Database of Long-Lived Symmetric Cryptographic Keys

draft-ietf-karp-crypto-key-table-03

R. Housley

T. Polk

S. Hartman

D. Zhang

Updates on Abstract

- Clarify that “In addition to describing the schema for the database, this document describes the operations that can be performed on the database as well as the requirements for the security protocols that wish to use the database.”

Updates on Introduction

- Add following statements:
  - The information in the database is used to key the authentication of security protocols such as cryptographic authentication for routing protocols
  - The database avoids the need to build knowledge of any security protocol into key management protocols and minimizes protocol-specific knowledge in operational/ management interfaces
  - Textual conventions are provided for the representation of keys and other identifiers.

Updates on Conceptual Database Structure

- LocalKeyID: replaced by LocalKeyName
- PeerKeyID: replaced by PeerKeyName
- Direction: a new value “disable” is defined
- Interface: clarify that this field may consist of a set of implementation specified strings

Textual Conventions

- Key Names: When a key for a given protocol is identified by an integer key identifier, the associated key name will be represented as lower case hexadecimal integers with the most significant octet first. This integer is padded with leading 0's until the width of the key identifier field in the protocol is reached.
- Keys: A key is represented as a lower-case hexadecimal string with the most significant octet of the key first. The length of this string depends on the associated algorithm and KDF.

Application of the Database in a Security Protocol (1)

- In order to use the key table database in a protocol specification, a protocol, essentially, needs to specify the following information.
  (1) The ways of mapping the information in a key table row to the information needed to produce an outgoing packet
  (2) The ways of locating the peer identifier (a member of the Peers set) and the LocalKeyName inside an incoming packet
  (3) The methods of verifying a packet given a key table row
  (4) The form and validation rules for LocalKeyName and PeerKeyName

Application of the Database in a Security Protocol (2)

(5) The form and validation rules for members of the Peers set

(6) The algorithms and KDFs supported

(7) The form of the Protocol Specific field

(8) The rules for canonicalizing LocalKeyName, PeerKeyName, entries in the Peers set, or ProtocolSpecifics; this may include normalizations such as lower-casing hexadecimal strings

(9) The Indication whether the support for Interfaces is required by this protocol

Discussion Related with TCP-AO

- Security protocols such as TCP-AO [RFC5925] are expected to use per- connection state. Implementations may need to supply keys to the protocol-specific databases as the associated entries in the conceptual database are manipulated.
- Rather than consulting the conceptual database, a security protocol such as TCP-AO may update its own tables as keys are added and removed. In this case, the protocol needs to maintain its own key information.
- TCP-AO implementations are unlikely to make the decision of what interface to use prior to key selection. In this case, the implementations are expected to use the same keying material across all of the interfaces and then require the "all" setting.

END