RSA meets DPA: Recovering RSA Secret Keys from Noisy Analog Data

Noboru Kunihiro and Junya Honda

The University of Tokyo, Japan

September 25th, 2014

The full version is available from
http://eprint.iacr.org/2014/513.

RSA Scheme & PKCS #1 standard

(Textbook) RSA

- $N(= pq), ed \equiv 1 \pmod{(p-1)(q-1)}$
- Public Key (N, e), Secret Key d
- Encryption $C = M^e \bmod N$
- Decryption $M = C^d \bmod N$

Speeding-up via Chinese Remainder Theorem

- Auxiliary Secret Key: $d_p = d \bmod p - 1, d_q = d \bmod q - 1.$
- Compute $M_p = C^{d_p} \bmod p$ and $M_q = C^{d_q} \bmod q$.
- Find M s. t. $M = M_p \bmod p$ and $M = M_q \bmod q$ via CRT.
- Secret Key tuples $(p, q, d, d_p, d_q, q^{-1} \bmod p)$

Secret keys have a redundancy.

Side Channel Attacks against RSA

Extract related values to secret key (p, q, d, d_p, d_q) by physical observation.

$$\boxed{\text{Correct Secret Key}} \xrightarrow[\text{Leakage}]{\text{Observation}} \boxed{\text{Measured Value}}$$

$$\begin{aligned} p &= 110011011 \cdots 1 \\ q &= 100100110 \cdots 1 \\ d &= 1 \cdots 00111 \cdots 1 \\ d_p &= 10111110 \cdots 10 \\ d_q &= 11110110 \cdots 100 \end{aligned} \xrightarrow[\text{Leakage}]{\text{Observation}} \begin{aligned} \tilde{p} &= 100111011 \cdots 1 \\ \tilde{q} &= 100000111 \cdots 1 \\ \tilde{d} &= 1 \cdots 00011 \cdots 1 \\ \tilde{d}_p &= 10111110 \cdots 10 \\ \tilde{d}_q &= 10010110 \cdots 100 \end{aligned}$$

Denote by m the number of involved key in attacks.

Previous Leakage Model for RSA

Discrete Leakage: Each bit is

- erased with prob. δ. (Heninger-Shacham (CRYPTO2009))
- bit-flipped with prob. ϵ. (Henecka-May-Meurer (CRYPTO2010))
- bit-flipped with asymmetric prob. (Paterson et al. (AC2012))
- erased with prob. δ and bit-Flipped with prob. ϵ. (K-Shinohara-Izu (PKC2013)).

Is This Leakage Model Appropriate?

Analog data is more natural as observed data through the actual physical attacks.

Our Goal

Propose efficient algorithms that recover RSA secret keys from noisy analog data.

Our Leakage Model

The observed value follows some fixed probability distribution depending on the corresponding correct secret key.
ex.) additive white noise: $b + \epsilon, \epsilon \sim \mathcal{N}$ for $b \in \{0, 1\}$.

More formally,

- Let $f_0(y), f_1(y)$ be probability density functions with average $1, -1$. The observed value y follows
  - $f_0(y)$ if the bit is 0 and
  - $f_1(y)$ if the bit is 1.
- In our model, we obtain a single sample.

$$
\begin{array}{l} p = 1100110\cdots \\ q = 1001001\cdots \\ d = 1010010\cdots \\ d_p = 1110001\cdots \\ d_q = 1010101\cdots \end{array}
\Longrightarrow
\begin{array}{l} \tilde{p} = -1.21, -0.85, +0.34, -0.45, -0.47, -1.05, -0.05, \cdots \\ \tilde{q} = -0.50, +0.12, -0.34, -1.67, -0.56, +0.23, -1.03, \cdots \\ \tilde{d} = -0.92, +0.93, -0.74, +0.45, +0.97, -1.35, +0.05, \cdots \\ \tilde{d}_p = +0.01, -0.12, -1.56, +1.67, +2.01, +0.93, -1.11, \cdots \\ \tilde{d}_q = -0.50, +0.12, -0.34, +1.11, -0.56, +1.00, -1.08, \cdots \end{array}
$$

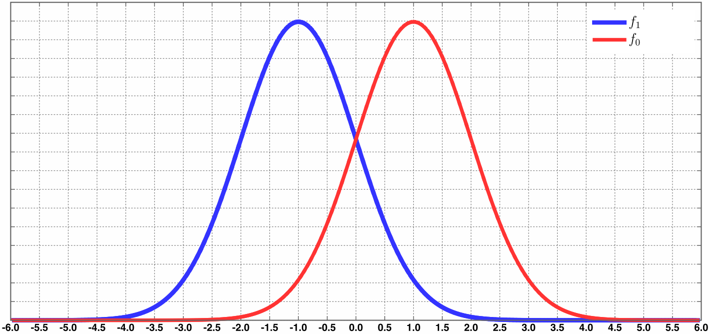
Leakage Model (cont.)

Gaussian Distribution: $\mathcal{N}(\mu, \sigma^2)$

Denote the Gaussian distribution with average μ and variance σ^2.

Symmetric Leakage

If $f_0(y) = f_1(-y)$, we say that f_0 and f_1 are symmetric.



Naive Approach

Quantization Approach

- Set an adequate threshold and quantize the observed values into $0, 1$ and "$?$".
- Apply the KSI algorithm to the quantized (discrete) values to recover the secret key.

Results

- Consider a symmetric Gaussian case: $f_x(y) = \mathcal{N}((-1)^x, \sigma^2)$.
- If σ satisfies

$$0 \leq \sigma < 1.533,$$

we can recover the secret key in polynomial time.

Our Contributions

1. We propose two algorithms for recovering secret key from analog observed data:
   1. Maximum Likelihood Ratio Based (ML-based) Algorithm:
      More effective than DPA-like Algorithm.
   2. DPA-like Algorithm:
      Works without knowledge of leakage distribution.
2. We derive the condition of f_1 and f_0 for recovering secret key.
   - Consider the Gaussian noise case: $f_x = \mathcal{N}((-1)^x, \sigma^2)$. If it satisfies
     $$0 \leq \sigma < 1.767,$$
     we can recover the secret key in polynomial time.

Common Framework

We use Tree-Based approach (proposed by Heninger and Shacham).

$$\mathbf{slice}(i) := (p[i], q[i], d[i+\tau(k)], d_p[i+\tau(k_p)], d_q[i+\tau(k_q)])$$

Assume we obtained a partial secret key up to $\mathbf{slice}(i-1)$.

Constraints that each bits satisfies in secret key

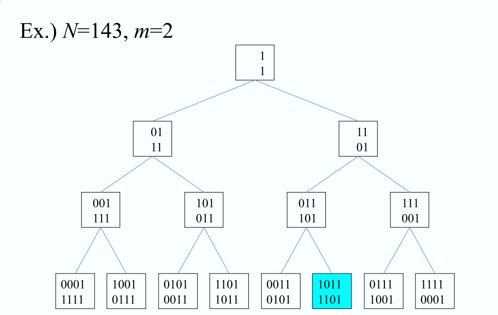
$$\begin{aligned} p[i] + q[i] &= c_1 \bmod 2, \\ d[i+\tau(k)] + p[i] + q[i] &= c_2 \bmod 2, \\ d_p[i+\tau(k_p)] + p[i] &= c_3 \bmod 2, \\ d_q[i+\tau(k_q)] + q[i] &= c_4 \bmod 2. \end{aligned}$$

Each bits in $\mathbf{slice}(i)$ have four constraints for five variables.

$\Rightarrow$ There are two candidates.

Tree-based Approach

- Represent $\mathbf{slice}(i)$ by binary tree.
- Once the public key is fixed, the whole binary tree is uniquely determined. The number of leafs in the tree is $2^{n/2}$.
- One of leafs corresponds to the correct secret key.
- Determine with an adequate rule whether each node is discarded or remained by using observed sequence and candidate sequence.



Generalized Algorithm

Score Function

- Introduce a score function.
  - Syntax: $\mathbf{Score}(\boldsymbol{x}, \boldsymbol{y})$.
  - Candidate sequence: $\boldsymbol{x} = (x_1, \ldots, x_{mT}) \in \{0, 1\}^{mT}$.
  - Observed sequence: $\boldsymbol{y} = (y_1, \ldots, y_{mT}) \in \mathbb{R}^{mT}$.
- $\mathbf{Score}$ functions are designed that $\mathbf{Score}(\boldsymbol{x}, \boldsymbol{y})$ becomes large if $\boldsymbol{x}$ is a correct candidate.

Core of Generalized Algorithm

Expansion Phase Generate 2^t nodes from remained L nodes. Then, we have $L2^t$ nodes: $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{L2^t}$.

Pruning Phase Remain top L nodes with $\mathbf{Score}(\boldsymbol{x}_i, \boldsymbol{y})$ among $L2^t$ nodes.

Proposed Algorithm 1 (ML-based Algorithm)

Our Score Function

Use log likelihood ratio

$$R_T(\boldsymbol{x}, \boldsymbol{y}) := \frac{1}{mT} \sum_{i=1}^{mT} \log \frac{f_{x_i}(y_i)}{g(y_i)}$$

as a score function, where $g(y) = (f_1(y) + f_0(y))/2$.

Differential Entropy

$h(f)$ is a differential entropy of f:

$$h(f) = -\int f(y) \log f(y) \mathrm{d}y.$$

Main Theorem

Let $R(x;y) = \frac{f_x(y)}{g(y)}$ for $x \in \{0,1\}$. Define the mutual information by $I(X;Y) = \mathrm{E}[R(X;Y)]$. Assume that $I(X;Y)$ and m satisfy

$$I(X;Y) > \frac{1}{m}.$$

For any parameters (t, L), the failure probability of ML-based Algorithm is less than

$$\frac{n}{2t}\rho_1 L^{-\rho_2}$$

for some $\rho_1, \rho_2 > 0$ which only depend on m and f_x.
$\Rightarrow$ the failure probability converges to zero as $L \to \infty$ for any $t > 0$.

Lemma

The $I(X;Y)$ is explicitly given by

$$I(X;Y) = h\left(\frac{f_0 + f_1}{2}\right) - \frac{h(f_0) + h(f_1)}{2}.$$

Gaussian Leakage Case

Success Condition for Gaussian Leakage

Assume that $f_x = \mathcal{N}((-1)^x, \sigma^2)$. The success condition is explicitly given by

$$\sigma < 1.767 \text{ for } m = 5.$$

$\Rightarrow$ Superior to the quantization method.

Equivalent Form of Score Function for Gaussian

The order of score itself is important. Absolute value of score is not. We ignore common terms to all candidates $\boldsymbol{x}_i$.

$$R_T(\boldsymbol{x}, \boldsymbol{y}) := \frac{1}{mT} \sum_{i=1}^{mT} (-1)^{x_i} y_i.$$

How does ML-based Algorithm work?: Intuitive

The log-likelihood ratio

$$R_T(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{mT} \sum_{i=1}^{mT} \log \frac{f_{x_i}(y_i)}{g(y_i)}$$

- For incorrect candidate $\boldsymbol{x}$, $\mathrm{E}(R_T(\boldsymbol{x}, \boldsymbol{y})) = 0$.
- For correct candidate $\boldsymbol{x}$,
  $\mathrm{E}(R_T(\boldsymbol{x}, \boldsymbol{y})) = h\left(\frac{f_0 + f_1}{2}\right) - \frac{h(f_0) + h(f_1)}{2} (> 0).$

The central limit theorem guarantees that $R_T(\boldsymbol{x}, \boldsymbol{y})$ is near to its expectation if T is large enough.

$\Rightarrow$ R_T for the correct candidate is the highest with high prob.

Proof Technique

Cramér's Theorem

Let $Z \in \mathbb{R}$ be a random variable and $\Lambda(\lambda) = \ln \mathrm{E}[\exp(\lambda Z)]$ be its cumulant generating function. For independently and identically distributed copies $Z_1, Z_2, \cdots, Z_n$ of Z and any $u \in \mathbb{R}$,

$$\Pr\left[\frac{1}{n}\sum_{i=1}^{n} Z_i \geq u\right] \leq \exp\left(-n \sup_{\lambda \geq 0}\{\lambda u - \Lambda(\lambda)\}\right).$$

We set

$$Z_i = \log \frac{f_{b_i}(X_i)}{g(X_i)}.$$

Proof Sketch of Main Theorem (1/2)

Let $u \in \left(\frac{1}{m}, h\left(\frac{f_0+f_1}{2}\right) - \frac{h(f_0)+h(f_1)}{2}\right)$.

For correct candidate $\boldsymbol{x}$

Since $\mathrm{E}[Z_i(\boldsymbol{x})] = h\left(\frac{f_0+f_1}{2}\right) - \frac{h(f_0)+h(f_1)}{2}$, the probability that $R_T(\boldsymbol{x}, \boldsymbol{y}) < u$ is less than

$$\exp(-mT\Lambda^*(u)),$$

where $\Lambda^*(u) = \sup_{\lambda \leq 0}\{\lambda u - \Lambda(\lambda)\}$.

For incorrect candidate $\boldsymbol{x}$

Since $\mathrm{E}[\exp(Z_i)(\boldsymbol{x})] = 1$, the probability that $R_T(\boldsymbol{x}, \boldsymbol{y}) > u$ is less than $\exp(-mTu)$.

Proof Sketch of Main Theorem (2/2)

The condition that the correct candidate is pruned in each pruning phase is given as follows:

For some $u \in \left(\frac{1}{m}, h\left(\frac{f_0+f_1}{2}\right) - \frac{h(f_0)+h(f_1)}{2}\right)$,

- The score $R_T(\boldsymbol{x}, \boldsymbol{y})$ for a correct $\boldsymbol{x}$ is less than u.
- The number of incorrect $\boldsymbol{x}$'s whose scores are bigger than u is larger than or equal to $L-1$.

Combining the above discussions, we have Main Theorem.

Proposed Algorithm 2 (DPA-like Algorithm)

Dawback of ML-based Algorithm

We MUST know the exact noise distribution.

$\Rightarrow$ It is not practical.

New Score Function

$$\mathbf{DPA}(\boldsymbol{x}, \boldsymbol{y}) := \frac{1}{mT} \sum_{i=1}^{mT} (-1)^{x_i} y_i.$$

Consideration: The new $\mathbf{DPA}$ function

- depends on only $\boldsymbol{x}$ and $\boldsymbol{y}$.
- can be calculated without knowledge of the specific form of noise distributions.
- is the same as the Gaussian noise case.

Our New DPA function: Intuitive

Transformation:

$$\mathbf{DPA}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{mT}\left(\sum_{\{i|x_i=0\}} y_i - \sum_{\{i|x_i=1\}} y_i\right)$$

Very similar to difference-of-means distinguisher in [KJJ@CRYPTO99].

Intuition:

- For correct candidate $\boldsymbol{x}$, $\mathrm{E}(\mathbf{DPA}(\boldsymbol{x}, \boldsymbol{y})) = 1$.
- For incorrect candidate $\boldsymbol{x}$, $\mathrm{E}(\mathbf{DPA}(\boldsymbol{x}, \boldsymbol{y})) = 0$.
- When T goes to ∞, $\mathbf{DPA}(\boldsymbol{x}, \boldsymbol{y})$ converges to 1 and 0, respectively.
- We can separate the correct candidate from incorrect one.

Success Condition for DPA-like Algorithm

Main Theorem 2

Suppose that $f_x(y)$ is a probability density function with average $(-1)^x$ and variance σ_x^2. The success condition is given by

$$h\left(\frac{f_0+f_1}{2}\right)-\frac{1}{2}\log(\pi \mathrm{e}(\sigma_0^2+\sigma_1^2))>\frac{1}{m}.$$

Symmetric Noise Case:

If f_0 and f_1 are symmetric, it will be

$$h\left(\frac{f_0(x)+f_0(-x)}{2}\right)-h(\mathcal{N}(1,\sigma^2))>\frac{1}{m}.$$

ML-based versus DPA-like method

Success Condition: Symmetric Leakage

$$h(g) - h(f) > 1/m \text{ for ML-based algorithm}$$
$$h(g) - h(\mathcal{N}(1, \sigma^2)) > 1/m \text{ for DPA-like algorithm}$$

Information Loss of DPA-like Score from ML-based score

can be expressed as

$$h(\mathcal{N}(1, \sigma^2)) - h(f)(\geq 0).$$

It increases as the true noise distribution deviates from Gaussian.
The equality is attained if and only if f is the Gaussian.

Implementation Results

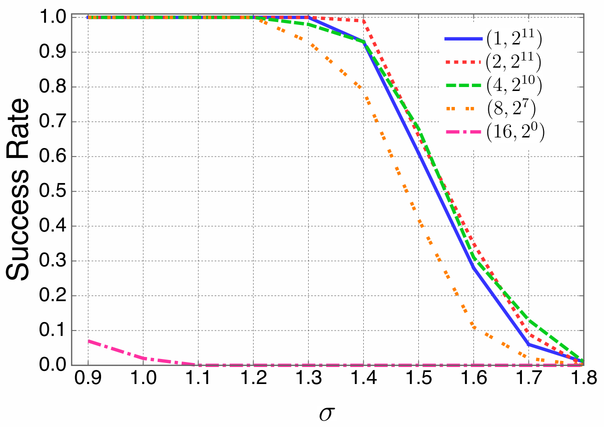


Figure: Experiments for $m = 5$ and $n = 1024$ and various (t, L).

Implementation Results II (For Fixed t)

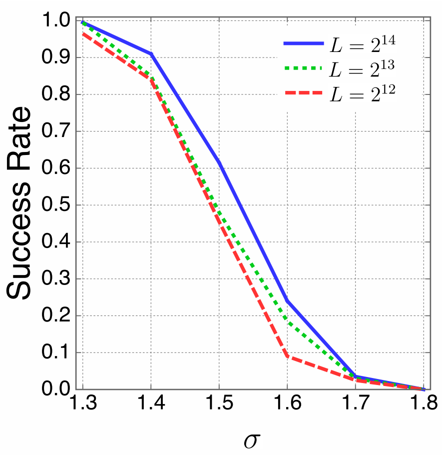


Figure: $m = 5$, $n = 1024$ and $t = 1$

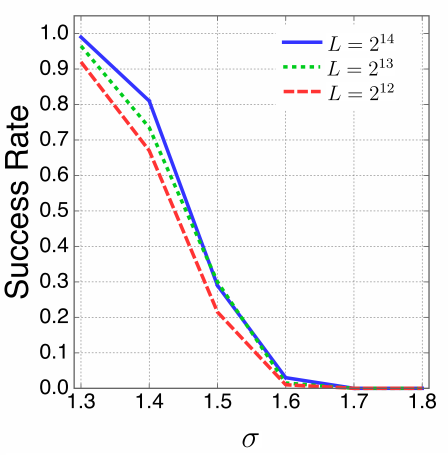


Figure: $m = 5, n = 2048$ and $t = 1$

Conclusions

- Evaluated the security of RSA when the secret key bits are leaked with some noise.
- Proposed two algorithms: ML-based and DPA-like algorithms.
- The ML-based algorithm can recover the secret key if

$$I(X;Y) = h\left(\frac{f_0 + f_1}{2}\right) - \frac{h(f_1) + h(f_0)}{2} > \frac{1}{m}.$$

    - Assume that $f_x = \mathcal{N}((-1)^x, \sigma^2)$. It can recover the secret key in polynomial time if $\sigma < 1.767$.
- For the DPA-like algorithm,
    - Need not know leakage distributions.
    - The success condition is slightly worse than ML-based algo.
    - BUT, it is completely equivalent to that of ML-based algorithm when the noise distribution is Gaussian.