Real Privacy Management™ (RPM)

Cryptographic Description

Version 3.1

Abstract

This paper provides a description of the RPM cryptographic functions.

Cryptographic Description PDAF Bundle

At the heart of the Real Privacy Management™ (RPM) system is an algorithm for generating a sequence of master keys from an initial master key. When the initial master key is known both to a sender and receiver, these communicants can use the generated master keys—more specifically, session and children keys derived from each master key—for encryption and authentication. The algorithm is randomized in the sense that the sender infuses pseudorandom "salt" into each master key in the sequence, with the sender transmitting the encrypted salt to the receiver (truly random salt would also work). The RPM system is primarily a key-management system that uses derived children keys in some standard symmetric encryption algorithm (*e.g.,* AES). Authentication is performed in one of two ways; implicitly: if the recipient decrypts the ciphertext into valid plaintext, then the recipient assumes the message came from the purported sender (RPM assumes there is sufficient redundancy, or some added token, in plaintext messages to enable checking for valid plaintext), or explicitly: the recipient will receive an authentication token created specifically to authenticate the sender. RPM uses two new, fast, low-level *ad-hoc* functions: a combining function h, and an extraction function ρ. In addition, RPM includes various high-level system components (*i.e.,* key stores, message transfer protocol, lost-message protocol, group keying protocol).

RPM aims to provide authentication in a fast and simple fashion, while maintaining adequate security. Toward this goal, the system avoids expensive public-key operations and complex key infrastructures, depends on derived key sequences, and uses randomization. In addition, the combining and extraction functions lose information, complicating the cryptanalytic task of computing a master key from a derived child key.

To understand RPM, it is helpful to examine its four major parts: the low-level cryptographic functions, the master-key generation algorithm, the derivation of session and children keys and their use in encryption, and the high-level system components.

Low-Level Cryptographic Functions - PDAFBundle

Following the security parameter recommendations of Relevant, let $\mathbf{K} = \{0,1\}^{256}$ be the keyspace, and $\mathcal{M}$ be the message space. Each master key $K = (k_1, k_2) \in \mathbf{K}^2$ is a pair of 256-bit keys. RPM assumes some standard encryption algorithm E: $\mathbf{K}^{1/2} \times \mathcal{M} \rightarrow \mathcal{M}$, where $\mathbf{K}^{1/2} = \{0,1\}^{128}$. RPM also assumes some secure pseudorandom number generator G producing values in $\mathbf{K}$ from an initial seed. Let G(seed, i) denote the ith value generated by G.

RPM defines two new cryptographic functions: a *combing function h*: $\mathbf{K}^2 \rightarrow \mathbf{K}$, and *an extraction function* $\rho : \mathbf{K} \rightarrow \mathbf{K}^{1/2}$. To define these functions, we view $\mathbf{K}$ with the blocking $\mathbf{K} \simeq \mathbf{A}^{64}$, where $\mathbf{A} = \{0,1\}^4$. That is, each key in $\mathbf{K}$ can be viewed as an array of 64 hexadecimal numbers. To this end, let $B = 2^4 = 16$ be the *blocksize*. To define the cryptographic functions, let $\boxplus$: $\mathbf{K}^2 \rightarrow \mathbf{K}$, denote component-wise addition modulo B (this operation is similar to exclusive-or $\oplus$).

For any $k \in \mathbf{K}$, define $\rho(k)$ as follows, viewing k as an array of 64 hexadecimal numbers. For each $0 \leq i < 64$, let

$$\rho(k)[i] = \quad k[2i] + k[2i + 1] \mod B \quad . \qquad (1)$$

Similarly, for any $k_1, k_2 \in \mathbf{K}$, define $h(k_1, k_2)$ as follows. For each $0 \leq i < 64$, compute in sequence

$$h(k_1, k_2)[i] = \quad k_1[i] + k_1[i + k_2[i] \mod 64] \mod B. \qquad (2)$$

Master Key Generation

Given any initial master key $K_0 = (k_{0,1}, k_{0,2}) \in \mathbf{K}^2$, and given any initial seed, RPM generates the following sequence of master keys $K_0, K_1, K_2, \ldots, K_i = (k_{i,1}, k_{i,2}), \ldots$ as follows. For each $i \geq 0$, $K_{i+1} = (k_{i+1,1}, k_{i+1,2})$, where

$$k_{i+1,\,1} = h(\ S_i \ , k_{i,2}\ ), \qquad (3)$$

and

$$k_{i+1,\,2} = h(\ k_{i,2} \ , k_{i,1}\ ), \qquad (4)$$

where $S_i = G$(seed, i) is the ith number generated by G.

Deviation of Session and Children Keys and their Use in Encryption

For each master key $K_i = (k_{i,1}, k_{i,2})$, RPM computes the following derived *session key* $x_i \in \mathbf{K}$ and *child key* $w_i \in \mathbf{K}^{1/2}$:

$$x_i = h(\ k_{i,2}\ ,\ S_i\ ) \tag{5}$$

and

$$w_i = \rho(\ x_i\ ). \tag{6}$$

To encrypt any message $m \in \mathcal{m}$, the sender computes the ciphertext $C = E(w_i, m)$ and sends the following message to the recipient:

$$(\ \text{MID, OpenIDSender}, C, S_i \boxplus k_{i,1}\ )\ , \tag{7}$$

where MID is the *message identification number* and OpenIDSender is the publicly-known identification of the sender. The final value $S_i \boxplus k_{i,1}$, called the *open return*, is the encryption of pseudorandom salt S_i.

Another option, called *Option A*, is to encrypt m as $m \boxplus w_i$, using w_i as a keystream in a one-time pad.

<u>Low-Level Cryptographic Function – Combine express</u>

The above combine/extract functions provide a formulation that delivers a one-way underdetermined equation set and the RPM message exchange capability. Two additional low-level functions, Combine Express and Extract Express are as follows that are available for generating underdetermined values underdetermined output values that can be used to seed, update and/or replace existing RPM key values internally to the operator:

Registers :
$R = \{R[0], R[1], \ldots ,R[n\text{-}1]\}$
$K = \{K[0], K[1], \ldots ,K[n\text{-}1]\}$
$A = \{A[0], A[1], \ldots ,A[n\text{-}1]\}$

Initial Value :
$i_{-1} = -1, \quad j_{-1} = -1$

Repeat for k from 0 to n-1 :
$i_k = (i_{k-1} + 1) + K[k] \pmod n$
$j_k = (j_{k-1} + 1) + R[k] \pmod n$
$A[k] = R[i_k] + K[j_k] \pmod B$

Example :
$n = 10; B = 16$
$R = (0123456789)$
$K = (9876543210)$
$A = (2FA3EDA589)$

1. The combining function details: Combine **R** and **K**, resulting in a n-bit ‘alphabet’, **A**
   - 1.1 Select an **R** digit by using the 1st digit of **K** as a pointer into **R** beginning at the 1st digit position and moving **K**’s value in digit positions to the right in **R** where the starting position in **R** is the 0th value position.
   - 1.2 Select a **K** digit by using the 1st digit of **R** as a pointer into **K** beginning at the 1st digit position and moving **R**’s value in digit positions to the right in **K** where the starting position in **K** is the 0th value position.
   - 1.3 Hexadecimal add without carry the selected **R** digit from Step 2.1 and the **K** digit from Step 2.2. This sum is the first digit of the result number, **A**.
   - 1.4 Repeat 1.1, 1.2 and 1.3 using the next digit to the right in **R** and **K** where the starting digits for the steps is one position to the right of the previously selected digit (the 0th value position). Continue until the result **A** is the same length as **R** and **K** (n-bits, 32 4-bit hex numbers for 128-bits).

Example

R = 0123456789 **K** = 9876543210

1.1: 9, using 9 from **K** and selecting 9 in **R**
1.2: 9, using 0 from **R** and selecting 9 in **K**
1.3: **A** first digit is 2 from (9 + 9) Mod 16 = 2

1.1: 8, using 8 from **K** and selecting 8 in **R** having started at the 1st position, which is the first digit position to the right of the previously selected last digit (9)
1.2: 7, using 1 from **R** and selecting 7 in **K** having started at the 2nd position, which is the first digit position to the right of the previously select first digit (9)
1.3: **A** second digit is F from (8 + 7) Mod 16 = F

A = 2FA3EDA589 from
(9+9) Mod 16 = 2
(8+7) Mod 16 = F
(6+4) Mod 16 = A
(3+0) Mod 16 = 3
(9+5) Mod 16 = E
(4+9) Mod 16 = D
(8+2) Mod 16 = A
(1+4) Mod 16 = 5
(3+5) Mod 16 = 8
(4+5) Mod 16 = 9

Low-Level Cryptographic Function – Extract express

Registers :
$A = \{A[0], A[1], \dots, A[n-1]\}$
$K_1 = \{K_1[0], K_1[1], \dots, K_1[n-1]\}$
$W = \{W[0], W[1], \dots, W[n-1]\}$

Initial Value :
$i_{-1} = -1$

Repeat for k from 0 to n-1 :
$i_k = (i_{k-1} + 1) + K_1[k] \pmod n$
$W[k] = A[i_k]$

Another optional child encrption key formulation:
W[k] = A[ik] ⊕ K1[k] (mod B)

2. The extraction function details: Extract n-bit key **W** out of **A** using $\mathbf{K_1}$
   2.1. Select an **A** digit by using the 1st digit of $\mathbf{K_1}$ as a pointer into **A** beginning at the 1st digit position and moving $\mathbf{K_1}$'s value in digit positions to the right in **A** where the starting position in **A** is the 0th value position.
   2.2. Use the selected **A** digit as the first digit of the result number, **W**.
   2.3. Repeat 2.1 and 2.2 using the next digit to the right in $\mathbf{K_1}$ and the starting digits in **A** as one position to the right of the previously selected digit (and this is the 0th value position). Continue until the result **W** is the same length as $\mathbf{K_1}$ and **A** (n-bits, 32 4-bit hex numbers for 128-bit).

Example

A = 2FA3EDA589 $\mathbf{K_1}$ = 9876543210

W = 98A39E8F3E

<u>RPM High-Level System Components</u>

RPM includes additional system components and protocols to facilitate network communications. These components include the following.

Trusted Key Stores
knows the initial master key for every communicant and can relay messages.

Message Transfer Protocol
relays messages through a trusted key store.

Lost-Message Protocol
provides a way for sender and recipient to resynchronize their master key sequence.

Group Keying Protocol
places the same ID-credentials on multiple devices (called *multiple location ID use*).

In addition, the following options can be used.

Message Authentication Code (MAC)
When using encryption Option A, a message authentication tag can be computed as

$tag_i = \rho h(x_i , C_i \square S_i)$, where

x_i is the seesion key, C_i is the ciphertext, and S_i is the pseudorandom salt.

Extraction Function Variants
Variants of the extraction function ρ can be defined by adding pairs of key components separated by a distance of l array positions. The basic definition uses $l = 1$.

Multiple Children Keys Per Session
For enhanced speed, multiple children keys could be derived for each session key using some simple variations of the *h* function [McG05, p. 5].

References

[Adc00] Adcock, Jamison M., David M. Balenson, David W. Carman, Michael Heyman, and Alan T. Sherman, "Trading off strength and performance in network authentication: Experience with the ACSA Project," *Proceedings of the DARPA Information Survivability Conference & Exposition (DISCEX 00)*, IEEE Computer Society (January 25-27, 2000), 127–139.

[McG99] McGough, Paul, "Method and system for performing secure electronic messaging," U.S. Patent 6,002,769, (December 14, 1999).

[McG00] McGough, Paul, "Method and system for performing secure electronic monetary transactions ," U.S. Patent 6,058,189, (May 2, 2000).

[McG02] McGough, Paul, "Method and system for performing secure electronic digital streaming," U.S. Patent 6,445,797, (September 3, 2002).

[McG06] McGough, Paul, "Method and system for performing perfectly secure key exchange and authenticated messaging," U.S. Patent Application 20060034456, (February 16, 2006).

[McG08] McGough, Paul, "Method and system for providing authentication service for Internet users," U.S. Patent Application 20080056501, (March 6, 2008).

[McG08a] McGough, Paul, "Method and system for establishing real-time authenticated and secured communications channels in a public network," U.S. Patent Application 20080065886, (March 13, 2008).

[McG08b] McGough, Paul, "Real Privacy Management Authentication System," U.S. Patent Application 20080184031, (July 31, 2008)

[McG09] McGough, Paul, "RPM Federated Trusted Directory Services," U.S. Provisional Patent, (August 18, 2009)

[RiS82] Rivest, Ronald L., and Alan T. Sherman, "Randomized encryption techniques" in *Advances in Cryptology: Proceedings of Crypto 82*, David Chaum, Ronald L. Rivest, and Alan T. Sherman, eds., Plenum Press (New York, 1983), 145–163.

[Shr05] Sherman, Alan T., "An initial assessment of the 2factor authentication and key-management system: highlights," *unpublished memorandum*, (May 27, 2005), 6 pages.

[ShM03] Sherman, Alan T., and David A. McGrew, David A., "Key establishment in large dynamic groups using one-way function trees," *IEEE Transactions on Software Engineering*, 29:5 (May 2003), 444–458.

[Tan06] Tanaka, Hatsukazu, "Security-function integrated simple cipher communication system," The 2006 Symposium on Cryptography and Information Security (SCIS 2006), The Institute of Electronics, Information and Communication Engineers, (Hiroshima, Japan, January 17 – 20, 2006).

[Tan07] Tanaka, Hatsukazu, "Generation of cryptographic random sequences and its application to secure enciphering," The 2007 Symposium on Cryptography and Information Security (SCIS 2007), The Institute of Electronics, Information and Communication Engineers, (Sasebo, Japan, January 23 – 26, 2007).

[Tan10] Tanaka, Hatsukazu, "Informationally Secure Secret-Key Sharing Scheme Using the Singularity of Source Coding," The 2010 Symposium on Cryptography and Information Security (SCIS 2010), The Institute of Electronics, Information and Communication Engineers, (Takamatsu, Japan, January 19 – 22, 2010).