bcgit / **bc-java** Public mirror

mirrored from https://www.bouncycastle.org/repositories/bc-java

Code | Issues 298 | Pull requests 41 | Discussions | Actions | Projects

Commit 94abbd5

peterdettman committed on Mar 4

```
Fix Frodo error sampling to be constant-time
```

main · r1rv84

1 parent 7b094ad commit 94abbd5

2 files changed +31 -22 lines changed

Top

Filter files…

core/src/main/java/org/bouncycastle/pqc/crypto/frodo

- FrodoEngine.java
- Noise.java

2 files changed +31 -22 lines changed

Search within code

...ncycastle/pqc/crypto/frodo/FrodoEngine.java

```
@@ -100,31 +100,10 @@ public FrodoEngine(int n, int D, int B, short[] cdf_table, Xof digest, FrodoMatr
100 100             this.gen = mGen;
101 101         }
102 102
103     -     private short sample(short r)
104     -     {
105     -         short t, e;
106     -         // 1. t = sum_{i=1}^{len_x - 1} r_i * 2^{i-1}
107     -         t = (short) ((r & 0xffff) >>> 1);
108     -         e = 0; // 2. e = 0
```

```
109         -         for (int z = 0; z < T_chi.length; z++)
110         -         {
111         -             if (t > T_chi[z]) // 4. if t > T_chi(z)
112         -                 e++; // 5. e = e + 1
113         -         }
114         -         // 6. e = (-1)^{r_0} * e
115         -
116         -         if (((r & 0xffff) % 2) == 1)
117         -             e = (short) ((e) * (-1) & 0xffff);
118         -
119         -         return e;
120         -     }
121         -
122   103           private short[] sample_matrix(short[] r, int offset, int n1, int n2)
123   104           {
124   105               short[] E = new short[n1 * n2];
125         -         for (int i = 0; i < n1; i++)
126         -             for (int j = 0; j < n2; j++)
127         -                 E[i*n2+j] = sample(r[i * n2 + j + offset]);
      106 +         Noise.sample(T_chi, r, offset, E);
128   107           return E;
129   108           }
130   109
```

...rg/bouncycastle/pqc/crypto/frodo/Noise.java

```
      ...   @@ -0,0 +1,30 @@
        1 + package org.bouncycastle.pqc.crypto.frodo;
        2 +
        3 + abstract class Noise
        4 + {
        5 +     static void sample(short[] cdf, short[] r, int rOff, short[] s)
        6 +     {
        7 +         // No need to compare with the last value.
        8 + //        assert cdf[cdf.length - 1] == 0x7FFF;
        9 +
       10 +         // Fills 's' with samples from the noise distribution 'cdf' using
          pseudo-random values 'r[rOff..]'
       11 +         for (int i = 0, n = s.length; i < n; ++i)
       12 +         {
```

```
13  +             int sample = 0;
14  +             int r_i = r[rOff + i] & 0xFFFF;
15  +             int prnd = r_i >>> 1;   // Drop the least significant bit
16  +             int sign = r_i & 1;     // Pick the least significant bit
17  +
18  +             for (int j = 0; j < cdf.length - 1; ++j)
19  +             {
20  +                 // Constant time comparison: 1 if cdf[j] < prnd, 0 otherwise.
21  +                 sample += (cdf[j] - prnd) >>> 31;
22  +             }
23  +
24  +             // Assuming that sign is either 0 or 1, flips sample iff sign = 1
25  +             sample = ((-sign) ^ sample) + sign;
26  +
27  +             s[i] = (short)sample;
28  +         }
29  +     }
30  + }
```

Comments 0

Please sign in to comment.