draft-ietf-karp-framework

J.W. Atwood, G. Lebovitz
KARP WG
2010/03/23
bill@cse.concordia.ca

Structure

- Copy-pasted from karp-roadmap
  - 1. Introduction
  - 2. Common Framework
    - Justification
    - Framework Elements
  - 3. Framework Components
    - KMP, KeyStore, RP Mechanisms
  - 4. Framework APIs
    - KMP-KS, KMP-RP, KS-RP

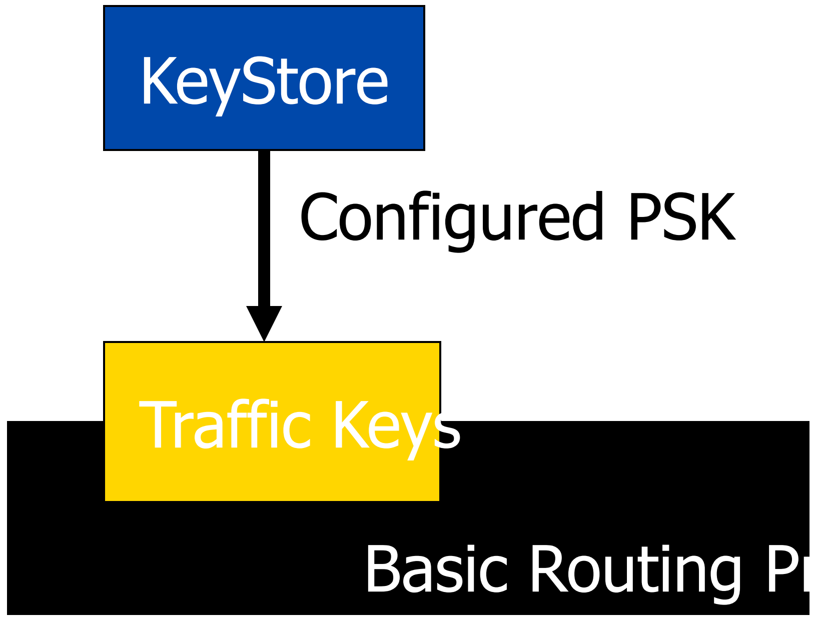
Introduction

- Most of this will vanish, replaced with a reference to the threats-req document.

Common Framework

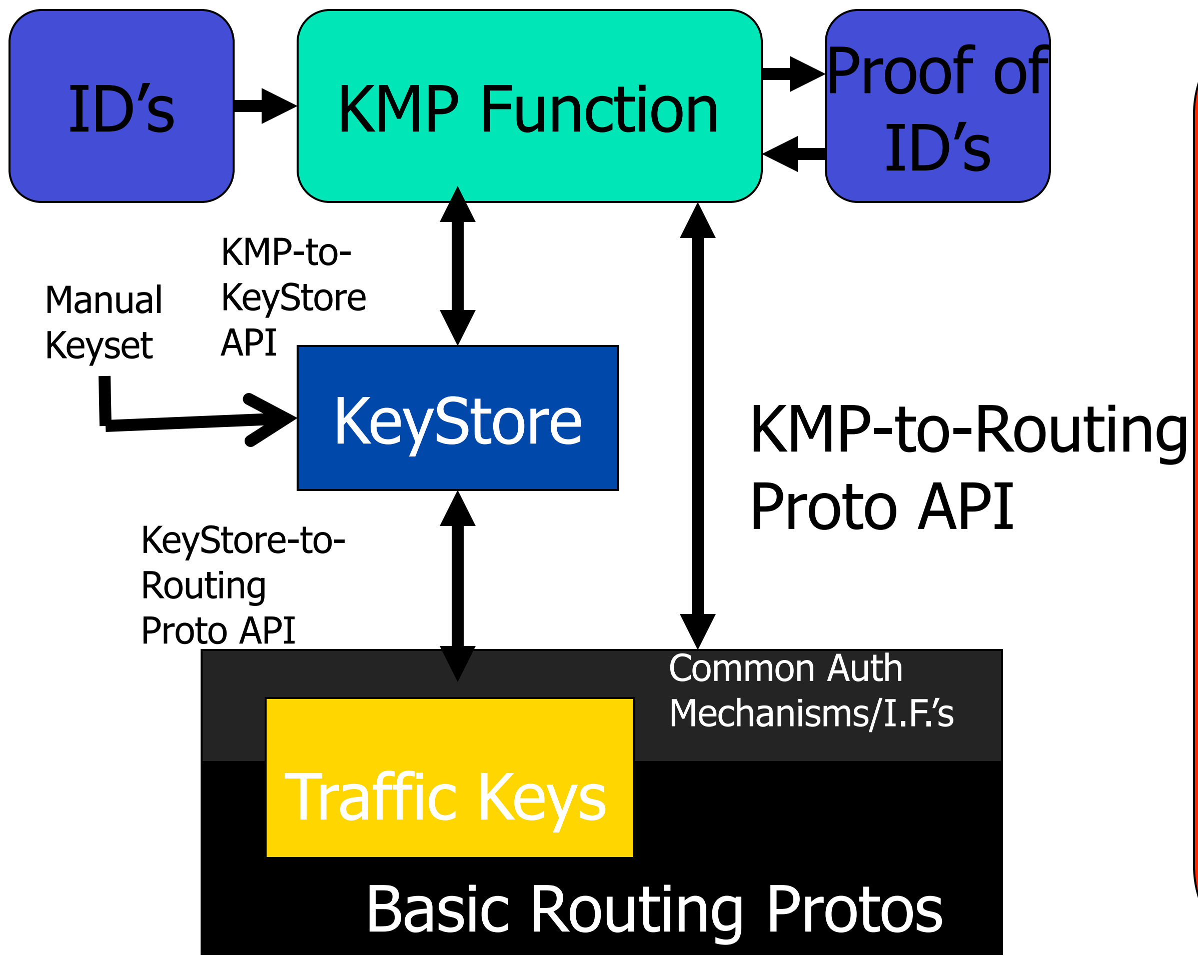
- List of the elements of the framework, along with a figure.

Step 1



1. Define protected elements
2. Strong algos
3. Algo agility
4. Secure use of simple PSK's
5. Inter-conn. replay protection
6. Intra-conn. replay protection
7. Change parameters forces change of traffic keys
8. Use new key within a connection without data loss
9. Efficient re-keying
10. Prevent in-scope DoS
11. Support manual keying
12. All for future use of KMP

Step 2



1. Layer in KMP
2. Define Identifier types/formats
3. Define ID proof mechanisms
4. Re-use KeyStore
5. Re-use Routing Proto's Manual key structure
6. Common Elements:
    1. KeyStore
    2. KeyStore-to-Routing Proto API
    3. KMP-to-KeyStore API
    4. KMP-to-Routing Proto API
    5. KMP Function

Framework Components

- Short descriptions of each component
- Questions going forward:
  - Suck in draft-housley-saag-crypto-key-table & draft-polk-saag-rtg-auth-keytable?
  - Validation of the usefulness of the framework: We need some experience
- Start some protocol-specific efforts and generate message sequences

Framework APIs

- Fairly well-defined as functional descriptions
  - What attributes are passed
  - What the exchange has to accomplish
  - NOT specify actual API code
- An open-source reference implementation would be wonderful.

Going Forward

- Defining the APIs will follow from attempting to produce Message Sequences from examination of actual protocols
  - Protocol-specific design teams
- Volunteers for Reference Example
  - Prototype and write OR
  - Abstractly conceive and write

Ekr Overall Comment

- “There seems to be a basic a assumption throughout these documents that the right design is a decomposed system with separate traffic protection and key management pieces. IMO this has not served us particularly well in IPsec, so I'm not sure why we would want to repeat it. In particular, there are settings where an integrated comsec protocol such as (D)TLS or SSH would be attractive candidates”

Discussion

- Do we want to cleanly separate the traffic protection key and management pieces?
- Is there a good technical reason for why an integrated comsec protocol is better in this application?
- Is there one or more cases where an integrated comsec protocol will **not** work?
- How would an integrated comsec protocol give us the modularity that the RPD teams are asking of this effort?

Ekr Framework Comments

- Why separate the KMP from the data security piece?
- Why define another abstract Key Store concept?
- Claim about security of
  - Self-signed certificates
  - CA signed certificates

..2

- Agreement of parties about configuration information
- I encourage those whose background is more security than mine to respond to these concerns on the list, so that consensus can be achieved.

Questions?