrap, the receiver must receive the following authenticated information (Note that the Cryptographic Type Assigned Number (CTAN) is a 16-bit integer describing the type of authentica tion used to generate the signature Sig. CTAN is described in Appendix A.):

- The id j of a specific time interval I_j.
- An NTP timestamp TI_j describing the beginning of that time interval.
- An NTP timestamp T_int describing the interval duration.
- A PRF CTAN describing the function that will be used to calculate the keychain (F).

- A PRF CTAN describing the function that will be used to derive the MAC key from the keychain (F').
- The key disclosure interval d (unit is intervals).
- A bit I telling whether or not the optional id field will be in the TESLA authentication tag.
- An Encryption CTAN representing the type of key to be used.
- A disclosed key K_i (i < j - d).
- The id n of the final key in this key chain, K_n.
- The interval d_n of the last key chain element.

The Network Time Protocol (NTP) timestamp is described in RFC 958 and is a 64 bit integer which can represent time with precision of .2 nanoseconds. It is noted that this format will overflow in year 2030, TESLA will agree with any format changes in NTP to accomodate this problem.

### 2.2.2 Direct Time Synchronization

In direct time synchronization, the receiver will synchronize its clock with the sender by determining an upper bound on the sender's clock. The protocol is very simple, and sufficies for the loose time synchronization required by TESLA.

The receiver sends an initial time synchronization request to the sender. This time synchronization request will contain only a random nonce N_r to later ensure that data received from the sender is authentic. The receiver will then record the time T_s at which the nonce was sent. Figure 1 describes the format of the nonce. The size of the nonce is a multiple of 8 bits, and the sender can deduce the nonce size from the size of the request packet.

Once N_r has been received, the sender responds with the bootstrap ping parameters as well as the following time synchronization parame ters:

- An NTP timestamp T_sig describing when the data was signed.
- An Authentication CTAN S_sig describing the signature type.
- A bit F, allowing for an optional formatting of the signature to include a public certificate chain.

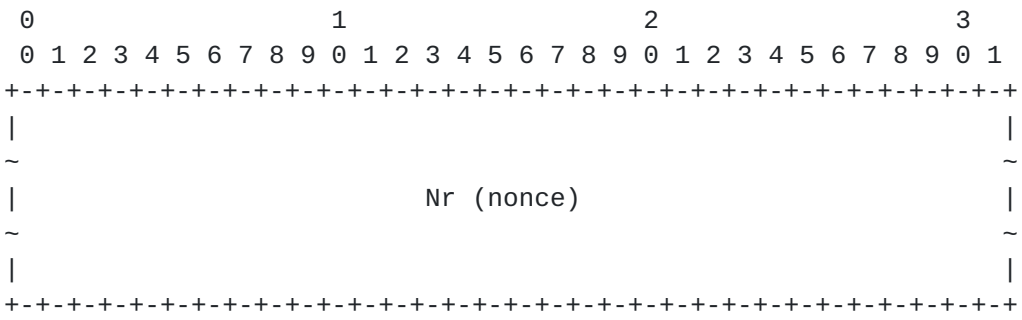


Figure 1: Time synchronization request format

· The signature Sig, used to authenticate the parameters.

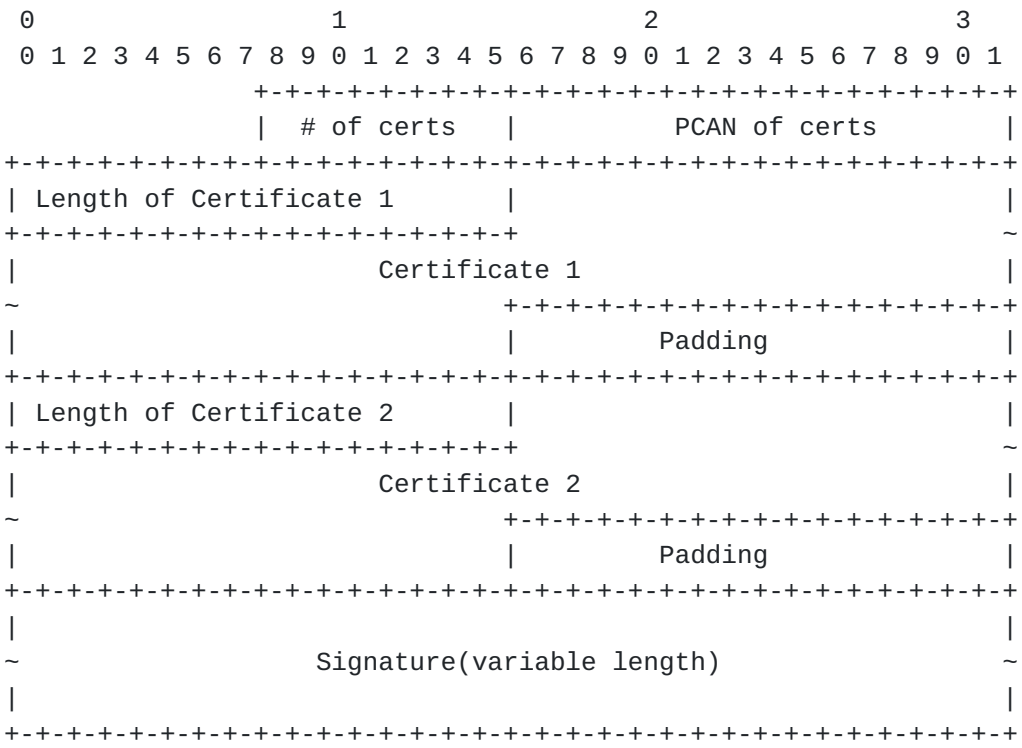


Figure 2: Optional Signature Format

Sig is a signature of the bootstrapping parameters, the time synchro nization parameters (exluding Sig), and N_r. The receiver can authen ticate the data by appending its nonce to the parameter and use this data to verify the signature. This method assumes that the receiver has an authentic public key of the sender, for example using a pub lic-key infrastructure, so the receiver has a list of authentic pub lic root key certificates of the certificate authorities, and the sender sends along its certificate signed by a certificate authority.

Once a receiver has received both sets of parameters, it records the time at which he received the parameters, T_r. The maximum clock off set is d_t = T_r - T_s. The receiver can now easily compute an upper bound on the sender's current time t_s, using it's current local time t_r as follows: t_s = t_r - T_r + T_s. If d_t is larger than a cer tain threshold, the receiver may repeat time synchronization.

TESLA will return the data in the form of a Signature Tag, which has the format described in Figure 3. If so required TESLA will include a certificate chain for authentication via RSA or another public cer tificate algorithm. If the bit F is set, then the optional format described by  2 is used. The PCAN is a Public Certificate Assigned Number describing what format the certificates are in.

### 2.2.3 Set-up using a multicast key management protocol.

Another way to set-up the parameters of the TESLA transform at the receiver is to obtain these parameters in the Security Association provided by the multicast key management protocol in use (e.g., GDOI). This way of setting the parameters will be further described in future versions of this draft.

## 2.3 Layer Placement

This document assumes TESLA to be implemented as a standalone library, which could reside either in the network, transport, or application layer. However, incorporating TESLA within MESP implies usage in the network layer.

TESLA relies upon timing of packets, that is, TESLA requires knowing the arrival time of incoming packets. TESLA SHOULD NOT be deployed on top of a protocol or layer which will aggressively buffer packets and hides the true packet arrival time, e.g.  TCP.

### 2.3.1 Interface Specification

The following describes the interface which the TESLA library will export for the above methods of bootstrapping.

#### 2.3.1.1 Time Synchronization request

TESLA will accept a nonce and write a time synchronization request tag (TSRT) to a user specified buffer which must be of required length. The nonce must have significant random properties (non-trivial and of certain length). The implementer must provide adequate space to write the TSRT.

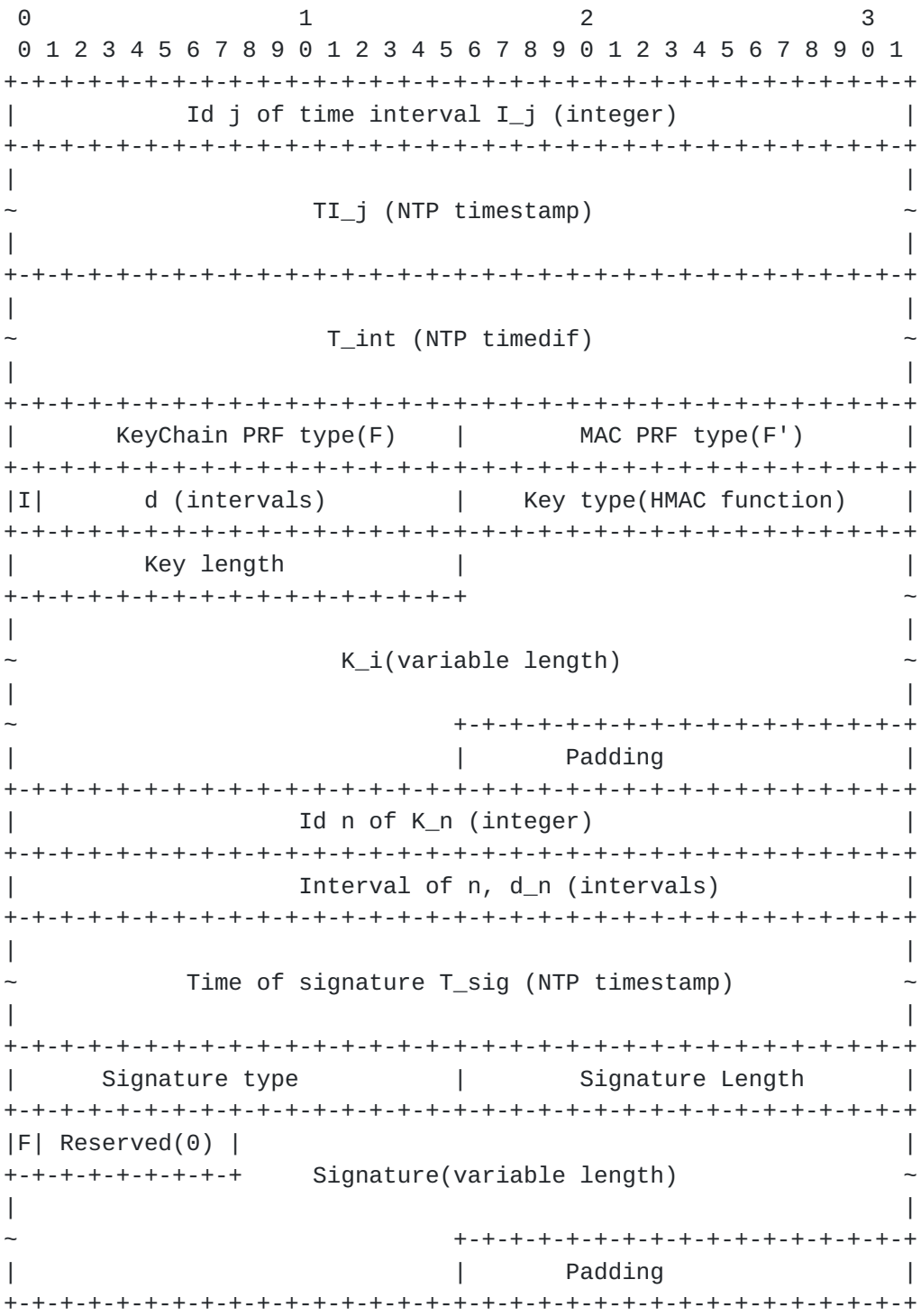


Figure 3: Signature Tag format

#### 2.3.1.2 Authentic Parameters

TESLA will accept the time synchronization request, which has the format described in Figure 1, and write the signature tag to a user specified buffer. It is assumed that the signature tag will be sent to the receiver immediately following the creation of the signature. Significant delay would result in too large a bound for d_t and pos sible synchronization failure.

### 2.4 Sending Authenticated Data

When the sender sends an authenticated message to all receivers, it adds a TESLA authentication tag to attach to the message. With the authentication tag, a receiver MAY be able to verify or disrepute previously received messages.

TESLA will be used as the external authentication transform in the MESP protocol. (Recall that MESP determines exactly which fields are covered under the TESLA authentication.) The format of the TESLA authentication tag is shown in Figure 4. Here M denotes the data in the fields covered by the external authentication in MESP. Within the TESLA tag, the Id i of K_i is always sent with the MAC of the message M computed using K_i. The last disclosed key K_(i-d) can be used to authenticate previous messages.

When a receiver receives the tag, he must first check to see that the time of the message does not violate the security conditions for the keys used. M is buffered, and he attempts to authenticate any mes sages which relied upon K_(i-d).

If i is not included in the message, the receiver determines i by the time the packet was received and the maximum time displacement from the server. With this time it then can determine the sender's current interval i.

When the receiver receives an MESP packet with external authentication done using TESLA, it first needs to verify whether the packet is safe, which is to check that the key used to compute the MAC of the packet was still secret upon packet arrival. For this, the receiver computes an upper bound on the sender's clock, and checks that the MAC key is still secret (based on the key disclosure schedule). If the packet is safe, the receiver buffers the packet.

Once the receiver has determined i, either directly or by the sender's time, it checks K_(i-d) against the most recently stored key, K_c. If i-d=c then the receiver does nothing. Otherwise he applies the PRF (i-d)-c times to K_(i-d) which should yield K_c. If K_(i-d) is authentic, the receiver uses it to authenticate all mes sages which used keys in the range K_(c+1) .. K_(i-d) as the MAC key.

Finally the receiver replaces K_c with K_(i-d). Note, that if i-d<c the packet would have been unsafe and discarded before this step.

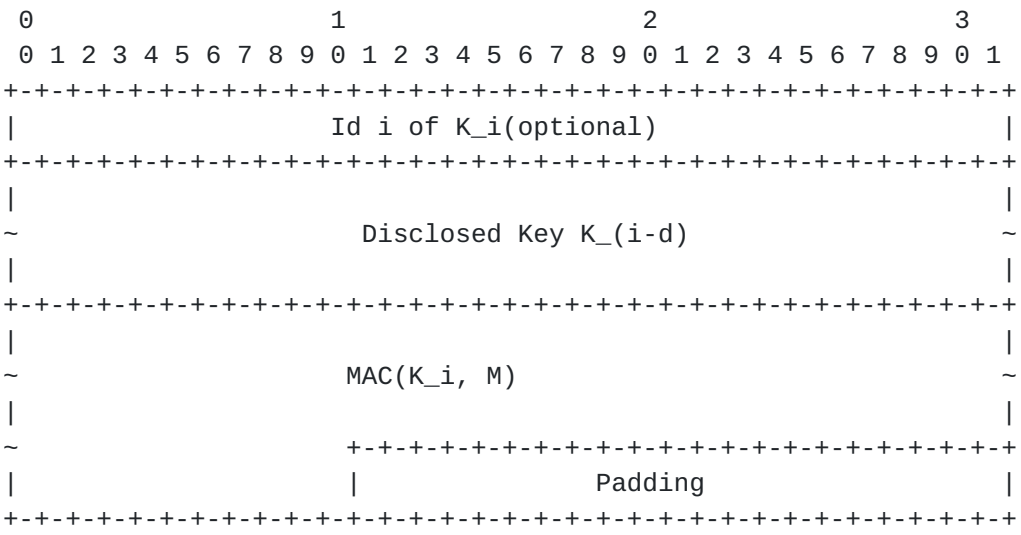


Figure 4: Authentication Tag format. (This is the external authentication tag of MESP with TESLA.)

A message in this interval may be authentic or tainted. Dealing with tainted messages is left to the implementor. It is possible that the sender and receiver have fallen out of sync, in which case it is RECOMMENDED that they resynchronize times. If i is not included in the message to the receiver and the two have fallen out of sync, the receiver will not correctly compute i, and failure will occur when attempting to authenticate the key.

## 3 Security Considerations

For a formal proof of the security of TESLA, please see [6]. An analysis of the robustness of TESLA to denial-of-

service attacks along with countermeasures is described in [5].

## 4 Acknowledgments

We would like to thank Mike Luby for his feedback and support.

## 5 Bibliography

[1] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Tesla: Multi cast source authentication transform introduction (draft-msec-tesla-intro-01.txt)," Internet Draft, Internet Engineering Task Force, Oct. 2002. Work in progress.

[2] R. Canetti, P. R. P.-C. Cheng, and M. Baugher, "Mesp: Multicast encapsulating security payload (draft-ietf-msec-mesp-00.txt)," Inter net Draft, Internet Engineering Task Force, Oct. 2002. Work in progress.

[3] A. Perrig and J. D. Tygar, Secure Broadcast Communication in Wired and Wireless Networks Kluwer Academic Publishers, Oct. 2002. ISBN 0792376501.

[4] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The tesla broadcast authentication protocol," RSA CryptoBytes , vol. 5, no. Summer, 2002.

[5] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in Network and Dis tributed System Security Symposium, NDSS '01 , pp. 35--46, February 2001.

[6] A. Perrig, R. Canetti, J. Tygar, and D. X. Song, "Efficient authentication and signing of multicast streams over lossy channels," in IEEE Symposium on Security and Privacy , May 2000.

[7] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient construc tions," in Infocom '99 , 1999.

[8] R. Gennaro and P. Rohatgi, "How to Sign Digital Streams," tech. rep., IBM T.J.Watson Research Center, 1997.

[9] P. Rohatgi, "A compact and fast hybrid signature scheme for mul ticast packet authentication," in 6th ACM Conference on Computer and Communications Security , November 1999.

[10] P. Rohatgi, "A hybrid signature scheme for multicast source authentication," Internet Draft, Internet Engineering Task Force,

June 1999. Work in progress.

[11] C. K. Wong and S. S. Lam, "Digital signatures for flows and mul ticasts," in Proc. IEEE ICNP `98 , 1998.

[12] B. Briscoe, "Flames: Fast, loss-tolerant authentication of mul ticast streams," tech. rep., BT Research, 2000. http://www.labs.bt.com/people/briscorj/papers.html.

[13] F. Bergadano, D. Cavagnino, and B. Crispo, "Chained stream authentication," in Selected Areas in Cryptography 2000 , (Waterloo, Canada), August 2000. A talk describing this scheme was given at IBM Watson in August 1998.

[14] S. Cheung, "An efficient message authentication scheme for link state routing," in 13th Annual Computer Security Applications Confer ence , 1997.

[15] F. Bergadano, D. Cavalino, and B. Crispo, "Individual single source authentication on the mbone," in ICME 2000 , Aug 2000. A talk containing this work was given at IBM Watson, August 1998.

[16] S. Bradner, "Key words for use in RFCs to indicate requirement levels," Request for Comments (Best Current Practice) 2119, Internet Engineering Task Force, Mar. 1997.

[17] M. Bellare, R. Canetti, and H. Krawczyk, "HMAC: Keyed-hashing for message authentication," Internet Request for Comment RFC 2104, Internet Engineering Task Force, Feb. 1997.

[18] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in Advances in Cryptology - Crypto '96 (N. Koblitz, ed.), (Berlin), pp. 1--15, Springer-Verlag, 1996. Lec ture Notes in Computer Science Volume 1109.

[19] M. Jakobsson, "Fractal hash sequence representation and traver sal." Cryptology ePrint Archive, http://eprint.iacr.org/2002/001/, Jan. 2002.

[20] D. Coppersmith and M. Jakobsson, "Almost optimal hash sequence traversal," in Proceedings of the Sixth International Financial Cryp tography Conference (FC '02) , March 2002.

## A Cryptographic Type Assigned Numbers

- PRF

  There are currently no pseudo-random functions defined.

values 1-65000 are reserved to IANA. Values 65001-65535 are for private use among mutually consenting parties.

| - Encryption Algorithm | | Defined In |
|---|---|---|
| DES-CBC | 1 | RFC 2405 |
| IDEA-CBC | 2 | |
| Blowfish-CBC | 3 | |
| RC5-R16-B64-CBC | 4 | |
| 3DES-CBC | 5 | |
| CAST-CBC | 6 | |

| - Authentication Method | |
|---|---|
| pre-shared key | 1 |
| DSS signatures | 2 |
| RSA signatures | 3 |
| Encryption with RSA | 4 |
| Revised encryption with RSA | 5 |

Public Certificate Assigned Numbers (PCAN)

Each PCAN is assigned to a specific Authentication Method assigned a CTAN.

- pre-shared key
  No certificates for pre-shared keys are defined.

- DSS signatures
  No certificates for DSS signatures are defined.

| - RSA signatures | |
|---|---|
| PEM encoded | 1 |
| DER encoded | 2 |

- Encryption with RSA
  Same as RSA signatures

- Revised encryption with RSA
  Same as RSA signatures

B Author Contact Information

Adrian Perrig
Carnegie Mellon University
1301 Hamerschlag Hall
5000 Forbes Avenue

Pittsburgh, PA 15218
US
perrig@cmu.edu

Ran Canetti
IBM Research
30 Saw Mill River Rd
Hawthorne, NY 10532
US
canetti@watson.ibm.com

Bram Whillock
Carnegie Mellon University
3150 La Mesa Dr.
San Carlos, CA 94070
US
bwhilloc@andrew.cmu.edu

C Full Copyright Statement