Algorithm 1 INVERT-BIT($[b]^i$)

if $i = n \wedge n$ is even **then**
  return $[b]^i$
else
  return $\neg[b]^i$
end if

Algorithm 2 R-CONT($[c], [d_i]$)

A $\leftarrow$ Array of size $l - r + 1$
for $f = r$ to l **do**
  B $\leftarrow$ Array of size r
  for $w = f - r + 1$ to f **do**
    $[z_w] = [c_w] - [d_{iw}]$
    $B[w - f + r] =$ INVERT-BIT($[z_w]$)
  end for
  $[v_f] = \wedge_{h=1}^{r} B[h]$
  $A[f - r + 1] =$ INVERT-BIT($[v_f]$)
end for
return INVERT-BIT($\wedge_{h=1}^{l-r+1} A[h]$)

Algorithm 3 TOLERIZE($[D]$)

On receiving connection c
  Share c in bitwise additive fashion
for $i = 1$ to q **do**
  $[u_i] = R - CONT([c], [d_i])$
  $u_i =$ RECONSTRUCT($[u_i]$)
  if u_i **then**
    remove ($[d_i]$)
  end if
end for

Algorithm 4 DETECT($[D], [M], [c], COUNT$)

```
for i = 1 to |D| do
    [u_i] = R − CONT([c], [d_i])
    u_i = RECONSTRUCT([u_i])
    COUNT[i] += u_i
    if COUNT[i] ≥ τ then
        [M].add([d_i])
        return 1
    end if
end for
for i = 1 to |M| do
    [u_i] = R − CONT([c], [d_i])
    u_i = RECONSTRUCT([u_i])
    if u_i then
        return 1
    end if
end for
return 0
```