A Summary of Bleichenbacher's Padding Oracle Attack

In this document we summarize the implementation details of Bleichenbacher's PKCS1 v1.5 padding oracle attack. We assume that the reader is acquainted with the attack and only aim to summarize its implementation and provide some basic details; of course, for details about the attack, refer to the original article.

We denote the public modulus by N, the public exponent by e, the private exponent by d, and the given ciphertext to be decrypted by $c = m^e \bmod N$. We denote by O the Bleichenbacher oracle which determines whether an encrypted message is PKCS-conforming or not.

Step 1: The "blinding" phase

This part of the attack is used to find an initial ciphertext c_0 that conforms to the PKCS standard. If a conforming ciphertext c is given, this stage can be skipped, and the output of it can be set to $c_0 = c$ and $s_0 = 1$. If c is not necessarily PKCS-conforming, in this step we search for it thus:

Algorithm 1 The "blinding" phase

1: **procedure** BLINDING(c, N, e, O)
2: **repeat**
3: $s_0 \leftarrow$ RANDOM_INT () mod N
4: $c_0 \leftarrow s_0^e \cdot c \bmod N$
5: **until** $O(c_0)$
6: **return** s_0, c_0
7: **end procedure**

Step 2: Searching for PKCS-conforming messages

In this stage, the algorithm maintains a set of possible intervals (ranges) of the plaintext m, we call it M. The algorithm iteratively reduces the number of possibilities for m; in each iteration, the algorithm searches for a value s such that $O(c \cdot (s^e) \bmod N) = 1$ — notice that this is similar to the blinding phase; in the blinding phase, random values of s are generated, while here different methods of searching for s are used — and then uses this value to reduce the set of possible values of m. The method of searching for such a value s varies according to the iteration number — the first iteration behaves differently to the others — and to the number of intervals in M. Notice that this step relies on the values of s_0 and c_0 returned from the previous step. We denote by k the byte-length of N.

The algorithm relies on a few procedures which we also describe here:

FIND_MIN_CONFORMING (c_0, s_{min}) finds the minimal value s that is greater than s_{min} such that $O(c_0 \cdot (s^e) \bmod N) = 1$. In other words, in this case s is searched for sequentially beginning with an initial value s_{min}. This is the searching method used for the first iteration and for iterations where M contains more than one interval (with appropriate values for s_{min} as shown below).

SEARCH_SINGLE_INTERVAL searches for a value s as described above in a more complex method; it is used when M consists of more than a single interval.

NARROW_M (M, s) is run in each iteration of the algorithm after finding its value of s as described above, and uses this value to find a new set of intervals M' for the next iteration. Notice that in the code below,

when constructing a new set of intervals M' (which becomes the set of intervals M for the next iteration of the algorithm), we treat the union operator $\cup$ as though it merges existing intervals in M'. We denote by $M[0]$ the first interval in the interval list M.

Algorithm 2 Procedures for step 2

$B \leftarrow 2^{8(k-2)}$

procedure FIND_MIN_CONFORMING(c_0, s_{min}, N, e, O)
    $s \leftarrow s_{min}$
    while $O(c_0 \cdot (s^e) \bmod N) = 0$ **do**
        $s \leftarrow s + 1$
    end while
    return s
end procedure

procedure SEARCH_SINGLE_INTERVAL($s_{prev}, M, c_0, N, e, O$)
    $a, b \leftarrow M[0]$
    $r \leftarrow \left\lceil 2 \cdot \frac{b \cdot s_{prev} - 2B}{N} \right\rceil$
    while true **do**
        for s from $\left\{ \left\lceil \frac{2B+rN}{b} \right\rceil \right\}$ to $\left\lfloor \frac{3B+rN}{s} \right\rfloor$ **do**
            if $O(c_0 \cdot (s^e) \bmod N) = 1$ **then**
                return s
            end if
        end for
    end while
end procedure

procedure NARROW_M(M, s, N)
    $M' \leftarrow \{\}$
    for a, b in M **do**
        for r from $\left\lceil \frac{as-3B+1}{N} \right\rceil$ to $\left\lfloor \frac{bs-2B}{N} \right\rfloor$ **do**
            $M' \leftarrow M' \cup \left\{ \left[\max\left(a, \left\lceil \frac{2B+rN}{s} \right\rceil \right), \min\left(b, \left\lfloor \frac{3B-1+rn}{s} \right\rfloor \right) \right] \right\}$
        end for
    end for
end procedure

The algorithm for step 2 follows; in it we denote by $|M|$ the number of intervals in M (the size of the set of intervals M). We also by $M[0][0]$ the left boundary of the first interval in M and by $|M[0]|$ the length of the first interval in M, i.e. if $M[0]$ is $[a, b]$ for $a, b \in \mathbb{Z}_{\geq 0}$, then $M[0][0]$ is equal to a and $|M[0]|$ is equal to $b - a + 1$.

Algorithm 3 Step 2

procedure STEP2(c_0, s_0, N, e, O)
  $M \leftarrow \{[2B, 3B-1]\}$
  $i \leftarrow 1$
  repeat
    if $i = 1$ **then**
      $s \leftarrow$ FIND_MIN_CONFORMING $\left(c_0, \left\lceil \frac{N}{3B} \right\rceil, N, e, O\right)$
    else if $|M| > 1$ **then**
      $s \leftarrow$ FIND_MIN_CONFORMING $(c_0, s+1, N, e, O)$
    else
      $s \leftarrow$ SEARCH_SINGLE_INTERVAL (s, M, c_0, N, e, O)
    end if
    $M \leftarrow$ NARROW_M (M, s, N)
    $i \leftarrow i + 1$
  until $|M| = 1$ and $|M[0]| = 1$
  return $M[0][0] \cdot s^{-1} \bmod N$
end procedure