Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 30, 2010

D. McGrew
B. Weis
Cisco Systems

February 26, 2010

Key Derivation Functions and their Uses
draft-irtf-cfrg-kdf-uses-00.txt

Abstract

This note surveys the existing designs for Key Derivation Functions (KDFs), the purposes for which they are used, and their security and usability goals. Importantly, some important protocols use KDFs for multiple purposes. We offer conclusions to guide future standards work and research on KDFs.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on August 30, 2010.

Table of Contents

1. Introduction

This section provides background and terminology.

There are many different KDF designs being utilized in the Internet community and beyond. Most of these designs appear in the specification of a protocol or application, and their use is intended only for that particular protocol or application. These protocol-specific KDFs are cataloged in Section 4. There are also some KDFs that are intended for use in multiple applications or protocols; we catalog these protocol-independent KDFs in Section 3. There are different purposes for which KDFs are used, each with its own distinct security goals. We summarize these purposes in Section 2. We take a broad view of what functions can be regarded as KDFs.

Some KDFs use a two-stage process. The first stage is called "extraction" and it takes a variable-length input that is random or pseudorandom, but which may not be uniformly distributed. The second stage is called "expansion", and it takes the fixed-length output of the extraction stage and generates a variable-length pseudorandom output.

An extraction stage can be based on a computational assumption (such as the inability of a computationally limited attacker to find preimages of a certain function, for instance). This case is called a computational extractor. Alternatively, an extraction stage can rely on demonstrable statistical properties of some function (as done with universal hashing, for example). This case is called a statistical extractor.

KDF terminology is non-uniform; they are sometimes called pseudo-random functions (PRFs).

This note is motivated in part by interesting recent work on general-purpose KDFs [I-D.krawczyk-hkdf].

1.1. Conventions Used In This Document

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].

2. Purposes of Key Derivation Functions

The purposes for which KDFs are used can be categorized as follows. KDFs are used to:

1. Generate one or more keys from a (uniformly random or pseudorandom) seed value. In this case, no extraction stage is needed. This use requires that the key provided as input is *uniformly* random or pseudorandom; it is not sufficient that the seed is unpredictable.

2. Generate one or more keys from a Diffie-Hellman shared secret. An extraction stage is needed, because the DH shared secret is not uniformly distributed.

3. Generate key material from a non-uniform random source. In this case, an extraction stage is needed.

4. Generate a key from a passphrase. An extraction stage is needed.

In use case #2, computational extraction seems to be needed, because statistical extraction would be inefficient. In use case #3, either statistical or computational extraction could be used.

KDFs designed for use case #4 usually have additional properties. Some functions dedicated to this purpose are designed to be computationally intensive in order to increase the cost of exhaustive password searching.

It is easy to design a KDF suitable for use #1, because the use case matches that of a pseudorandom function. It is much trickier to design a function suitable for #2 or #3. There has been a good deal of theoretical work in this area, and it is now possible to cite some provable security results, though many of those methods are not currently used in practice (and some methods may be too inefficient to warrant use in practice). It is open to question how much effort should be exerted designing a function for use #4, considering that any system that derives secret keys from passwords will be vulnerable to dictionary attacks.

2.1. Inputs

The following inputs appear in some different KDF designs:

An unpredictable seed value.

The length of derived keying material.

A "salt" value that is different across different runs of the protocol, but which may be public information.

An identifier indicating the use for which the derived keys are intended.

An identifier indicating an entity associated with the seed value and derived key.

Not all KDF designs have all of these inputs.

2.2. Design Goals for KDFs

The essential goal for a KDF is security; if used properly it must meet its security goals. Ideally, it should also be robust against accidental misuse.

It is desirable that a KDF design be simple. Complexity should be avoided in the specification, implementation, and the interface to the KDF.

It is desirable that a KDF be reusable, that is, that a KDF design can be used across multiple applications.

A protocol or application that uses a KDF should be able to easily replace the KDF that it uses; i.e. it should have KDF agility. IKE and TLS have this property to some extent.

It should be easy to validate a KDF implementation as being correct. To this end, a KDF design should define test cases that can be used in known-answer tests.

3. Protocol Independent KDFs

The following KDF definitions have been specified, and are used by one or more of the networking standards documented later in this memo.

3.1. HKDF

The HMAC-based Extract-and-Expand Key Derivation Function (HKDF) [I-D.krawczyk-hkdf] specifies a KDF with an optional extraction phase followed by an expansion phase. The combination of the extraction and expansion steps of HKDF is intended for use cases #2 or #3 (or even #4) whereas the expansion phase by itself is suitable for use case #1.

3.2. NIST SP 800-108

The NIST Recommendation for Key Derivation Using Pseudorandom Functions KDFs [SP800-108] defines three KDFs (using counter mode, feedback mode, and a double-pipeline iteration mode) all of which assume an input of a uniform pseudorandom value, and which require

identifiers for the key usage and optionally accept identifiers for the entities associated with the keys. They were developed specifically for use case #1. These functions use a cryptographic hash function or a block cipher as an underlying primitive.

3.3. NIST SP 800-90

The NIST Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised) [SP800-90] defines KDFs intended for use case #3. Of the inputs that we consider, these functions accept only a seed value. They have additional properties, such as backtracking resistance.

3.4. PKCS #5

The first KDF described by PKCS #5: Password-Based Cryptography Specification Version 2.0 [RFC2898] is PBKDF. It is a hash function in OFB mode. The second KDF described is PBKDF2, which is an HMAC iterated KDF, with all of the iterates XORed together. These functions have the property that they can be computationally expensive, in order to thwart dictionary attacks against keys derived from user-generated passwords.

These KDFs were designed for use case #4, but is seen in other use cases as well.

Notes:

- Neither KDF defined in PKCS #5 matches either the NIST KDF definitions or HKDF. However, it should be noted that none of these KDF definitions were intended to be used for a use case #4.

4. Protocol Specific KDFs

Many Internet and other network protocols have described one or more particular KDFs as part of their specification. This section summarizes how the KDF was designed, and in which of the use cases described in Section 2 each KDF is deployed. Particular attention is paid as to whether the KDF meets the requirements of the specifications in the previous section ([SP800-108], [I-D.krawczyk-hkdf], Section 3.4) or Section 5 of [SP800-56A].

4.1. NIST SP 800-56A

The NIST Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography [SP800-108] describes two counter mode KDFs. They are similar, but differ in that one is intended for

use with ASN.1 encodings. These KDFs take a Diffie-Hellman shared secret as a seed input, and are intended for use case #2. They require as input identifiers for the key usage and identifiers for the entities associated with the key establishment protocol.

4.2. ANSI X9.42

The ANSI Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography [X9.42] defines a hash function KDF. It is intended for use cases #1 and #2.

4.3. ANSI X9.63

The ANSI Public Key Cryptography for the Financial Services Industry Key Agreement and Key Transport Using Elliptic Curve Cryptography [X9.63] defines a hash function in counter mode. It is intended for use cases #1 and #2.

4.4. IEEE 1363-2000

The IEEE Standard Specifications for Public-Key Cryptography [IEEE1363] defines KDF1, a key derivation function used with public key key agreement schemes. It describes a keyed hash of the DH result and a string. This is use case #2.

Notes:

- KDF1 nearly matches SP 800-56A, but it does not include a counter input. The standard also does not clearly define how the key derivation parameters associated with the session are determined.

- The KDF1 specification says that it was derived from ANSI X9.42.

- KDF1 does not match HKDF. It does not include an Extract function, but nearly matches the Expand function (again, the absent counter input) as long as the output only requires one iteration of the HKDF HMAC-Hash function.

4.5. IEEE 1363a-2004

Am amendment to IEEE 1363 [IEEE1363a] adds a new KDF2 to the IEEE 1363 standard. It describes a KDF that is a hash function in counter mode. This is use case #2.

Notes:

- KDF2 matches SP 800-56A (other than the ordinal position of the counter differs). However, the standard does not clearly define how the key derivation parameters associated with the session are determined.

- The KDF2 specification says that it was derived from ANSI X9.42-2001 and ANSI X9.63.

- KDF2 does not match HKDF. It does not include an Extract function, and the Expand function is a counter mode KDF, not a feedback mode KDF.

4.6. ISO-18033-2

The KDFs defined in Encryption algorithms -- Part 2: Asymmetric ciphers [ISO-18033-2] are a more general version of those those in SP 800-56A; the latter specification mandates inputs that are not present in the ISO-18033-2 specification. The ISO-18033-2 KDFs are intended to be used with a DH result as input, which make them suitable for use case #2.

4.7. TLS Version 1.2

P_SHA256 is defined in Section 5 of The Transport Layer Security (TLS) Protocol, Version 1.2 [RFC5246]. This KDF uses HMAC in output feedback mode. Two KDF computations are specified: deriving the "master_secret" from the "pre_master_secret", followed by derivation of session keys (the "key_block") from the "master_secret". The KDF is applied as both use cases #1 and #2, as follows:

- Use case #1 occurs when the pre_master_secret is distributed by the client (Section 8.1.1 of RFC 5246).

- Use case #2 occurs when the pre_master_secret is derived from Diffie-Hellman (Section 8.1.2 of RFC 5246).

- Use case #1 occurs when the key_block is computed from the master_secret (Section 6.3 of RFC 5246).

Notes:

- When P_SHA256 is used for a use case #1, it nearly matches the "KDF in Feedback Mode" specified in SP 800-108. It is missing only the L ("length of the derived keying material") input.

- The feedback construction of P_SHA256 does not conform to either Concatenation Key Derivation Function defined in SP 800-56A. Also, PSHA256 uses an HMAC construction whereas SP 800-56A describes a

keyed hash function.

- When P_SHA256 is used for a use case #1, it nearly matches the HKDF-Expand function, but is missing the counter input).

- For use case #2, P-SHA256 does not conform to HKDF. It specifies an HMAC of the seed rather than a HKDF-Extract function, and omits the counter input to HKDF-Expand.

4.8. TLS Version 1.1

This KDF is the result of P_MD5 XOR P_SHA1 (Section 5 of The Transport Layer Security (TLS) Protocol Version 1.1 [RFC4346].

This KDF is an HMAC in an OFB mode variant. An HMAC PRF is called twice for each derivation: once with MD5 as the hash function and once with SHA1 as the hash function. The results of the two PRF calls are XORed together and become the output of the KDF. The KDF is applied as both use cases #1 and #2, as described for TLS 1.2:

Notes:

- The P_MD5 XOR P_SHA1 does not match either SP 800-108 or SP 800-56A. The use of MD5 is strongly discouraged, in general, though this use of MD5 is believed to provide adequate security.

- The P_MD5 XOR P_SHA1 does not match HKDF.

4.9. IKEv1

The Internet Key Exchange (IKE) [RFC2409] defines a KDF, which it calls a PRF. The "HMAC version of the negotiated hash algorithm" is used as the default KDF (Section 4 of RFC 2409). Hash algorithms currently defined int the IANA IPsec Registry include SHA-1, SHA-256, SHA-384, and SHA-512. The KDF used to protect IKEv1 messages ("phase 1 prf") is composed of two serialized PRF operations, where the data given to the PRF depends on the type of authentication used in the IKEv1 exchange. The KDF is applied as use case #1 or #2 depending on the type of authentication.

IKEv1 also generates keying material for IPsec using a PRF, where the keying material used with the PRF is the result of the phase 1 PRF.

Notes:

- None of the KDFs match either SP 800-108 or SP 800-56A.

- None of the KDFs match HKDF.

4.10. IKEv2: HMAC

The Internet Key Exchange (IKEv2) Protocol[RFC4306] defines a KDF, which it calls PRF+ (Section 2.13 of RFC 4306). PRF+ uses a negotiated hash algorithm. In this section we consider negotiated HMAC algorithms, which used creates a KDF that is an HMAC in an OFB mode. It is first used to extract the SKEYSEED from a shared Diffie-Hellman result (use case #1). The KDF is then used to create a set of shared keys using SKEYSEED (use case #2).

Notes:

- PRF+ strictly conforms to HKDF: deriving SKEYSEED matches the HKDF-Extract step, and derivation of the shared keys matches the HKDF-Expand step.

- Deriving SKEYSEED does not conform to SP 800-56A: it specifies the use of an HMAC (where nonces are used as the "key" input) rather than a hash. Also, the it is missing data inputs of a counter and "other information".

- Deriving the set of shared keys from SKEYSEED very nearly conforms to SP 800-108 but the inputs are different: a counter is included but concatenated after S, S is not necesarily a null terminated string, and the ("length of the derived keying material") input is missing.

4.11. IKEv2: CMAC

A CMAC hash function has been defined for IKEv2: The Advanced Encryption Standard- Cipher-based Message Authentication Code-Pseudo-Random Function-128 (AES-CMAC-PRF-128) [RFC4615]. This KDF can be used as an alternative to the HMAC methods described in the previous IKEv2 section; it is used identically.

This KDF uses AES with a fixed key, and offers no theoretical justification for this practice. This makes it vulnerable to a related-key attack, making it unsuitable for general-purpose use.

Note:

- Deriving SKEYSEED does not conform to SP 800-56A (as described in the previous section).

- Deriving the set of shared keys from SKEYSEED very nearly conforms to SP 800-108 but the inputs are different (as described in the

previous section).

- o HKDF only supports the use of HMAC-based key derivation functions, so AES-CMAC-PRF-128 does not conform to HKDF.

4.12. SSH

The Secure Shell (SSH) Transport Layer Protocol [RFC4253] KDF is defined in Section 7.2 of RFC 4253. The input keying material to the KDF is a Diffie-Hellman result, which is use case #2.

Notes:

- o The SSH KDF does not conform to SP 800-56A. A hash function is used to expand the DH result and it uses a counter beginning at 0x65, which are necessary components to conform. However, the ordinal positions of the counter inputs does not match SP 800-56A. Also, the other inputs (H and session_id) do not match either SP 800-56A definition, although it can be noted that the session_id argument was generated from identifiers unique to the two communicating parties.

4.13. SRTP

The Secure Real-time Transport Protocol [RFC3711] defined AES-CM-PRF, which construction is defined in Section 4.3. This KDF uses AES in counter (CTR) mode to produce keying material.

A counter mode PRF requires a uniformly random key, so without any further processing must fall into use case #1. Most key management standards (e.g., RFC 4568) for SRTP do result in a uniformly random secret key.

An emerging key management method for SRTP [I-D.ietf-avt-dtls-srtp] uses the TLS KDF to provide a shared secret to SRTP. In the cases where the TLS shared secret is a secret key, use case #1 applies. However, in the case of a Diffie-Hellman result being the source of the TLS pre-master-secret, this would be considered a use case #2 (even though the master_secret used as source to the SRTP KDF is a uniformly random key).

Notes:

- o The AES-CM-PRF KDF is based on a counter mode cipher, which does not conform to SP 800-56A or SP 800-108, or HKDF.

4.14. S/MIME DH

The S/MIME Diffie-Hellman Key Agreement Method (RFC 2631) contains a KDF in Section 2.1.2. Two inputs are given to the SHA-1 hash function: The DH result, and a set of "other information" specific to the communicating parties encoded as ASN.1. This is strictly a use case #2.

Notes:

- The KDF matches SP 800-56A Approved Alternative 2, with the exception that the counter is included included in the "OtherInfo" input rather than as an explicit argument to the hash function.

- The KDF does not match the specification of HKDF.

4.15. CMS

The Cryptographic Message Syntax (CMS) Algorithms [RFC3770] describes the use of PBKDF2. See Section 3.4.

4.16. S/MIME

The Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification (RFC 5751) specifies the KDF in CMS [RFC5754]. See Section 3.4.

4.17. Kerberos

A Pseudo-Random Function (PRF) for the Kerberos V Generic Security Service Application Program Interface (GSS-API) Mechanism [RFC4402] defines a KDF described as "PRF+". It is a pseudorandom function in counter mode, where the pseudorandom function is defined as part of a cryptographic profile. RFC 3962 [RFC3962] defines an AES profile that defines its pseudorandom function as PBKDF2 with a HMAC SHA-1 hash. See Section 3.4.

4.18. OpenPGP

String-to-Key (S2K) Function in "OpenPGP Message Format" [RFC4880]. Hash function in counter mode. The preferred implementation takes a salt and passphrase as inputs, and is iterated. In this mode a configured hash function (e.g., SHA256).

Notes:

- The KDF does not match either the NIST KDF definitions or HKDF.

5. Conclusions

There are many KDF designs, and there is discouragingly little compatibility between them. Even where KDFs are similar, they most often have incompatibilities that would prevent interoperability. On the positive side, it appears that there are many ways to design adequately secure KDFs.

The most prevalent Internet key management protocols, IKE and TLS, use their KDFs for both use cases #1 and #2. This fact shows the value of a multi-purpose KDF, such as HKDF. Ideally, any KDF designed for multi-purpose Internet use should be suitable for both IKE and TLS. (Interestingly, both IKE and TLS use HMAC as an extractor, but IKE uses nonces as the HMAC key and the DH shared secret as the "message" input, while TLS does the reverse.)

One potential demerit of a multi-purpose design is complexity. The benefit of using one KDF for two or three purposes could be negated if the KDF was two or three times as complex as each of the single-purpose KDFs that it is intended to replace.

In order to promote KDF reusability and agility, it may be worthwhile to devise a standard interface between a KDF and a protocol. Since the complexity of a KDF increases as the number of its inputs increases, any such standard should avoid the temptation to proliferate inputs. (Consider, for instance, the need to validate an implementation across the range of admissible inputs.)

There are a great many KDFs designed for use in Diffie-Hellman protocols (use case #2), all of which use hash functions, some in the HMAC construction.

Any KDF suitable for use case #3 is likely to be suitable for use case #2. It could be valuable if the existing protocol-independent KDFs that target use case #3 [SP800-90] are found to be suitable for use case #2.

5.1. Future Work

The importance of KDF security warrants more attention; it would be useful to reference and summarize theoretical analyses of

There are no standards that make use of statistical extractors, though there have been a number of theoretical results in the area. One promising approach is the use of statistical extractors for

Diffie-Hellman protocols, in which the extractor is tailored to the specific mathematical group in which the DH protocol is operating. For instance, an elliptic curve group would use a specific extractor, and a particular group of integers modulo a prime would use a different extractor. To use this approach in a protocol, it would be necessary to associate an extractor with a mathematical group; extraction could be incorporated into the group-specific computations. There would still need to be an expand stage. It may be useful to add a statistical extractor to an existing protocol as an additional step, in order to add robustness to the extraction stage, while retaining the existing KDF. In this scenario, the existing KDF is relieved of meeting the security goals of use case #2, and instead need only meet those of use case #1.

Further analysis of the KDFs referenced in this note would be useful. For each of the KDFs referenced in this note, it would be good to know

- Under what conditions is it secure? If it relies on a computational assumption, what is that assumption? If it relies on a statistical extractor, what are the requirements of the inputs of that extractor?

- Are there test cases for the KDF?

- If the KDF is used in a protocol or application, does that protocol or application admit the replacement of the KDF? If so, what is its interface (does it take a "salt" input, for example).

There are more KDFs that could be analyzed and included in future versions of this note, but we believe that analysis of these other less-used KDFs would not change our conclusions.

6. Security Considerations

The KDF selected for use should meet or exceed the security requirements of its particular usage.

7. IANA Considerations

This note has no actions for IANA. This section should be removed by the RFC editor before publication as an RFC.

8. References

8.1. Normative References

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

8.2. Informative References

[I-D.ietf-avt-dtls-srtp]
McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for Secure Real-time Transport Protocol (SRTP)", draft-ietf-avt-dtls-srtp-03 (work in progress), July 2008.

[I-D.krawczyk-hkdf]
Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", draft-krawczyk-hkdf-01 (work in progress), January 2010.

[IEEE1363]
"IEEE Standard Specifications for Public-Key Cryptography", IEEE Std 1363-2000 , January 2000.

[IEEE1363a]
"IEEE Standard Specifications for Public-Key Cryptography - Amendment 1: Additional Techniques", IEEE Std 1363a-2004 , March 2004.

[ISO-18033-2]
"Information technology -- Security techniques -- Encryption algorithms -- Part 2: Asymmetric ciphers", International Organization for Standardization , May 2006.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2898] Kaliski, B., "PKCS #5: Password-Based Cryptography Specification Version 2.0", RFC 2898, September 2000.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC3770] Housley, R. and T. Moore, "Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol (PPP) and Wireless Local Area Networks (WLAN)", RFC 3770, May 2004.

[RFC3962] Raeburn, K., "Advanced Encryption Standard (AES)

Encryption for Kerberos 5", RFC 3962, February 2005.

[RFC4253] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[RFC4402] Williams, N., "A Pseudo-Random Function (PRF) for the Kerberos V Generic Security Service Application Program Interface (GSS-API) Mechanism", RFC 4402, February 2006.

[RFC4615] Song, J., Poovendran, R., Lee, J., and T. Iwata, "The Advanced Encryption Standard-Cipher-based Message Authentication Code-Pseudo-Random Function-128 (AES-CMAC-PRF-128) Algorithm for the Internet Key Exchange Protocol (IKE)", RFC 4615, August 2006.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, November 2007.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5754] Turner, S., "Using SHA2 Algorithms with Cryptographic Message Syntax", RFC 5754, January 2010.

[SP800-108]
Chen, L., "Special Publication 800-108: Recommendation for Key Derivation Using Pseudorandom Functions", U.S. National Institute of Standards and Technology (NIST) Special Publication (SP) 800-108, 2008, <http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf>.

[SP800-56A]
Barker, E., Johnson, D., and M. Smid, "Special Publication 800-56A: Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised)", U.S. National Institute of Standards and Technology (NIST) Special Publication (SP) 800-56A, <http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-2007.pd>.

[SP800-90]

Barker, E. and J. Kelsey, "Special Publication 800-90: Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)", U.S. National Institute of Standards and Technology (NIST) Special Publication (SP) 800-90, 2007, <http://csrc.nist.gov/publications/nistpubs/800-90/SP800-90revised_March2007.pdf>.

[X9.42] "Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography", American National Standard for Financial Services , November 2003.

[X9.63] "Public Key Cryptography for the Financial Services Industry Key Agreement and Key Transport Using Elliptic Curve Cryptography", American National Standard for Financial Services , November 2001.

Authors' Addresses

David A. McGrew
Cisco Systems
510 McCarthy Blvd.
Milpitas, CA 95035
USA

Phone: (408) 525 8651
Email: mcgrew@cisco.com
URI: http://www.mindspring.com/~dmcgrew/dam.htm

Brian Weis
Cisco Systems
510 McCarthy Blvd.
Milpitas, CA 95035
USA

Phone: (408) 526 4796
Email: bew@cisco.com