Network Working Group
Internet-Draft
Expires: April 9, 2006

T. Otto
TU Braunschweig
October 6, 2005

The EAP-SKL protocol
draft-otto-eap-skl-03.txt

Status of this Memo

Copyright Notice

Abstract

This document describes EAP-SKL, an Extensible Authentication Protocol (EAP) method which provides mutual authentication and key derivation based on a pre-shared secret. EAP-SKL relies on the cryptographic protocol SKEME (Secure Key Exchange MEchanism protocol), and hence may benefit from its simplicity and the provable security. EAP-SKL complies with the mandatory requirements for an EAP method which is intented for deployment in Wireless LAN environments.

Table of Contents

1. Introduction

IEEE 802.11i makes use of IEEE 802.1X which in turn chooses the Extensible Authentication Protocol (EAP) for carrying authentication messages. EAP itself is only a framework, and runs over various link layers, like IEEE 802.3 Ethernet, IEEE 802.11 Wireless LAN or PPP. In fact, the proper authentication is performed by so called EAP methods.

In 1998, when RFC 2284 was adopted, an usage of EAP outside of wired networks was not taken into account. EAP was thoroughly reviewed, which yielded in May 2004 in RFC 3748. For compatibility reasons, these three EAP methods are still part of the EAP specification, even if declared as deprecated. EAP-MD5, for instance, lacks of mutual authentication and derivation of session keying material, as mandated in [RFC3748].

It has been widely recognized that this state is dissatisfying. In particular, it lacks of a pre-shared key EAP method, which is simple (that is lightweight in terms of bandwidth, latency and computational costs as well as easy in deployment) and, at the same time, provides enough security strength to be used in hostile environments like the Internet or Wireless LAN. Such an EAP method could possibly serve as a replacement for the deprecated EAP-MD5.

EAP-SKL attempts to be an straightforward EAP method with as much features as needed but as few as possible. More precisely, EAP-SKL takes all mandatory requirements into account, with main focus on mutual authentication and adequate session key derivation. Current pre-shared key EAP methods are EAP-FAST, EAP-SIM/AKA, EAP-PSK, EAP-PAX, EAP-TLS (with appropriate extensions) as well as EAP-IKEv2, where EAP-SKL claims to be the simplest one. Further informations about EAP methods can be found in the comprehensive compilation [I-D.bersani-eap-synthesis-sharedkeymethods].

Furthermore, EAP-SKL relies on only one cryptographic primitive,

namely the hash function SHA-1 ([FIPS.180-1.1994]) and optionally the Diffie-Hellman key agreement protocol ([RFC2631]). The choice of SKEME as an appropriate key establishment protocol makes it possible to fully abstain from block ciphers and stream ciphers. Of course, this restricts EAP-SKL's ability to solely perform the mutual authentication task, and does not allow for encrypted communication, like, for instance, EAP-PSK does so. The author is convinced that there are many deployment scenarios, where indeed nothing more than minimalistic functionality is required and desired. Consequently, and in contrast to many other EAP methods, EAP-SKL does not implement, for instance, version negotiation, fragmentation, session resuming, protected result indication or end-user identity hiding.

Nevertheless, there is a single feature which is highly desireable in some cases, namely Perfect Forward Secrecy.

The rest of this document is organized as follows. In the remaining section 1, some terminology is introduced. Section 2 highlights the cryptographic background of EAP-SKL. Section 3 and 4 cover the EAP-SKL protocol and the packet format of the messages. Section 5 and 6 are IANA Considerations and Security Considerations, respectively.

1.1. Requirements notation

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 [RFC2119].

1.2. EAP Terminology

This section contains variables and abbreviations which will be referenced in later sections. For the exact meaning of the roles "peer", "Supplicant", "backend authentication server" and "EAP server" please refer to the EAP specification [RFC3748]. In this document, the words peer and client are used equivalently. The EAP conversation takes place between the client and the EAP server.

Ko: 160-bit symmetric key; shared between client and EAP server. Ko is the most longlife credential within EAP-SKL.

MK: Master Key; shared between the client and EAP server from which all other EAP method session keys are derived.

MSK: Master Session Key; generated from the MK and exported by the EAP method to the authenticator.

EMSK: Extended Master Session Key; also generated from the MK. It contains additional keying material.

G: public Diffie-Hellman group

g: public generator of group G

x: 256-bit nonce generated by the client

y: 256-bit nonce generated by the server

.: (dot); string or binary data concatenation.

^: modular exponentiation

"": NULL string

PRF: Pseudo-random function

F_Ko: pseudorandom function (PRF) keyed with Ko

H: hash function

SK: session key

id_P: client's identity

id_S: EAP server's identity

PFS: Perfect forward secrecy

The corresponding instantiation by EAP-SKL is described in Section 2.

2. Cryptographic Considerations

This section describes the cryptographic design of EAP-SKL. The cryptographic protocol SKEME was chosen to realize mutual authentication and session key establishment. SKEME provides four operational modes, which offers high flexibility. More precisely, it allows the usage of PKIs, KDC and manual (pre-shared) distribution of keying material for an initial keying relationship.

Since EAP-SKL is a pre-shared key authentication method, only the corresponding two modes of SKEME are implemented. The one, from now on referred to as "mode 1", provides Perfect Forward Secrecy (PFS) and hence incorporates a Diffie-Hellman (DH) Key Exchange, which is known to be computational expensive. If this property is not needed, the other mode, referred to as "mode 2", is chosen. This mode

achieves freshness by exchanging random values (nonces), and in absence of asymmetric cryptography mode 2 is notably efficient. In a nutshell, mode 1 provides a better security strength but higher computational costs, whereas mode 2 performs very fast but with fewer security strength. See section 4 for more details.

The notation introduced by [SKEME] is maintained to avoid confusion. There is a collision-resistant hash function, H, a pseudorandom function using key Ko, F_Ko, and as parameters for the DH key exchange g, x, y, m, where g is the generator, m the modulus, and x and y are the client's and EAP server's private DH key, respectively. In EAP-SKL, H is SHA-1 and F_Ko over message M is SHA-1(Ko.M).

In the next section, the two modes of EAP-SKL are described. For mode 2, the session key is computed via

SK =F_Ko(F_Ko(value_S.value_P.id_P.id_S)),

that is, a keyed PRF is applied twice on an input. This idea has finally lead to the HMAC-construction. A HMAC (hash-based MAC) over the input "text" is defined as

HMAC(K;text) = H(K XOR opad . H(K XOR ipad . text)).

The output of SHA-1 and subsequently of HMAC-SHA1 is 160 bit.

As recommended in [RFC2104], the length of K should be equal to the hash output length, less is explicitely discouraged (as itwould decrease the the security strength). Therefore, EAP-SKL mandates the length of Ko to be 160 bit.

Furthermore, SK corresponds to the EAP Master Key (MK), which remains solely within the EAP method.

Finally, the DH key exchange parameters have been chosen in accordance to [RFC3766]. The strength of the DH key exchange is based on the modulus size as well as the exponent size. An 3000-bit MODP public-key scheme corresponds roughly to a 128-bit symmetric-key scheme. EAP-SKL utilizes the 3072-bit MODP Group (id 15), defined in [RFC3526]. According to [RFC3526], the exponent size used in the DH key exchange should have double the entropy of the strength of the entire system, in particular the size of any symmetric key that will

be derived from it (SK, MSK, EMSK). Since EAP-SKL has an effective key strength of 128 bit, the DH exponents (private keys) MUST have at least 256 bit. As cyclic group, g=2 is chosen (has advantage for hardware accelerated computation).

SKEME	EAP-SKL
g	2
m	3072-bit MODP Group
x, y	256 bit
H	SHA-1
F_Ko(M)	SHA-1(Ko.M)
F_Ko(F_Ko(M))	HMAC-SHA1(Ko;arg)

Figure 1

The rest of this section provides further details about the two modes.

Before any run of EAP-SKL, client and EAP server must agree on some 160-bit pre-shared key, referred to as Ko. How Ko is established, is beyond the scope of EAP-SKL and thus must be realized by some secure out-of-band mechanism. At the end of a successful authentication, both client and EAP server have independently derived a session key, SK, in length of 160 bit. This key in turn is adequately expanded to generate the required keying material, that is a 64-byte MSK and a 64-byte EMSK.

This is realized by a pseudorandom function, T-PRF, which is based on SHA-1 and widely accepted. EAP-FAST and PEAP make, for instance, use of this function. Please see Appendix A for internals of T-PRF.

T-PRF is invoked with the parameters (Key, S, OutputLength), where S = label + 0x00 + seed.

EAP-SKL calls T-PRF with the following parameters: TPRF(Ko, "EAP-SKL" +0x00+ SK, 128).

Since T-PRF generates in each iteration 20 byte random data, EAP-SKL

let T-PRF iterate 7 times and then skip the last 12 byte. Figure 2 illustrates the keying material in EAP-SKL.

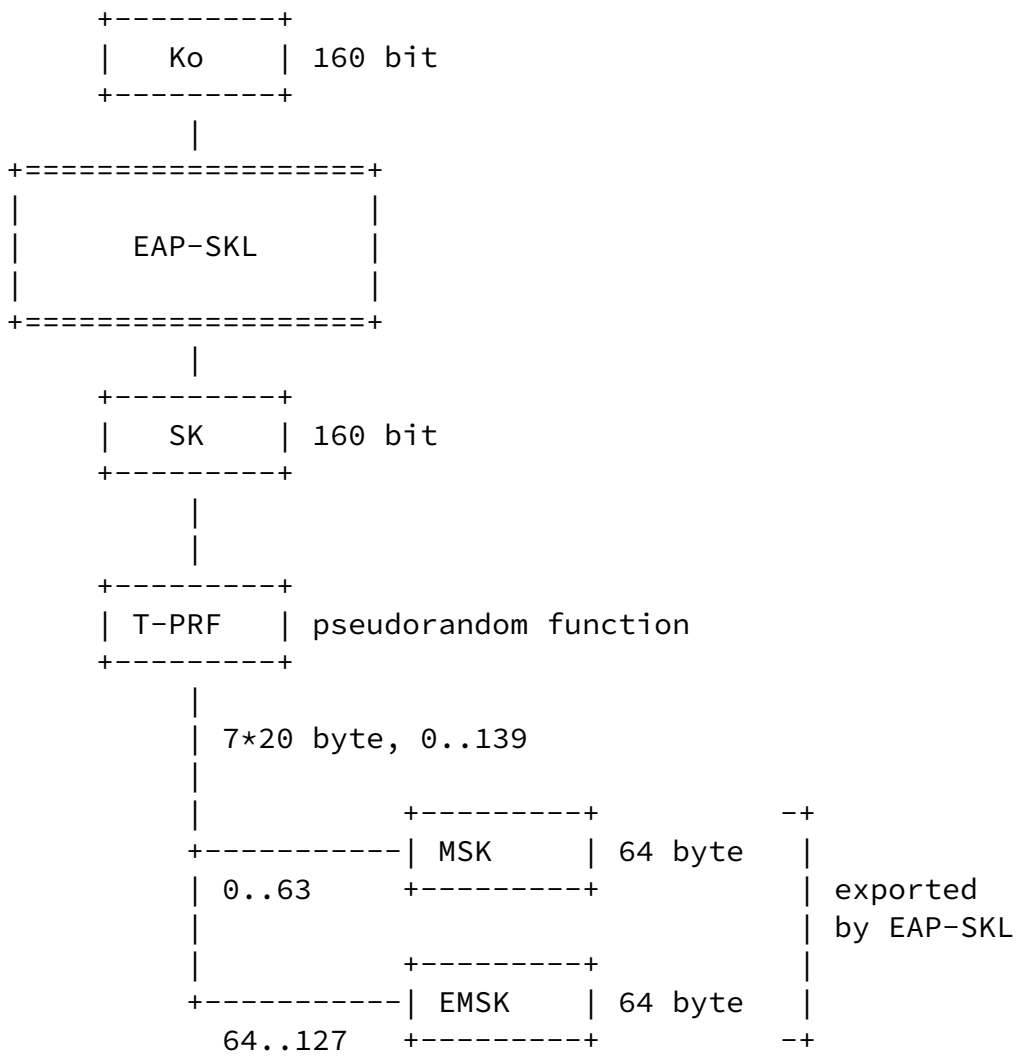

Figure 2

3. Protocol Overview

The message flow of EAP-SKL is shown in Figure 3. The conversation begins with the exchange of EAP Identity messages, where "id" is reserved for special tasks, like routing purposes or session resumption.

```
P <--- S: EAP-Request/Identity                                    (1)
P ---> S: EAP-Response/Identity(id)                               (2)
P <--- S: EAP-Request/EAP-SKL (Start)                             (3)
P ---> S: EAP-Response/EAP-SKL (id_P,value_P)                     (4)
P <--- S: EAP-Request/EAP-SKL(id_S,value_S,
              F_Ko(value_P.value_S.id_S.id_P))                    (5)
P ---> S: EAP-Response(F_Ko(value_S.value_P.id_P.id_S))           (6)
P <--- S: EAP-Success                                             (7)
```

Figure 3

The respective values of value_S and value_P and thus the derivation of SK depend on the chosen mode (Figure 4).

	PFS	value_P	value_S	SK	H(M)	F_Ko(M)
mode 1	yes	g^x	g^y	H(g^xy)	SHA-1(M)	SHA-1(Ko.M)
mode 2	no	nonce_P	nonce_S	F_Ko(arg)		

where arg = F_Ko(value_S.value_P.id_P.id_S).

Figure 4

The choice of the mode is left solely to the EAP server. The client finds this out by the type of the received TLV payload in message (3). If the client does not agree with the proposed mode, he indicates this by sending EAP-Response/Nak. Subsequently, the authentication conversation is aborted.

After the initial identity request/response, the EAP server sends in message 3 an indication which mode to choose. In Message 4, the peer sends his freshness parameter (either a DH public key or a nonce). Message 5 contains the server`s identity, the server`s freshness parameter, and a MAC. This is verified by the peer, if the verification is successful, it responds with message 6, which also contains a MAC. The EAP server does the same computatation. If the

outcome is equal to received MAC, the EAP server is sure to talk to a legitimate client and finish the EAP conversation with an EAP-

Success. Otherwise, the authentication has failed and an EAP-Failure is sent.

The derivation of SK is, in general SK=F_Ko(arg). But for mode 1, [SKEME] offers the alternative SK=H(g^xy), which is chosen here.

For mode 1, it holds SK = H(g^(xy)) = SHA-1(g^xy).

For mode 2, we have

SK = F_Ko(arg) = F_Ko(F_Ko(value_S.value_P.id_P.id_S)) =^! HMAC-SHA1(Ko;arg).

Note that the client has already computed "arg" in message (4) and thus can reuse the result, which would save computational costs.

4. Packet formats

EAP-SKL messages are encapsulated within the payload of EAP-Request and EAP-Response frames, respectively. Therefore, a generic Type-length-value (TLV) attribute format is chosen (figure Figure 5).

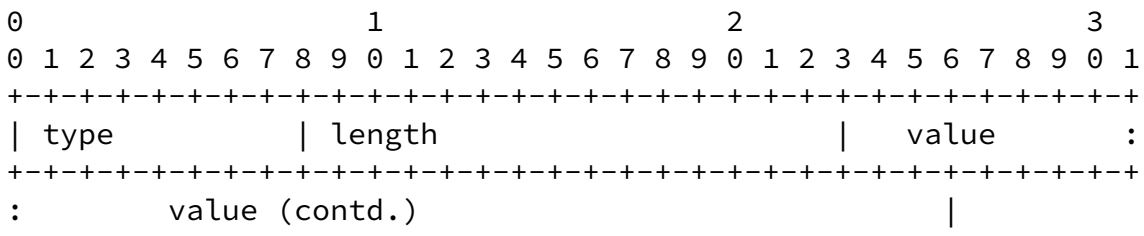

Figure 5

Length contains the length of the TLV in byte.

attribute name	type	used for	payload size
AT_START	0	start	1 byte
AT_ID	1	id_P	max. 607 byte
AT_NONCE	2	nonce_P, nonce_S	384 byte
AT_DH	3	DH public key, g^x, g^y	384 byte
AT_MAC	4	HMAC-SHA1	20 byte

Figure 6

That is, the EAP packet payload of messages (2)-(6) consists of one

or more TLVs, as shown in Figure 7. In case of mode 1, messages (3) and (4) contain AT_DH TLVs, in case of mode 2 AT_NONCE TLVs. AT_START has a payload of one byte, 0x01 for mode 1, 0x02 for mode 2.

```
P <--- S: AT_START                          (3)
P ---> S: AT_ID . AT_NONCE/AT_DH            (4)
P <--- S: AT_ID . AT_NONCE/AT_DH . AT_MAC   (5)
P ---> S: AT_MAC                            (6)
```

Figure 7

5. Computational costs

This section briefly considers the computational costs for the client. Modular exponentiations are known to be computationally expensive, SHA-1 computations are in contrast to this very lightweight.

	mod exp.	SHA-1
mode 1	2	9
mode 2	0	11

Figure 8

In mode 1, the peer performs 9 SHA-1 calculations, namely 1x for computing the MAC in message 6, 1x for verification of the MAC in message 5, and 7x for T-PRF. In mode 2, the peer requires 11 SHA-1 calculations. Overall, especially the computational costs of mode 2 become obvious.

Next, we consider the amount of bytes what each side generates and transmits. Only messages 3 to 6 are analyzed. An EAP-Request or

EAP-Response frame consists of the fields Code, Identifier, Length, Type, which are 1+1+2+1=5 bytes in total, followed by a variable length payload. Only this payload is considered.

size/byte		mode 1	mode 2
peer	msg 4	390+idP	390+idP
	msg 6	23	23
EAP server	msg 3	4	4
	msg 5	413+idS	413+idS

Figure 9

The Diffie-Hellman public key with 384 byte (=3072 bit) size must be considered as heavyweight, compared to the lightweight EAP frame structure. An EAP-Request (resp. Response) has 5+sizeof(payload) bytes length, an EAP-Success or EAP-Failure only 4 bytes.

6. Security Considerations

This section analyzes the security of EAP-SKL. In particular, it is shown that EAP-SKL complies with all mandatory requirements for EAP methods used in IEEE 802.11 wireless LAN deployments, as outlined in RFC 4017.

Mechanism:

EAP-SKL belongs to the category of pre-shared key EAP methods.

Mutual authentication:

EAP-SKL provides mutual authentication. If both parties can verify the received HMAC, the other side is identified and authenticated. Since the MAC is computed over a string which contains fresh random values previously generated, both MACs are

interlocked adequately.

Key derivation:
EAP-SKL generates the required keying material, namely a 64 byte MSK and a 64 byte EMSK, which are exported for derivation of further keying material.

Replay protection:
EAP-SKL is resistant to replay attacks.
Assume an attacker has captured a whole, successful conversation of EAP-SKL. First, we investigate what happens if the attacker wants to impersonate the legitimate client. If value_S of the captured session is different from the current one, the attack fails in computing the MAC in message (4). Suppose now, value_S is the same. It follows that replay of the captures message (4) as well as (6) are recognized as valid ones by the EAP server and thus the attacker will receive an EAP-Success. Unfortunately, he can not derive from HMAC-SHA1(Ko;"success".SK) the session key SK and thus can not derive MSK and EMSK. The attack ends here, that is although the EAP authentication has been successful, subsequent conversation attempts will fail.

This replay attack is avoided by strengthening the server's implementation. The space for value_P is large enough to reject any authentication request with the same value_P. Therefore, the server implementation should maintain a table with all previously received (id_P,value_P) pairs. With this precaution, a replayed session is recognized as a such and appropriate countermeasures

can take place.

Confidentiality:
EAP-SKL does not provide confidentiality in terms of [RFC3748], section 7.2.1. The term "confidentiality" refers to encryption of EAP messages and successful and failure result indications.

Key strength:
The effective key strength of EAP-SKL is 128 bit (see also Section

2).

Resistance to dictionary attacks:
EAP-SKL relies on a pre-shared key, which consists of random data. It is highly discouraged to generate this random data from a dictionary entry by applying some pseudorandom function.

Protection against man-in-the-middle attacks:
If the link layer is a shared media, there is generally a possibility for a third party (MitM) to place between two parties who want to communicate. The trust relation is established by deriving some common secret, which a third party can not derive from the captured conversation. For EAP-SKL, this is the session key SK, whose lifetime is the session. Without knowledge of Ko this common secret can not be derived. And without knowledge of Ko and SK, both MSK and EMSK can not be derived.

Assume now, a MitM sits between peer and EAP Server, and intercepts all server messages and relays them to the peer. This works fine for the whole conversation, that is for messages (1) to (7). But, since the MitM does not know SK, this attack has only denial-of-service character.

Protected ciphersuite negotiation:
EAP-SKL does not support ciphersuite negotiation.

Fast reconnect:
EAP-SKL does not support fast reconnect.

Cryptographic binding:
EAP-SKL performs exactly one authentication and hence this technique is not applicable.

Fragmentation:

An EAP method may assume a mimimum EAP MTU of 1020 byte. Message (4) is the largest one. An EAP Request or Response packet begins with four fields (Code, Identifier, Length and Type), with total 5 byte, followed by the payload. value_P as modulo 3072 can be at most 384 byte, and finally HMAC-SHA1 produces a 20-byte MAC. These are 409 byte, so that AT_ID MAY NOT exceed 1020-409=611 byte. The encapsulated id_P thus may not exceed 611-4=607 byte.

End-user identity hiding:

EAP-SKL does not provide end-user identity hiding. The identity of the client is sent in plaintext.

7. IANA Considerations

This document introduces one new IANA consideration. It requires IANA to allocate a new EAP Type for EAP-SKL.

8. Acknowledgment

The author would like to thank Dieter Stolte for useful comments on the initial version of this draft.

9. References

9.1. Normative References

[FIPS.180-1.1994]
National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-1, May 1994.

[I-D.cam-winget-eap-fast]
Salowey, J., "EAP Flexible Authentication via Secure Tunneling (EAP-FAST)", draft-cam-winget-eap-fast-02 (work in progress), April 2005.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.

[RFC4017] Stanley, D., Walker, J., and B. Aboba, "Extensible

Authentication Protocol (EAP) Method Requirements for Wireless LANs", RFC 4017, March 2005.

[SKEME] Krawczyk, H., "SKEME: A Versatile Secure Key Exchange Mechanism for Internet", June 1996.

9.2. Informative References

[I-D.bersani-eap-psk]
Tschofenig, H. and F. Bersani, "The EAP-PSK Protocol: a Pre-Shared Key EAP Method", draft-bersani-eap-psk-09 (work in progress), August 2005.

[I-D.bersani-eap-synthesis-sharedkeymethods]
Bersani, F., "EAP shared key methods: a tentative

synthesis of those proposed so far", draft-bersani-eap-synthesis-sharedkeymethods-00 (work in progress), April 2004.

[I-D.clancy-eap-pax]
Clancy, C. and W. Arbaugh, "EAP Password Authenticated Exchange", draft-clancy-eap-pax-04 (work in progress), June 2005.

[I-D.tschofenig-eap-ikev2]
Tschofenig, H., "EAP IKEv2 Method (EAP-IKEv2)", draft-tschofenig-eap-ikev2-07 (work in progress), July 2005.

[PRF] Adams, Carlisle et al., "On the Security of Key Derivation Functions, LNCS 3225, Springer", 2004.

[RFC2284] Blunk, L. and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998.

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

Appendix A. T-PRF

The following description of T-PRF corresponds to [I-D.cam-winget-eap-fast], Appendix A.

PRF(key, label, seed) = HMAC-SHA1(key, label + "\0" + seed)

Where '+' indicates concatenation and "\0" is a NULL character. Label is intended to be a unique label for each different use of the T-PRF.

To generate the desired OutputLength octet length of key material, the T-PRF is iterated as follows:

T-PRF (Key, S, OutputLength) = T1 + T2 + T3 + T4 + ...

Where S = label + 0x00 + seed; and

T1 = HMAC-SHA1 (Key, S + OutputLength + 0x01)

```
T2 = HMAC-SHA1 (Key, T1 + S + OutputLength + 0x02)

T3 = HMAC-SHA1 (Key, T2 + S + OutputLength + 0x03)

T4 = HMAC-SHA1 (Key, T3 + S + OutputLength + 0x04)
```

OutputLength is a two octet value that is represented in big endian order. The NULL character, 0x00 shall be present when a label string is provided.

Since each Ti generates 20 byte of keying material, the last Tn is truncated to accommodate the desired length specified by OutputLength. EAP-SKL has to derive 128 byte, so i is set to 7 and the last 12 byte are skipped.

Author's Address

Thomas Otto
TU Braunschweig
38106 Braunschweig
Germany

Phone:
Email: t.otto@tu-bs.de

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in

this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.