CPA-security for Padded RSA

Tingxiang Zou

Recap: Padded RSA

Padded RSA

Let l be a function with $l(n) \leq 2n - 2$:

•Gen: on input 1^n, output public key (N, e) and secret key (N, d).

•Enc: on message $m \in \{0, 1\}^{l(n)}$, $\text{Enc}_{(N,e)} = [(r\|m)^e \mod N]$, where $r \leftarrow \{0, 1\}^{\|N\|-l(n)-1}$.

•Dec: on ciphertext $c \in \mathbb{Z}_N^*$, $\text{Dec}_{(N,d)} =$ the $l(n)$ low-order bits of $[c^d \mod N]$

- For $l(n) = 2n - O(\log n)$, not CPA-secure;
- For $l(n) = O(\log n)$, CPA-secure under the RSA assumption.
- **RSA assumption**: there is no efficient algorithm, which given N, e and a random y, can find x with non-negligible probability, such that $y = [x^e \mod N]$.

CPA-security for Padded RSA with $l(n) = 1$

We say that **the RSA least significant-bit is unpredictable** if there is no efficient algorithm, which given N, e and a randomly chosen y, can find the least significant bit of x with non-negligible probability over $\frac{1}{2}$, such that $y = [x^e \mod N]$.

Theorem: The RSA least significant-bit is unpredictable under the RSA assumption.

Corollary: Padded RSA with $l(n) = 1$ is CPA-secure under the RSA assumption.

This result can be generalised to the j-least significant bits, for $j = O(\log n)$.

A Reduction Proof

Lemma

If there is a PPT $\mathcal{A}$, that given N, e and a random y can find the least significant bit of x with non-negligible probability over $\frac{1}{2}$, such that $y = [x^e \mod N]$, then there is a PPT $\mathcal{A}'$, which can find x with non-negligible probability.

Two important techniques:

- Improve the performance of $\mathcal{A}$ on the RSA lsb by executing independent measurements and taking the majority vote.

- Invert the RSA encoding function by a gcd algorithm (Brent-Kung gcd procedure) in the presence of a reliable adversary for RSA lsb.

Independent measurements and the majority vote

• Suppose you want to answer a yes-or-no question Q by asking some consultant $\mathcal{O}$.

• Suppose each time you ask, the probability you get the right answer is $\frac{2}{3}$.

• Ask it independently for 3 times, and give the majority answer. Now the probability that your answer is wrong is:
$Pr[\text{3 wrong answers}] + Pr[\text{2 wrong answers}]$
$= \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} + (\frac{1}{3} \cdot \frac{1}{3} \cdot \frac{2}{3} + \frac{1}{3} \cdot \frac{2}{3} \cdot \frac{1}{3} + \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{1}{3}) = \frac{7}{27} < \frac{1}{3}$.

• If you ask it 5 times independently, then you can do even better.

• For PPT $\mathcal{A}$ which can guess the RSA lsb with $Pr[Succ(n)] = \frac{1}{2} + \frac{1}{n^c}$, polynomial many independent runs will give $Pr[Succ(n)] \approx 1 - \frac{1}{n}$.

Brent-Kung gcd Procedure

Compute gcd(10,15):

(10,15)$\longrightarrow$ Only one of them is even, 2 cannot be a common divisor.

It won't hurt to replace 10 with 5.

(5,15)$\longrightarrow$ They are all odd.

They have the same common divisors as $(\frac{15+5}{2}, \frac{15-5}{2})$.

(10,5)$\longrightarrow$ Only 10 is even, replace it with $\frac{10}{2}$.

(5,5)$\longrightarrow$ 5 must be the greatest common divisor.

• We only need to know the parity of r, s (which is the lsb), and be able to do the linear combination.

Inverting RSA encryption function

- Convention: Let $[x]_N$ denote $[x \mod N]$.

- If $\mathcal{A}$ can guess the RSA lsb with probability almost 1, then given N, e, y with $y = [x^e]_N$, he knows almost for sure the parity of any $[ax]_N$ and $[bx]_N$. He also can calculate $[(2^{-1}(ax \pm bx))^e]_N$, hence knows the parity of $[2^{-1}(a \pm b)x]_N$.

- If $[ax]_N$ and $[bx]_N$ are coprime, then applying the Brent-Kung gcd procedure for $([ax]_N, [bx]_N)$, $\mathcal{A}$ can efficiently get a c, such that $[cx]_N = 1$. Then $x = [c^{-1} \mod N]$, which is efficiently computable.

- Theorem (Dirichlet 1849): The probability that two random integers in $[1, N]$ are coprime converges to $\frac{6}{\pi^2} \approx 0.608$ as N tends to $+\infty$.
  Hence, take two randomly chosen $a, b \in \mathbb{Z}_N$, $[ax]_N$ and $[bx]_N$ are coprime with high probability.

Conclusion

- Main result: in RSA, determining the least-significant bit of the plaintext is as hard as inverting the RSA encryption function (i.e., knowing the whole plaintext.)
- We see two useful techniques:
  1. Independent measurements + the majority vote;
  2. Brent-Kung gcd procedure.

Thank You!