Viper — Network Security Project

Nicholas Capo

May 2, 2012

Overview

Serpent
Project Proposal
Problem 1: Cryptographers Speak Math
Problem 2: Pseudo-Code Has No Types
Lessons Learned

Serpent

“Serpent is a symmetric key block cipher which was a finalist in the Advanced Encryption Standard (AES) contest, where it came second to Rijndael. Serpent was designed by Ross Anderson, Eli Biham, and Lars Knudsen.”[6]

By using a bit-slice implementation, Serpent can be efficiently implemented on a “processor with two 32-bit integer ALUs (such as the popular Intel MMX series)”.[2, p. 2]

From Wikipedia: “Serpent was designed so that all operations can be executed in parallel, using 32 1-bit slices. This maximizes [the] parallelism [of the algorithm]”[6]

Project Proposal

To develop an implementation of the Serpent-1 Cipher [6][1] in C and Python.

Both implementations of the cipher shall be constructed using threads to take advantage of the parallelism of the Serpent Algorithm. The number of threads to be used (up to a limit) shall be specified at runtime.

Each implementation shall be cipher-text compatible with each other and also the reference implementations. [5]

An Enormous Difference

“There is an enormous difference between a mathematical algorithm and its concrete implementation in hardware or software.

Cryptographic system designs are fragile. Just because a protocol is logically secure doesn’t mean it will stay secure when a designer starts defining message structures and passing bits around.

Close isn’t close enough; these systems must be implemented exactly, perfectly, or they will fail.”
—Bruce Schneier [4]

Problem 1: Cryptographers Speak Math

For Example (from the Specification):
“ Thus the cipher may be formally described by the following equations:

$$\hat{B}_0 := IP(P)$$
$$B_{i+1} := R_i(\hat{B}_i)$$
$$C := FP(\hat{B}_32)$$

where

$$R_i(X) = L(\hat{S}_i(X \oplus \hat{K}_i)) \qquad i = 0, \ldots, 30$$
$$\hat{R}_i(X) = \hat{S}_i(X \oplus \hat{K}_i) \oplus \hat{K}_{32} \qquad i = 31$$

where S_i is the application of the S-box $S_{i\,mod\,8}$ 32 times in parallel, and L is the linear transformation. ”[2, p. 3]

Problem 2: Pseudo-Code Has No Types

“Our cipher requires 132 32-bit words of key material. We first pad the user supplied key to 256 bits, if necessary, as described in section 2.

We then expand it to 33 128-bit subkeys K_0, ..., K_{32} , in the following way.

We write the key K as eight 32-bit words w_{-8}, ..., w_{-1} and expand these to an intermediate key (which we call prekey) w_0, ..., w_{131} by the following affine recurrence:
$w_i := (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi \oplus i) <<< 11$”[2, p. 7]

The Pitfalls of Cryptography

“Building a secure cryptographic system is easy to do badly, and very difficult to do well. Unfortunately, most people can’t tell the difference.

In other areas of computer science, functionality serves to differentiate the good from the bad: a good compression algorithm will work better than a bad one; a bad compression program will look worse in feature-comparison charts. Cryptography is different. Just because an encryption program works doesn’t mean it is secure.

What happens with most products is that someone reads Applied Cryptography, chooses an algorithm and protocol, tests it to make sure it works, and thinks he’s done. He’s not. Functionality does not equal quality, and no amount of beta testing will ever reveal a security flaw. Too many products are merely ”buzzword compliant”; they use secure cryptography, but they are not secure.”
—Bruce Schneier [3]

Lessons Learned

With very few exceptions, and unless you are a well trained cryptographer, you should abide by the following three rules:

Don’t	Write your own encryption algorithm: it will be insecure
Don’t	Re-implement a known algorithm: it will be insecure
Do	Use well-tested implementations of well-tested algorithms
Do	Test for security, not just functionality (maybe hire a Cryptographer or a Security Engineer)

References

*

References

[1] Ross J. Anderson. *Serpent Home Page*. [Online; accessed 26-January-2012]. URL: `https://www.cl.cam.ac.uk/~rja14/serpent.html` (cit. on p. 3).

[2] Ross Anderson, Eli Biham, and Lars Knudsen. *Serpent: A proposal for the Advanced Encryption Standard*. [Online; accessed 18-February-2012]. URL: `http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf` (cit. on pp. 2, 4).

[3] Bruce Schneier. *Security Pitfalls in Cryptography*. [Online; accessed 18-Apr-2012]. 1998. URL: `https://www.schneier.com/essay-028.html` (cit. on p. 5).

[4] Bruce Schneier. *Why Cryptography Is Harder Than It Looks*. [Online; accessed 18-Apr-2012]. 1997. URL: `https://www.schneier.com/essay-037.html` (cit. on p. 3).

[5] Frank Stajano. *Serpent reference implementation*. [Online; accessed 26-January-2012]. URL: `https://www.cl.cam.ac.uk/~fms27/serpent/` (cit. on p. 3).

[6] Wikipedia. *Serpent (cipher) — Wikipedia, The Free Encyclopedia*. [Online; accessed 18-February-2012]. 2012. URL: `http://en.wikipedia.org/w/index.php?title=Serpent_(cipher)&oldid=469573199` (cit. on pp. 2, 3).