

Network Working Group  
Request for Comments: 2792  
Category: Informational

M. Blaze  
J. Ioannidis  
AT&T Labs - Research  
A. Keromytis  
U. of Pennsylvania  
March 2000

## DSA and RSA Key and Signature Encoding for the KeyNote Trust Management System

### Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

### Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

### Abstract

This memo describes RSA and DSA key and signature encoding, and binary key encoding for version 2 of the KeyNote trust-management system.

### 1. Introduction

KeyNote is a simple and flexible trust-management system designed to work well for a variety of large- and small-scale Internet-based applications. It provides a single, unified language for both local policies and credentials. KeyNote policies and credentials, called 'assertions', contain predicates that describe the trusted actions permitted by the holders of specific public keys. KeyNote assertions are essentially small, highly-structured programs. A signed assertion, which can be sent over an untrusted network, is also called a 'credential assertion'. Credential assertions, which also serve the role of certificates, have the same syntax as policy assertions but are also signed by the principal delegating the trust. For more details on KeyNote, see [BFIK99]. This document assumes reader familiarity with the KeyNote system.

Cryptographic keys may be used in KeyNote to identify principals. To facilitate interoperation between different implementations and to allow for maximal flexibility, keys must be converted to a normalized canonical form (depended on the public key algorithm used) for the purposes of any internal comparisons between keys. For example, an

RSA [RSA78] key may be encoded in base64 ASCII in one credential, and in hexadecimal ASCII in another. A KeyNote implementation must internally convert the two encodings to a normalized form that allows for comparison between them. Furthermore, the internal structure of an encoded key must be known for an implementation to correctly decode it.

In some applications, other types of values, such as a passphrase or a random nonce, may be used as principal identifiers. When these identifiers contain characters that may not appear in a string (as defined in [BFIK99]), a simple ASCII encoding is necessary to allow their use inside KeyNote assertions. Note that if the identifier only contains characters that can appear in a string, it may be used as-is. Naturally, such identifiers may not be used to sign an assertion, and thus no related signature encoding is defined.

This document specifies RSA and DSA [DSA94] key and signature encodings, and binary key encodings for use in KeyNote.

## 2. Key Normalized Forms

### 2.1 DSA Key Normalized Form

DSA keys in KeyNote are identified by four values:

- the public value,  $y$
- the  $p$  parameter
- the  $q$  parameter
- the  $g$  parameter

Where the  $y$ ,  $p$ ,  $q$ , and  $g$  are the DSA parameters corresponding to the notation of [Sch96]. These four values together make up the DSA key normalized form used in KeyNote. All DSA key comparisons in KeyNote occur between normalized forms.

### 2.2 RSA Key Normalized Form

RSA keys in KeyNote are identified by two values:

- the public exponent
- the modulus

These two values together make up the RSA key normalized form used in KeyNote. All RSA key comparisons in KeyNote occur between normalized forms.

### 2.3 Binary Identifier Normalized Form

The normalized form of a Binary Identifier is the binary identifier's data. Thus, Binary Identifier comparisons are essentially binary-string comparisons of the Identifier values.

## 3. Key Encoding

### 3.1 DSA Key Encoding

DSA keys in KeyNote are encoded as an ASN1 SEQUENCE of four ASN1 INTEGER objects. The four INTEGER objects are the public value and the  $p$ ,  $q$ , and  $g$  parameters of the DSA key, in that order.

For use in KeyNote credentials, the ASN1 SEQUENCE is then ASCII-encoded (e.g., as a string of hex digits or base64 characters).

DSA keys encoded in this way in KeyNote must be identified by the "dsa-XXX:" algorithm name, where XXX is an ASCII encoding ("hex" or "base64"). Other ASCII encoding schemes may be defined in the future.

### 3.2 RSA Key Encoding

RSA keys in KeyNote are encoded as an ASN1 SEQUENCE of two ASN1 INTEGER objects. The two INTEGER objects are the public exponent and the modulus of the DSA key, in that order.

For use in KeyNote credentials, the ASN1 SEQUENCE is then ASCII-encoded (e.g., as a string of hex digits or base64 characters).

RSA keys encoded in this way in KeyNote must be identified by the "rsa-XXX:" algorithm name, where XXX is an ASCII encoding ("hex" or "base64"). Other ASCII encoding schemes may be defined in the future.

### 3.3 Binary Identifier Encoding

Binary Identifiers in KeyNote are assumed to have no internal encoding, and are treated as a sequence of binary digits. The Binary Identifiers are ASCII-encoded, similarly to RSA or DSA keys.

Binary Identifiers encoded in this way in KeyNote must be identified by the "binary-XXX:" algorithm name, where XXX is an ASCII encoding ("hex" or "base64"). Other ASCII encoding schemes may be defined in the future.

## 4. Signature Computation and Encoding

### 4.1 DSA Signature Computation and Encoding

DSA signatures in KeyNote are computed over the assertion body (starting from the beginning of the first keyword, up to and including the newline character immediately before the "Signature:" keyword) and the signature algorithm name (including the trailing colon character, e.g., "sig-dsa-sha1-base64:").

DSA signatures are then encoded as an ASN1 SEQUENCE of two ASN1 INTEGER objects. The two INTEGER objects are the r and s values of a DSA signature [Sch96], in that order.

For use in KeyNote credentials, the ASN1 SEQUENCE is then ASCII-encoded (as a string of hex digits or base64 characters).

DSA signatures encoded in this way in KeyNote must be identified by the "sig-dsa-XXX-YYY:" algorithm name, where XXX is a hash function name ("sha1", for the SHA1 [SHA1-95] hash function is currently the only hash function that may be used with DSA) and YYY is an ASCII encoding ("hex" or "base64").

### 4.2 RSA Signature Computation and Encoding

RSA signatures in KeyNote are computed over the assertion body (starting from the beginning of the first keyword, up to and including the newline character immediately before the "Signature:" keyword) and the signature algorithm name (including the trailing colon character, e.g., "sig-rsa-sha1-base64:").

RSA signatures are then encoded as an ASN1 OCTET STRING object, containing the signature value.

For use in KeyNote credentials, the ASN1 OCTET STRING is then ASCII-encoded (as a string of hex digits or base64 characters).

RSA signatures encoded in this way in KeyNote must be identified by the "sig-rsa-XXX-YYY:" algorithm name, where XXX is a hash function name ("md5" or "sha1", for the MD5 [Riv92] and SHA1 [SHA1-95] hash algorithms respectively, may be used with RSA) and YYY is an ASCII encoding ("hex" or "base64").

### 4.3 Binary Signature Computation and Encoding

Binary Identifiers are unstructured sequences of binary digits, and are not associated with any cryptographic algorithm. Thus, they may not be used to validate an assertion.

## 5. Security Considerations

This document discusses the format of RSA and DSA keys and signatures and of Binary principal identifiers as used in KeyNote. The security of KeyNote credentials utilizing such keys and credentials is directly dependent on the strength of the related public key algorithms. On the security of KeyNote itself, see [BFIK99].

## 6. IANA Considerations

Per [BFIK99], IANA should provide a registry of reserved algorithm identifiers. The following identifiers are reserved by this document as public key and binary identifier encodings:

- "rsa-hex"
- "rsa-base64"
- "dsa-hex"
- "dsa-base64"
- "binary-hex"
- "binary-base64"

The following identifiers are reserved by this document as signature encodings:

- "sig-rsa-md5-hex"
- "sig-rsa-md5-base64"
- "sig-rsa-sha1-hex"
- "sig-rsa-sha1-base64"
- "sig-dsa-sha1-hex"
- "sig-dsa-sha1-base64"

Note that the double quotes are not part of the algorithm identifiers.

## 7. Acknowledgments

This work was sponsored by the DARPA Information Assurance and Survivability (IA&S) program, under BAA 98-34.

## References

- [Sch96] Bruce Schneier, Applied Cryptography 2nd Edition, John Wiley & Sons, New York, NY, 1996.
- [BFIK99] Blaze, M., Feigenbaum, J., Ioannidis, J. and A. Keromytis, "The KeyNote Trust-Management System Version 2", RFC 2704, September 1999.

- [DSA94] NIST, FIPS PUB 186, "Digital Signature Standard", May 1994.
- [Riv92] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
- [RSA78] R. L. Rivest, A. Shamir, L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, v21n2. pp 120-126, February 1978.
- [SHA1-95] NIST, FIPS PUB 180-1, "Secure Hash Standard", April 1995.  
<http://csrc.nist.gov/fips/fip180-1.txt> (ascii)  
<http://csrc.nist.gov/fips/fip180-1.ps> (postscript)

## Contacts

Comments about this document should be discussed on the  
keynote-users@nsa.research.att.com mailing list.

Questions about this document can also be directed to the authors as  
a group at the keynote@research.att.com alias, or to the individual  
authors at:

Matt Blaze  
AT&T Labs - Research  
180 Park Avenue  
Florham Park, New Jersey 07932-0000

Email: mab@research.att.com

John Ioannidis  
AT&T Labs - Research  
180 Park Avenue  
Florham Park, New Jersey 07932-0000

Email: ji@research.att.com

Angelos D. Keromytis  
Distributed Systems Lab  
CIS Department, University of Pennsylvania  
200 S. 33rd Street  
Philadelphia, Pennsylvania 19104-6389

Email: angelos@cis.upenn.edu

## Full Copyright Statement

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Acknowledgement

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

