SPRITZ—A SPONGY RC4-LIKE STREAM CIPHER AND HASH FUNCTION

Ronald L. Rivest[1] Jacob C. N. Schuldt[2]

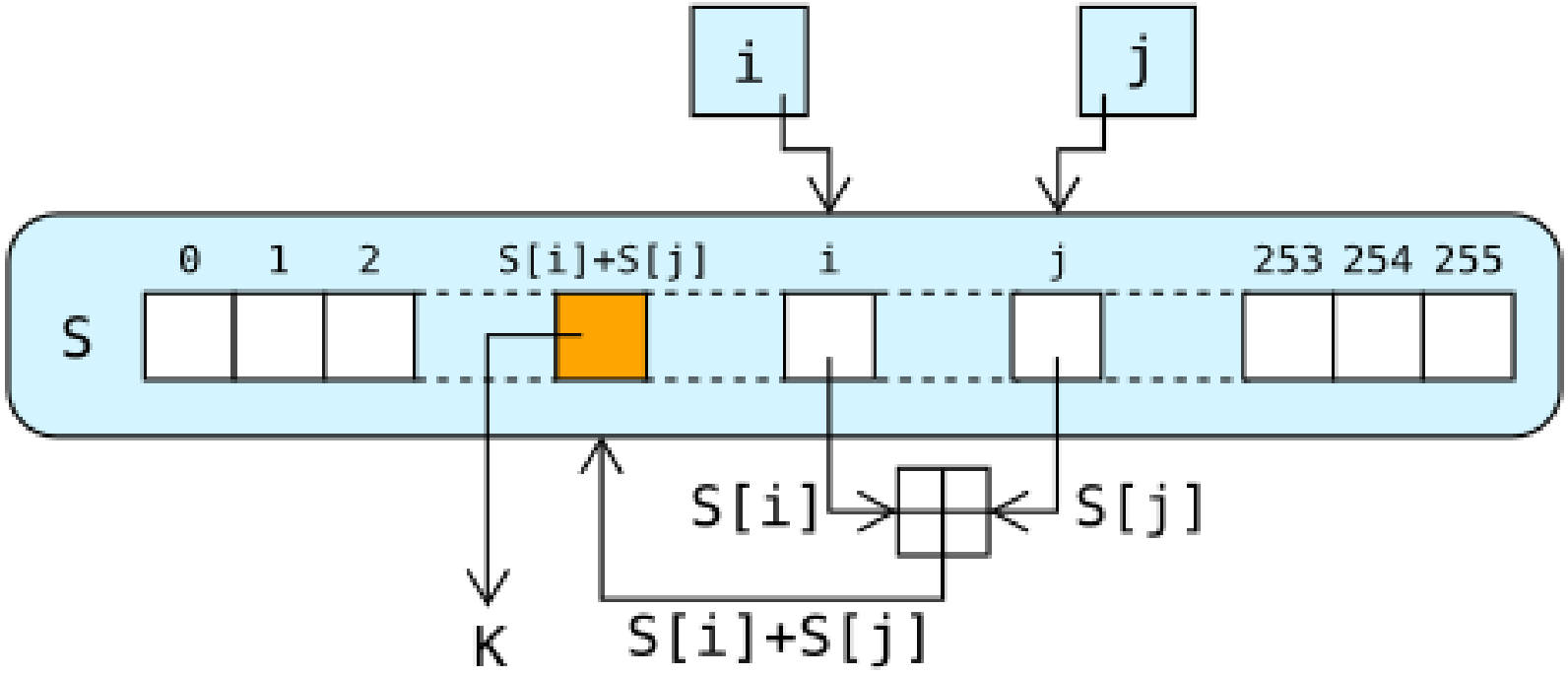
[1]Vannevar Bush Professor of EECS
MIT CSAIL
Cambridge, MA 02139
rivest@mit.edu

[2]Information Security Group
Royal Holloway, University of London
jacob.schuldt@rhul.ac.uk

CRYPTO rump session
August 19, 2014

RC4

- Stream cipher RC4 designed by Rivest (1987).
- Widely used (50% of all TLS connections).
- Simple, fast.
-



(Wikipedia)

RC4 attacks

RC4 has numerous vulnerabilities and “soft spots” [see paper for citations]:

- Key-dependent biases of initial output
- Key collisions (producing same internal state)
- Key recovery possible from known internal state
- Related-key attacks (WEP)
- State recovery from known output (feasible?)
- Output biases; distinguishers

SPRITZ

- We started design after CRYPTO 2013.
- Principles:
  - Drop-in replacement for RC4
  - Retain “RC4 style” (e.g. state is a few registers plus a permutation S of $\{0, 1, \ldots, N-1\}$)
  - Minimize statistical vulnerabilities
  - Redo key-setup entirely
- Automatically examined many thousands of candidates; filtered by syntactic, statistical, and cryptographic criteria. Approximately 5 “core-months” of CPU time used!

Code comparison (Output routines)

Winner is #4933 (postfix for i, j, k, z):

```
i1+, kjiS+S+, ikjS++, jikz+S+S+S
```

RC4()

1 $i = i + 1$
2 $j = j + S[i]$
3 SWAP$(S[i], S[j])$
4 $z = S[S[i] + S[j]]$
5 **return** z

SPRITZ()

1 $i = i + 1$
2 $j = k + S[j + S[i]]$
3 $k = i + k + S[j]$
4 SWAP$(S[i], S[j])$
5 $z = S[j + S[i + S[z + k]]]$
6 **return** z

- About 50% longer
- Uses new register k as well RC4 registers i, j; output register z also used in feedback.

Much better statistics!

- Spritz statistical biases are *much* fainter than for RC4.
- We estimate the biases for $N = 256$ by extrapolating from Spritz with $N = 16, 24, 32$.
- For $N = 256$:
  - Can distinguish RC4-256 from random with 2^{41} samples.
  - Our tests suggest that 2^{81} samples are required to distinguish SPRITZ-256 from random.

But wait, there's more! SPRITZ is spongy!

- SPRITZ is also a (modified) sponge function, and usable as a hash function:

```
1  InitializeState(N)
2  Absorb("abc")      – accept input piecemeal.
3  Absorb("def")
4  Squeeze(32)        – output 32 byte hash.


5  Absorb("ghi")      – keep going...
6  Squeeze(1000)
```

- Large state space (like KECCAK), but also has built-in protection against inference of key from knowledge of internal state (which KECCAK does not).
- (But very much slower than Keccak...)

More...

Our paper on SPRITZ can be found on my web site:

`people.csail.mit.edu/rivest/pubs.html#RS14`

More security review needed; comments and analysis appreciated!