Timing attack na RSA

Kód ze cvičení předmětu NI-KRY:

In[1]:=
```
(*Timing attack on RSA*)
(*J.F.Dhem,F.Koeune,P.-A.Leroux,P.Mestré,J.-J.Quisquater,
and J.-L.Willems,"A practical implementation of the timing attack"*)
(*This notebook contains a simplified simulation with a short RSA key*)
```

In[2]:=
```
SeedRandom[1234];
(* generate RSA parameters - short keys for faster analysis *)
p = RandomPrime[{2^11, 2^12}];(*12 + 12 = 24 bit modulus*)
q = RandomPrime[{2^11, 2^12}];
n = p * q
d = RandomInteger[{2^23 + 1, 2^24 - 1}];
(*simulate random private exponent - not quite RSA*)
d = BitSet[d, 0];
r = 2^24 (* Montgomery R *)
{g, {l, ri}} = ExtendedGCD[n, r]
k = -l (* k = -n^-1 mod r *)
r * ri == k * n + 1
```

Out[5]= 11 002 307

Out[8]= 16 777 216

Out[9]= {1, {-4 253 973, 2 789 707}}

Out[10]= 4 253 973

Out[11]= True

In[12]:=
```
(* Mongomery reduction *)
redc[t_] := Module[{m, tt},
  m = Mod[k * Mod[t, r], r];
  tt = (t + m * n) / r;
  If[tt >= n, tt - n, tt]
 ]
(* Instrumented Montgomery reduction - signal final subtraction *)
redc2[t_] := Module[{m, tt},
  m = Mod[k * Mod[t, r], r];
  tt = (t + m * n) / r;
  If[tt >= n, {tt - n, 1}, {tt, 0}]
  (* {xxx,1} = final subtraction occured; {xxx,0} otherwise *)
 ]
```

In[14]:=
```
(* Montgomery multiplication *)
monmult[a_, b_] := redc[a * b]
monmult2[a_, b_] := redc2[a * b]
```

In[16]:=
```
(* Square & Multiply exponentiation using Montgomery multiplication *)
sm[a_, d_] := Module[{aa, x, len, i}, (* compute a^d mod n *)
  aa = monmult[a, Mod[r^2, n]]; (* Convert input to Montgomery domain *)
  x = aa;
  len = BitLength[d];
  For[i = len - 2, i >= 0, i--,
   x = monmult[x, x]; (* square *)
   If[BitGet[d, i] == 1, x = monmult[x, aa]]; (* multiply *)
  ];
  monmult[x, 1]](* Convert output back to integer domain *)

(* Instrumented Square & Multiply - gives the number of final subtraction occured *)
sm2[a_, d_] := Module[{aa, x, len, i, t, cnt = 0},
  aa = monmult[a, Mod[r^2, n]];
  x = aa;
  len = BitLength[d];
  For[i = len - 2, i >= 0, i--,
   {x, t} = monmult2[x, x]; (* square, accumulate final subtraction *)
   cnt += t;
   If[BitGet[d, i] == 1,
    {x, t} = monmult2[x, aa]; (* multiply, accumulate final subtraction *)
    cnt += t;
   ];
  ];
  {monmult[x, 1], cnt}]
(*test*)
sm[7, 5] == PowerMod[7, 5, n]
```

Out[18]= True

In[19]:=
```
(* generate some random messages *)
zpravy = RandomInteger[{2, 2^16}, 10 000];
(* measure times for d_22=0 *)
casy0 = sm2[#, BitClear[d, 22]][[2]] & /@ zpravy;
(* measure times for d_22=1 *)
casy1 = sm2[#, BitSet[d, 22]][[2]] & /@ zpravy;
```

In[22]:=
```
(*************** Method 1 - attack on multiplication ***************)
```

In[23]:=
```
(* oracle: did the final subtraction occur in c^2*c? Assumes d22=1 *)
orak[c_] := Module[{cc, tmp, t},
   cc = monmult[c, Mod[r^2, n]];
   tmp = monmult[cc, cc];
   {tmp, t} = monmult2[tmp, cc];
   t];
```

In[24]:=
```
oo = orak[#] & /@ zpravy;(* Apply the oracle on all messages,
produces a vector {0,1,0,0,0,1,...} *)
```

In[25]:=
```
(* test for d22=0 *)
F10 = Pick[casy0, oo, 1];
F20 = Pick[casy0, oo, 0];
Length[F10]
Length[F20]
(Mean[F10] // N) - Mean[F20] // N
 (* should not differ significantly because the oracle assumes d22=
  1 but we have d22=0*)
```

Out[27]= 1658

Out[28]= 8342

Out[29]= 0.597571

In[30]:=
```
(* test for d22=1 *)
F11 = Pick[casy1, oo, 1];
F21 = Pick[casy1, oo, 0];
Length[F11]
Length[F21]
(Mean[F11] // N) - Mean[F21] // N
 (* should differ significantly because the oracle assumes d22=1 which is correct *)
```

Out[32]= 1658

Out[33]= 8342

Out[34]= 1.29744

Nová orákula pro útok na druhou mocninu:

In[35]:=
```
(* oracle: did the final subtraction occur in (c^2*c)^2? Assumes d22=1 *)
ora1[c_] := Module[{cc, tmp, t},
    cc = monmult[c, Mod[r^2, n]];
    tmp = monmult[cc, cc];
    tmp = monmult[tmp, cc];
    {tmp, t} = monmult2[tmp, tmp];
    t];
(* oracle: did the final subtraction occur in (c^2)^2? Assumes d22=0 *)
ora2[c_] := Module[{cc, tmp, t},
    cc = monmult[c, Mod[r^2, n]];
    tmp = monmult[cc, cc];
    {tmp, t} = monmult2[tmp, tmp];
    t];
oo1 = ora1[#] & /@ zpravy;
oo2 = ora2[#] & /@ zpravy;
```

Využití nově vytvořených orákul pro server 0 (tedy casy0). Nejprve roztřídíme pomocí prvního orákula proměnnou casy0 do dvou množin F1c0 a F2c0, poté zkontrolujeme jejich délky a vypočteme rozdíl průměrů délek těchto dvou množin. F1c0 je množina kde docházelo k závěrečným odečtením a F2c0, kde nedocházelo. To samé provedeme pro druhé orákulum. Za lepší výsledek orákula považujeme markantnější rozdíl průměrů délek daných dvou množin. U orákula s lepším výsledkem předpokládáme, že má pravdu :

In[39]:=
```
F1c0 = Pick[casy0, oo1, 1];
F2c0 = Pick[casy0, oo1, 0];
F3c0 = Pick[casy0, oo2, 1];
F4c0 = Pick[casy0, oo2, 0];
"Délka množiny F1c0: " <> ToString[Length[F1c0]]
"Délka množiny F2c0: " <> ToString[Length[F2c0]]
result10 = (Mean[F1c0] // N) - Mean[F2c0] // N;
"Rozdíl průměrů délek pro první orákulum: " <> ToString[result10]
"Délka množiny F3c0: " <> ToString[Length[F3c0]]
"Délka množiny F4c0: " <> ToString[Length[F4c0]]
result20 = (Mean[F3c0] // N) - Mean[F4c0] // N;
"Rozdíl průměrů délek pro druhé orákulum: " <> ToString[result20]
If[result10 > result20, "První orákulum má pravděpodobně pravdu.",
 "Druhé orákulum má pravděpodobně pravdu."]
```

Out[43]= Délka množiny F1c0: 2137

Out[44]= Délka množiny F2c0: 7863

Out[46]= Rozdíl průměrů délek pro první orákulum: −0.077051

Out[47]= Délka množiny F3c0: 2122

Out[48]= Délka množiny F4c0: 7878

Out[50]= Rozdíl průměrů délek pro druhé orákulum: 0.617168

Out[51]= Druhé orákulum má pravděpodobně pravdu.

Stejný postup provedeme znovu akorát pro server 1 (tedy casy1):

In[52]:=
```
F1c1 = Pick[casy1, oo1, 1];
F2c1 = Pick[casy1, oo1, 0];
F3c1 = Pick[casy1, oo2, 1];
F4c1 = Pick[casy1, oo2, 0];
"Délka množiny F1c1: " <> ToString[Length[F1c1]]
"Délka množiny F2c1: " <> ToString[Length[F2c1]]
result11 = (Mean[F1c1] // N) - Mean[F2c1] // N;
"Rozdíl průměrů délek pro první orákulum: " <> ToString[result11]
"Délka množiny F3c1: " <> ToString[Length[F3c1]]
"Délka množiny F4c1: " <> ToString[Length[F4c1]]
result21 = (Mean[F3c1] // N) - Mean[F4c1] // N;
"Rozdíl průměrů délek pro druhé orákulum: " <> ToString[result21]
If[result11 > result21, "První orákulum má pravděpodobně pravdu.",
 "Druhé orákulum má pravděpodobně pravdu."]
```

Out[56]= Délka množiny F1c1: 2137

Out[57]= Délka množiny F2c1: 7863

Out[59]= Rozdíl průměrů délek pro první orákulum: 0.67947

Out[60]= Délka množiny F3c1: 2122

Out[61]= Délka množiny F4c1: 7878

Out[63]= Rozdíl průměrů délek pro druhé orákulum: -0.0811356

Out[64]= První orákulum má pravděpodobně pravdu.

V těchto dvou měřeních jsme řešili pouze hodnotu 22. bitu. Můžeme si všimnout, že pro server 0 vyšel výsledek lépe u druhého orákula, které předpokládá hodnotu 22. bitu rovnou hodnotě 0, což je správný výsledek, jelikož server 0 má hodnotu 22. bitu skutečně rovnou 0. Server 1 má opačnou hodnotu 22.bitu, tudíž by lepší výsledek mělo mít první orákulum, jelikož to předpokládá hodnotu 22. bitu rovnou 1. Vidíme, že i pro pro server 1 vyšel výsledek správně.

In[65]:=