Partial Key Exposure: Generalized Framework to Attack RSA

Santanu Sarkar

Indian Statistical Institute, 203 B T Road, Kolkata 700 108, India
sarkar.santanu.bir@gmail.com

Abstract. In the domain of modern public key cryptography, RSA is the most popular system in use. Efficient factorization of the RSA modulus N, constituted as a product of two primes p, q of 'large' bitsize, is a challenging problem in RSA cryptanalysis. The solution to this factorization is aided if the attacker gains partial knowledge about the decryption exponent of RSA. This line of attack is called the Partial Key Exposure attack, and there exists an extensive literature in this direction.

In this paper, we study partial key exposure attacks on RSA where the number of unexposed blocks in the decryption exponent is more than one. The existing works have considered only one unexposed block and thus our work provides a generalization of the existing attacks. We propose lattice based approaches to factorize the RSA modulus $N = pq$ (for large primes p, q) when the number of unexposed blocks is $n \geq 1$. We also analyze the ISO/IEC 9796-2 standard signature scheme (based on CRT-RSA) with partially known messages.

Keywords: Factorization, ISO/IEC 9796-2 Signature, Lattice, Partial Key Exposure, RSA.

1 Introduction

RSA cryptosystem, publicly proposed in 1978 and named after its inventors Ron Rivest, Adi Shamir and Len Adleman, is the most popular Public Key Cryptosystem till date. One can describe the RSA scheme in a nutshell as follows [15].

Cryptosystem 1 (RSA). *Let us define $N = pq$ where p and q are primes. By definition of the Euler totient function, $\phi(N) = (p-1)(q-1)$.*

- KeyGen: *Choose integer e co-prime to $\phi(N)$ and find $d = e^{-1} \bmod \phi(N)$.*
- KeyDist: *Publish public key $\langle N, e \rangle$ and keep private key $\langle N, d \rangle$ secret.*
- Encrypt: *For message $M \in \mathbb{Z}_N$, ciphertext $C = M^e \bmod N$.*
- Decrypt: *For ciphertext C, message $M = C^d \bmod N$.*

For encryption and decryption in RSA cryptosystem, one needs modular exponentiation. To reduce the cost of encryption, one can take a small e. In such a case, d becomes quite large, i.e., of the order of N, and the decryption process

D.J. Bernstein and S. Chatterjee (Eds.): INDOCRYPT 2011, LNCS 7107, pp. 76–92, 2011.
© Springer-Verlag Berlin Heidelberg 2011

will be much less efficient than the encryption. Consider that one likes to make the decryption process faster. Then the secret decryption exponent d has to be made small. However, Wiener [17] showed that when $d < \frac{1}{3}N^{0.25}$, one can factor N efficiently, thus making RSA insecure. This result has been improved by Boneh and Durfee [3] to provide the upper bound $d < N^{0.292}$.

Attacks Based on Partial Knowledge of *d*. Kocher [13] proposed a new attack on RSA to obtain the private exponent d. He showed that an attacker can get a few bits of d by timing characteristic of an RSA implementing device. In [2], it has been studied how many bits of d need to be known to mount an attack on RSA. The constraint in the work of [2] was the upper bound on e which is $\sqrt{N}$. The study attracted interest and the idea of [2] has been improved in [1] where the bound of e was increased upto $N^{0.725}$. Then the work of [7] improved the result for full size public exponent e. These attacks require knowledge of contiguous blocks of bits of the RSA secret keys. The results of Ernst et al. [7] are as follows:

Theorem 1. *Let $N = pq$ be an RSA modules with p, q are of same bit size. Let e be of full bit size and $d \leq N^{\delta}$. Then given $(\delta - \gamma)\log N$ Most Significant Bits of d, one can factor N in polynomial time if*

- $\gamma < \frac{5}{6} - \frac{1}{3}\sqrt{1 + 6\delta}$ *or*
- $\gamma < \frac{\beta}{3} + \frac{1}{2} - \frac{1}{3}\sqrt{4\beta^2 + 6\beta}$ *where* $\beta = \max\{\gamma, \delta - \frac{1}{2}\}$

Attacks on a CRT-RSA signature scheme. CRT-RSA is used to devise one of the most popular digital signature schemes. To sign a message m, one first needs to calculate

$$s_p = m^{d_p} \bmod p \quad \text{and} \quad s_q = m^{d_q} \bmod q.$$

Now the signature s for modulus $N = pq$ can be computed using CRT with s_p and s_q. Boneh et al. [4] showed that CRT-RSA implementations are vulnerable due to fault attacks. Suppose the attacker is able to induce a fault while s_q is being calculated, such that $s_q \neq m^{d_q} \bmod q$. Then using the faulty signature s, the attacker can factor N as $\gcd(s^e - m, N) = p$. The attack of Boneh et al. also works for any deterministic RSA encoding or any probabilistic signature scheme where some randomness is used to generate the signature and is also sent as a part of the signature. However, if some part of the message is unknown, the attack of Boneh et al. does not work any more. Recently Coron et al. [5,6] showed how to use the same idea in such a situation, and their attack was illustrated against ISO/IEC 9796-2 standards [11]. In ISO/IEC 9796-2, the encoded message is of the form

$$\mu(m) = 6A_{16} \mid\mid m[1] \mid\mid H(m) \mid\mid BC_{16},$$

where $m = m[1] \mid\mid m[2]$ is split into two parts. Note that $H(m)$, the hash value of m is unknown to the attacker. Coron et al. proved that when unknown part of $m[1]$ is small, one can factor N efficiently, thus making the scheme vulnerable.

1.1 Our Contribution

Motivated by the work of [8], we consider the situation where a few contiguous blocks of the RSA secret exponent d are unknown, and the encryption exponent e is of full bit size. In such a situation, the reconstruction idea of [9] will not work as e is of full bit size. We propose a lattice based approach to handle the situation. In [7], authors analyzed the case when number n of unknown contiguous blocks is one. We prove that when the number n of unknown blocks increases, one needs more bits to be known for polynomial time factorization of N.

We prove the apparently surprising result that even for an arbitrary n number of blocks, only $(\delta + \frac{1}{2}\sqrt{1+4\delta} - 1)\log N$ bits of $d(= N^{\delta})$ are sufficient to reconstruct the complete d in the case when $d < N^{0.75}$. Note that as long as $\delta < 0.275$, $\delta + \frac{1}{2}\sqrt{1+4\delta} - 1$ is less than zero. Although the runtime of our reconstruction algorithm is polynomial in $\log N$, it is exponential in the number of unknown blocks n. So, in case $n = O(\text{poly}(\log\log \text{N}))$, our algorithm is a poly(log N) time algorithm.

Next, we consider the case when a large block MSBs of d is exposed. Using these known MSBs of d, we propose another lattice based approach. We prove that if attacker knows d_0 such that $|d - d_0| < N^{\delta - \frac{1}{2}}$, the knowledge of

$$\sqrt{2\left(\delta - \frac{1}{2}\right)^2 + \left(\delta - \frac{1}{2}\right)} \cdot \log N$$

many bits of d is sufficient for the factorization of N in time polynomial in $\log N$ but exponential in n.

Finally, we turn our attention to the CRT-RSA signature scheme, and consider the case where two faults occur for the two different primes individually. That is attacker inject a fault modulo p for one signature, and a fault modulo q for another signature. Let the sizes of the unknown blocks for each of the messages for two faulty signatures be $\delta_1 \log N, \cdots, \delta_n \log N$ and the size of hash value be $\gamma \log N$. We prove when

$$\delta_1 + \cdots + \delta_n + \gamma < \frac{n+2}{2(2n+3)},$$

one can factor N in time polynomial in $\log N$ but exponential in n. For the case $n = 1$, i.e., if the size of the unknown block of each of the messages is $\delta \log N$ and the size of the hash value is $\gamma \log N$, the approach of Coron et al. [5] works if $\delta + \gamma < 0.167$. Putting $n = 1$ in our bound, we obtain the upper bound as 0.30. The idea of using Coppersmith's method to improve on the bound 0.167, was already suggested in [5].

Organization. In summary, our work in this paper is stated and organized as follows.

- In Section 2, we study partial key exposure attacks when more than one block of d is unknown. In Section 3, we consider the case when a large block of MSBs of d is known.

– In Section 4, we present our attack on ISO/IEC 9796-2 standard signature scheme with partially known messages.

Our approaches are based on the technique of [12], which itself follows from the idea of [10]. For all the results that we obtain in this paper related to lattice based techniques, we need the following assumption. For brevity, we do not refer to this assumption in each of the results, though it should be considered implied in each of them.

Assumption 1. Let lattice reduction be executed using the idea of LLL [14] algorithm leading to polynomials in u variables. Consider that LLL outputs the polynomials $f_1, f_2, \ldots, f_i$, $i \geq u$, that have a common root. Then one can efficiently compute this root from $f_1, f_2, \ldots, f_i$ using techniques like calculation of resultants of these polynomials or finding a Gröbner basis.

We have implemented all the programs in SAGE 3.1.1 over Linux Ubuntu 8.04 on a laptop with Dual CORE Intel(R) Pentium(R) D CPU 1.83 GHz, 2 GB RAM and 2 MB Cache. In all cases except in Section 4 we take a 1024-bit RSA modulus N. In Section 4, we consider both 1024-bit and 2048-bit RSA moduli.

2 General Attacks Based on Partial Knowledge of d

In this section we consider the scenario when certain blocks of the decryption exponent d are known (exposed) and naturally the other parts are unknown (unexposed). Figure 1 provides a pictorial view of the attack model.

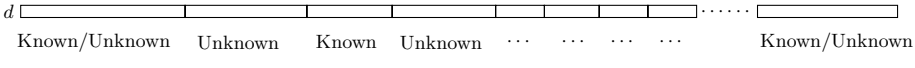

Fig. 1. Exposed and unexposed blocks of d

Before proceeding further, let us state the following technical result [16] that we will be using later.

Proposition 1. *For any fixed positive integer $r \geq 1$, and a large integer m,*

$$\sum_{t=1}^{m} t^r = \frac{m^{r+1}}{r+1} + o(m^{r+1}).$$

Theorem 2. *Let e be $O(N)$ and $d \leq N^{\delta}$. Suppose the bits of d are exposed except n many blocks, each of size $\gamma_i \log N$ bits for $1 \leq i \leq n$. Then one can factor N in polynomial in $\log N$ but exponential in n time if*

$$\sum_{i=1}^{n} \gamma_i < 1 - \frac{1}{2(n+2)} - \frac{n+1}{2(n+2)}\sqrt{4\delta + 1 + \frac{4\delta}{n+1}}.$$

Proof. Since d is unknown for n many blocks, one can write $d = a_0 + a_1 y_1 + \ldots + a_n y_n$, where $y_1, y_2, \ldots, y_n$ are unknown. Now from $ed = 1 + k(N + 1 - p - q)$, so we have $ea_0 + ea_1 y_1 + \ldots + ea_n y_n - 1 - k(N + 1 - p - q) = 0$. Hence, we are interested to find the root of the polynomial

$$f(x_1, \ldots, x_{n+1}, x_{n+2}) = ea_0 + ea_1 x_1 + \ldots + ea_n x_n - 1 + N x_{n+1} + x_{n+1} x_{n+2}.$$

Clearly, $f\,(y_1, \ldots, y_n, -k, 1 - p - q) = 0$. Let, $X_i = N^{\gamma_i}$ for $1 \leq i \leq n, X_{n+1} = N^{\delta}, X_{n+2} = N^{0.5}$. Clearly, $X_1, \ldots, X_{n+2}$ is the upper bound of absolute value of $y_1, \ldots, y_n, -k, 1 - p - q$, neglecting small constants. Using the extended strategy of [12, Section 2.2], we define

$$S = \bigcup_{0 \leq j \leq t} \{x_1^{i_1} x_2^{i_2} \ldots x_{n+2}^{i_{n+2}+j} : x_1^{i_1} x_2^{i_2} \ldots x_{n+2}^{i_{n+2}} \text{ is a monomial of } f^m\}$$

$$M = \{\text{monomials of } x_1^{i_1} x_2^{i_2} \ldots x_{n+2}^{i_{n+2}} f : x_1^{i_1} x_2^{i_2} \ldots x_{n+2}^{i_{n+2}} \in S\}$$

where m, t are non-negative integers. To use Coppersmith's method, it suffices to find at least $n + 1$ more polynomials $f_1, f_2, \ldots f_{n+1}$ that share the same root $(y_1, \ldots, y_n, -k, 1 - p - q)$ over the integers.

Considering a fixed n, we know from [12] that these polynomials can be found by LLL [14] algorithm in poly(log N) time if

$$X_1^{s_1} X_2^{s_2} \cdots X_{n+2}^{s_{n+2}} < W^s$$

for $s_j = \sum_{x_1^{i_1} \cdots x_{n+2}^{i_{n+2}} \in M \setminus S} i_j$ with $j = 1, \ldots, n + 2$, $s = |S|$, and

$$W = ||f(x_1 X_1, \ldots x_{n+2} X_{n+2})||_\infty \geq N X_{n+1} = N^{1+\delta}.$$

The calculation's of $s, s_1, \ldots, s_{n+2}$ are quite tedious and those are presented Appendix A. From the structure of the polynomial f, it is clear that $s_1, \ldots, s_n$ are equal. We get,

$$\begin{aligned}
s &\approx \frac{m^{n+2}}{(n+2)!} + \frac{t m^{n+1}}{(n+1)!}, \\
s_1 = \cdots = s_n &\approx \frac{m^{n+2}}{(n+2)!} + \frac{t m^{n+1}}{(n+1)!}, \\
s_{n+1} &\approx \frac{2m^{n+2}}{(n+2)!} + t\frac{m^{n+1}}{(n+1)!}, \\
s_{n+2} &\approx \frac{m^{n+2}}{(n+2)!} + \frac{t m^{n+1}}{(n+1)!} + \frac{t^2 m^n}{2n!}
\end{aligned}$$

Consider $t = \tau m$, where $\tau \geq 0$ is a real number. Now putting values of $X_1, \ldots, X_{n+2}$ and $s_1, \ldots, s_{n+2}, s$, and the lower bound of W in

$$X_1^{s_1} X_2^{s_2} \ldots X_{n+2}^{s_{n+2}} < W^s,$$

we get

$$\left(\frac{1}{4}n^2+\frac{3}{4}n+\frac{1}{2}\right)\tau^2+\left(n\sum_{i=1}^{n}\gamma_i+2\sum_{i=1}^{n}\gamma_i-\frac{1}{2}n-1\right)\tau+\left(\sum_{i=1}^{n}\gamma_i+\delta-\frac{1}{2}\right)<0. \tag{1}$$

The optimal value of τ is $\frac{1-2\sum_{i=1}^{n}\gamma_i}{n+1}$. Putting this optimal value of τ in Equation (1), we get

$$\sum_{i=1}^{n}\gamma_i<\frac{(2n^2+3n)-\sqrt{4n^4\delta+n^4+12n^3\delta+2n^3+8n^2\delta+n^2}}{2(n^2+2n)}.$$

From which we have the required condition.

Using the strategy of [12, Section 2.2], one can construct a lattice L from S, M. The bit size of the entries of L is poly(log N), and

$$\dim(L)=|M|=\frac{(m+1)^{n+2}}{(n+2)!}+\frac{t(m+1)^{n+1}}{(n+1)!}+o((m+1)^{n+2}).$$

The running time of our algorithm is dominated by the LLL algorithm run on L, which takes time polynomial in the dimension of the lattice and in the bitsize of the entries. Since the lattice dimension in our case is exponential in n, so the total running time for this method is polynomial in $\log N$ but exponential in n. □

Note that when $n=1$ i.e, number of unknown block is one, then the situation is analyzed in [7].

Now putting $n=1$ in our Theorem 2, we get $\gamma_1<\frac{5}{6}-\frac{1}{3}\sqrt{1+6\delta}$, same bound as in the first result of the Theorem 1. However, since we assume bits of d are known from any position, we can not get any partial information of k. This is the reason we can not obtain other result of Theorem 1. Also note that total number of unknown bits of d is $\sum_{i=1}^{n}\gamma_i\log N$. Now from Theorem 2

$$\sum_{i=1}^{n}\gamma_i<1-\frac{1}{2(n+2)}-\frac{n+1}{2(n+2)}\sqrt{4\delta+1+\frac{4\delta}{n+1}}.$$

Also when n increases, then upper bound of $\sum_{i=1}^{n}\gamma_i$ decreases, i.e, one needs much number of bits $\left(\delta-\sum_{i=1}^{n}\gamma_i\right)\log N$ to be known. In Table 1, we present few numerical values of the upper bound of unknown bits.

Now we have,

$$\lim_{n\to\infty}1-\frac{1}{2(n+2)}-\frac{n+1}{2(n+2)}\sqrt{4\delta+1+\frac{4\delta}{n+1}}=1-\frac{\sqrt{1+4\delta}}{2}.$$

Table 1. Numerical upper bound of unknown bits of d for different n

δ	$n=1$	$n=2$	$n=3$	$n=4$
0.30	0.275	0.270	0.267	0.266
0.35	0.246	0.240	0.237	0.234
0.40	0.219	0.211	0.207	0.205
0.45	0.192	0.183	0.179	0.176
0.50	0.167	0.157	0.152	0.148
0.55	0.142	0.131	0.125	0.122
0.60	0.118	0.106	0.100	0.096
0.65	0.095	0.082	0.075	0.071
0.70	0.073	0.059	0.051	0.047
0.75	0.051	0.036	0.028	0.023
0.80	0.030	0.014	0.005	0.000

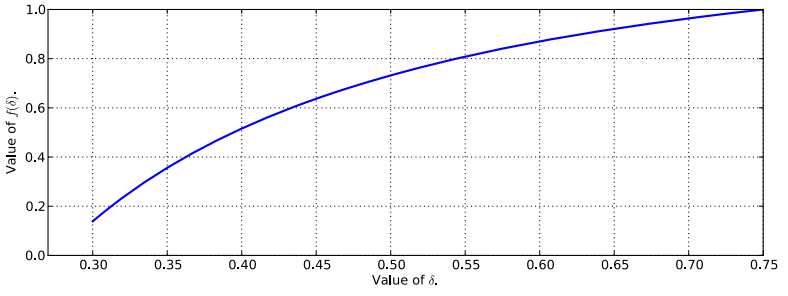

Fig. 2. Partial Key Exposure Attack for d. Plot of $f(\delta) = 1 + \frac{\sqrt{1+4\delta}}{2\delta} - \frac{1}{\delta}$ vs. values of δ.

So, when $\sqrt{1+4\delta} < 2$, i.e., when $\delta < 0.75$, knowledge of $\left(\delta + \frac{\sqrt{1+4\delta}}{2} - 1\right)\log N$ many bits of d is sufficient to factor N irrespective of their position in time polynomial in $\log N$ and exponential in number of unknown blocks n. So we can formally state the formally the following result.

Corollary 1. *Let e be full bit size and $d \leq N^{\delta}$ with $\delta < 0.75$. Then knowledge of*

$$\left(\delta + \frac{\sqrt{1+4\delta}}{2} - 1\right)\log N$$

many bits of d is sufficient to factor N in time polynomial in $\log N$ and exponential in number of unknown blocks of d.

Figure 2, represents the proportion of bits of d need to be known irrespective of their position such that one can factor N in time polynomial in $\log N$ and exponential in number of unknown blocks of d.

In our experiments, Assumption 1 always holds and we have successfully collected the desired root. We present the experimental results in Table 2. LD

Table 2. Experimental results for $n = 2$ and $n = 3$

n	δ	$\sum_{i=1}^{n} \gamma_i$	(m, t)	LD	Time (Sec.)
2	0.30	0.200	(2,1)	55	99.69
2	0.35	0.145	(2,1)	55	107.94
2	0.40	0.095	(2,1)	55	114.12
2	0.45	0.060	(2,1)	55	122.82
2	0.50	0.045	(2,1)	55	114.23
2	0.55	0.010	(2,1)	55	99.68
3	0.30	0.195	(2,1)	91	911.31
3	0.35	0.140	(2,1)	91	901.11
3	0.40	0.090	(2,1)	91	1002.15
3	0.45	0.040	(2,1)	91	914.22

denotes the lattice dimension. First we take $n = 2, m = 2, t = 1$. In this case lattice dimension will be 55. When $\delta = 0.30$ i.e., bit size of d is 308, factoring N requires knowledge of $(0.30 - 0.20) \times 1024$ i.e., 103 many bits of d. When $\delta = 0.40$ and $n = 2$, we need the knowledge of $(0.40 - 0.095) \times 1024 = 313$ many bits of d. For larger values of δ or n, we need more number of bits to be known for efficient factorization.

3 Attacks Using the Partial Knowledge of k

Now consider the situation when few MSBs of d are known. Figure 3 provides a pictorial view of this case. In this situation one can also find an approximation of k.

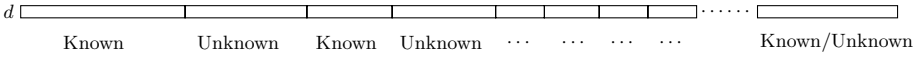

Fig. 3. Exposed and unexposed blocks of d

Theorem 3. *Let e be full bit size and $d \leq N^{\delta}$. Suppose one knows an approximation d_0 of d such that $|d - d_0| < N^{\lambda}$. Also assume that attacker knows bits of d except n many blocks and size of each such block is $\gamma_i \log N$ for $1 \leq \gamma_i \leq n$. Then he/she can factor N in time polynomial in $\log N$ and exponential in n if*

$$\sum_{i=1}^{n} \gamma_i < \frac{(1 + \beta n + \frac{n}{2}) - \sqrt{(2\,n^2 + 2\,n)\,\beta^2 + (n^2 + 3\,n + 2)\,\beta}}{n + 2}$$

where $\beta = \max\{\lambda, \delta - \frac{1}{2}\}$.

Proof. Since attacker knows an approximation d_0 of d, he/she can also find an approximation $k_0 = \lfloor \frac{ed_0-1}{N} \rfloor$ of k. In [1], it is proved that $|k - k_0| < N^\beta$ where $\beta = \max\{\lambda, \delta - \frac{1}{2}\}$ when e is of full bit size. Let, $k_1 = k - k_0$. Since, d is unknown for n many blocks, so one can write $d = a_0 + a_1y_1 + \cdots + a_ny_n$, where $y_1, y_2, \ldots, y_n$ are unknown. Now from $ed = 1 + k(N + 1 - p - q)$, we have

$$ea_0 + ea_1y_1 + \cdots + ea_ny_n - 1 - (k_0 + k_1)(N + 1 - p - q) = 0.$$

Hence, we are interested to find the root of the polynomial

$$\begin{aligned} f(x_1, \ldots, x_{n+2}) = (ea_0 - k_0N - 1) + ea_1x_1 + \cdots + ea_nx_n + Nx_{n+1} \\ - k_0x_{n+2} + x_{n+1}x_{n+2}. \end{aligned}$$

Clearly, $f(y_1, \ldots, y_n, -k_1, 1 - p - q) = 0$.

Let $X_i = N^{\gamma_i}$, for $1 \le i \le n$. Further, suppose $X_{n+1} = N^\beta$ and $X_{n+2} = N^{0.5}$. Clearly, $X_1, \ldots, X_{n+2}$ are the upper bounds of absolute values of $y_1, \ldots, y_n$, $-k_1, 1 - p - q$, neglecting small constants. Using the extended strategy of [12, Section 2.2], we define

$$S = \bigcup_{0 \le j \le t} \{x_1^{i_1}x_2^{i_2} \ldots x_{n+1}^{i_{n+1}+j}x_{n+2}^{i_{n+2}} : x_1^{i_1}x_2^{i_2} \ldots x_{n+2}^{i_{n+2}} \text{ is a monomial of } f^m\}$$

$$M = \{\text{monomials of } x_1^{i_1}x_2^{i_2} \ldots x_{n+2}^{i_{n+2}} f : x_1^{i_1}x_2^{i_2} \ldots x_{n+2}^{i_{n+2}} \in S\}$$

where t is a non-negative integer.

Apart from f, we need to find at least $n+1$ more polynomials $f_1, f_2, \ldots f_{n+1}$ that share the same root $(y_1, \ldots, y_n, -k_1, 1 - p - q)$ over the integers.

Considering a fixed n, we know that these polynomials can be found by LLL [14] algorithm if

$$X_1^{s_1}X_2^{s_2} \cdots X_{n+2}^{s_{n+2}} < W^s$$

for $s_j = \sum_{x_1^{i_1} \cdots x_{n+2}^{i_{n+2}} \in M \setminus S} i_j$ with $j = 1, \ldots, n+2$, $s = |S|$, and

$$W = ||f(x_1X_1, \ldots x_{n+2}X_{n+2})||_\infty \ge NX_{n+1} = N^{1+\beta}.$$

From the structure of the polynomial f, it is clear that $s_1, \ldots, s_n$ are equal.

Using similar kind computation as in Theorem 2, We get

$$\begin{aligned} s &\approx \frac{2m^{n+2}}{(n+2)!} + \frac{tm^{n+1}}{(n+1)!}, \\ s_1 = \cdots = s_n &\approx \frac{2m^{n+2}}{(n+2)!} + \frac{tm^{n+1}}{(n+1)!}, \\ s_{n+1} &\approx \frac{3m^{n+2}}{(n+2)!} + t\frac{2m^{n+1}}{(n+1)!} + t^2\frac{m^n}{2n!}, \\ s_{n+2} &\approx \frac{3m^{n+2}}{(n+2)!} + \frac{tm^{n+1}}{(n+1)!} \end{aligned}$$

Table 3. Numerical upper bound of unknown bits of d for different n using the partial knowledge of k

δ	β	γ_1	$\sum_{i=1}^{2}\gamma_i$	$\sum_{i=1}^{3}\gamma_i$
0.30	0.25	0.1424	0.1408	0.1400
0.40	0.25	0.1424	0.1408	0.1400
0.60	0.25	0.1424	0.1408	0.1400
0.75	0.25	0.1424	0.1408	0.1400
0.80	0.30	0.1101	0.1092	0.1087
0.85	0.35	0.802	0.0797	0.0794
0.90	0.40	0.0521	0.0519	0.0518
0.95	0.45	0.0255	0.0254	0.0254

Now consider $t = \tau m$, where $\tau \geq 0$ is a real number. Putting the values of X_1, $X_2, \ldots, X_{n+2}$, $s_1, \ldots, s_{n+2}, s$, and the lower bound of W in the condition

$$X_1^{s_1} X_2^{s_2} \ldots X_{n+2}^{s_{n+2}} < W^s,$$

we get

$$\left(\frac{n^2\beta}{2} + \frac{3n\beta}{2} + \beta\right)\tau^2 + \left(n\sum_{i=1}^{n}\gamma_i + 2\sum_{i=1}^{n}\gamma_i + n\beta + 2\beta - \frac{n}{2} - 1\right)\tau + 2\sum_{i=1}^{n}\gamma_i + \beta - \frac{1}{2} < 0.$$

The optimal value of τ is $\frac{\frac{1}{2} - \sum_{i=1}^{n}\gamma_i - \beta}{n\beta + \beta}$.

Putting this optimal value of τ in Equation (2),

$$\sum_{i=1}^{n}\gamma_i < \frac{(1 + \beta n + \frac{n}{2}) - \sqrt{(2\,n^2 + 2\,n)\,\beta^2 + (n^2 + 3\,n + 2)\,\beta}}{n+2}.$$

□

Putting $n = 1$ in Theorem 3, we get $\gamma_1 < \frac{1}{2} + \frac{\beta}{3} - \frac{1}{3}\sqrt{4\beta^2 + 6\beta}$. This bound is the same as the second result of Theorem 1. In Table 3, we present a few numerical values of the upper bound of unknown bits. In all cases we take $\lambda = 0.25$.

Now,

$$\lim_{n\to\infty} \frac{(1 + \beta n + \frac{n}{2}) - \sqrt{(2\,n^2 + 2\,n)\,\beta^2 + (n^2 + 3\,n + 2)\,\beta}}{n+2} = \beta + \frac{1}{2} - \sqrt{2\beta^2 + \beta}. \tag{2}$$

Hence, knowledge of $\left(\delta - \beta - \frac{1}{2} + \sqrt{2\beta^2 + \beta}\right)\log N$ many bits of d is sufficient to factor N in time polynomial in $\log N$ and exponential in number of unknown blocks of d. When $\lambda < \delta - \frac{1}{2}$, value of β will be $\beta = \delta - \frac{1}{2}$.

Table 4. Experimental results for $n = 2$

δ	$\sum_{i=1}^{2} \gamma_i$	(m, t)	LD	Time (Sec.)
0.38	0.116	(2,0)	50	224.28
0.38	0.120	(2,1)	70	831.55
0.40	0.093	(2,0)	50	276.32
0.40	0.098	(2,1)	70	727.19
0.45	0.058	(2,0)	50	225.38
0.45	0.065	(2,1)	70	956.81
0.50	0.040	(2,0)	50	211.85
0.50	0.044	(2,1)	70	1005.91
0.55	0.012	(2,0)	50	187.43
0.55	0.016	(2,1)	70	984.42
0.56	0.004	(2,0)	50	194.51
0.56	0.008	(2,1)	70	1155.82

So from Equation (2), we can say that in this case $\sum_{i=1}^{n} \gamma_i$ should less than $\delta - \sqrt{2(\delta - \frac{1}{2})^2 + (\delta - \frac{1}{2})}$ as $n \to \infty$.

Thus, knowing $\left(\sqrt{2(\delta - \frac{1}{2})^2 + (\delta - \frac{1}{2})}\right) \log N$ many bits of d is sufficient to factor N in time polynomial in $\log N$ and exponential in n when $\lambda < \delta - \frac{1}{2}$.

In Table 4, we present some experimental results for different sizes of d. In all cases, we assume $\lambda = 0.25$ i.e., $(\delta - 0.25) \times 1024$ many MSBs of d to be known. For larger values of δ, we need more number of bits to be known for efficient factorization.

4 Attack on ISO/IEC 9796-2

Recall that in ISO/IEC 9796-2 [11], the encoded message is of the form

$$\mu(m) = 6A_{16} \mid\mid m[1] \mid\mid H(m) \mid\mid BC_{16},$$

where $m = m[1] \mid\mid m[2]$ is split into two parts. Let the message $m = m[1] \mid\mid m[2]$ be of the form $m[1] = \alpha_1 \mid\mid r_1 \mid\mid \cdots \mid\mid \alpha_n \mid\mid r_n \mid\mid \alpha_{n+1}$ and $m[2]$=data, where $r_1, r_2, \cdots, r_n$ are unknown and $\alpha_1, \alpha_2, \cdots, \alpha_{n+1}$ are known to the attacker, while the attacker may or may not know $m[2]$, and does not know the hash value $H(m)$. Coron et al. [5] first studied the case when faults are injected for the same prime. Next, they considered the case when two faulty signatures occur for two different primes. In this section, we consider two faults for two different primes. We first assume that after injecting a fault, the attacker obtains a faulty signature s such that

$$s^e = \mu(m) \mod p \text{ and } s^e \neq \mu(m) \mod q.$$

Hence we have $s^e = a_0 + a_1 r_1 + \cdots + a_n r_n + a_{n+1} H(m) \bmod p$. So in this case the attacker needs to find the root $(r_1, \cdots, r_n, H(m))$ of a polynomial of the form

$$b_0 + b_1 x_1 + \cdots + b_n x_n + b_{n+1} x_{n+1}$$

in $\mathbb{Z}_p$.

Now suppose that another faulty signature, which is incorrect modulo p but correct modulo q, and n different blocks in the first part of the message are unknown. So here we have

$$s_1^e = c_0 + c_1 r_1' + \cdots + c_n r_n' + c_{n+1} H(m') \bmod q.$$

Thus one needs to find the root $(r_1', \cdots, r_n', H(m'))$ of a polynomial of the form

$$d_0 + d_1 x_1 + \cdots + d_n x_n + d_{n+1} x_{n+1}$$

in $\mathbb{Z}_q$. Hence when two faults occur, one in modulo p and the other in modulo q, we have

$$(b_0 + b_1 r_1 + \cdots + b_n r_n + b_{n+1} H(m)) \times (d_0 + d_1 r_1' + \cdots + d_n r_n' + d_{n+1} H(m')) = 0 \bmod N.$$

So in this case the attacker needs to calculate the root of

$$f_N(z_1, \ldots, z_{n+1}, z_{n+2}, \ldots, z_{2n+2}) = (b_0 + b_1 z_1 + \cdots + b_n z_n + b_{n+1} z_{n+1}) \times (d_0 + d_1 z_{n+2} \cdots + d_{2n+2} z_{2n+2})$$

in $\mathbb{Z}_N$.

Let $X_i = X_{n+1+i} = N^{\delta_i}$ for $1 \le i \le n$ be the upper bounds of the absolute values of $r_1, r_1', r_2, r_2' \cdots, r_n, r_n'$ respectively. Also let $X_{n+1} = X_{2n+2} = N^{\gamma}$ be the upper bound of $H(m)$ and $H(m')$.

Using the strategy [12, Section 2.1], we define

$$M_k = \{z_1^{i_1} z_2^{i_2} \cdots z_{2n+2}^{i_{2n+2}} \mid z_1^{i_1} z_2^{i_2} \cdots z_{2n+2}^{i_{2n+2}} \text{ is a monomial of } f_N^m \ \& \frac{z_1^{i_1} \cdots z_{2n+2}^{i_{2n+2}}}{l^k} \text{ is a monomial of } f_N^{m-k}\},$$

where m is a non-negative integer and $l = z_1 z_{n+2}$ for $0 \le k \le m$. It follows that

$$z_1^{i_1} z_2^{i_2} \cdots z_{2n+2}^{i_{2n+2}} \in M_k \Leftrightarrow \begin{cases} 0 \le i_1 + \cdots + i_{n+1} \le m, \text{ and } k \le i_1 \le m \\ 0 \le i_{n+2} + \cdots + i_{2n+2} \le m \text{ and } k \le i_{n+2} \le m \end{cases}$$

For fixed n and using the idea of [12, Section 2.1], the root of f_N can be found in time polynomial in $\log N$ but exponential in n if

$$X_1^{s_1} \cdots X_{2n+2}^{s_{2n+2}} < N^s$$

Table 5. Experimental results when two faults occur with p and q

$\log N$	n	$\delta \log N$	$\gamma \log N$	m	LD	Time (Sec)
1024	1	74	160	2	36	21.71
2048	1	278	160	2	36	98.18
2048	1	180	256	2	36	95.05

for $s_k = \sum_{z_1^{i_1} \cdots z_{2n+2}^{i_{2n+2}} \in M_0} i_k$ with $k = 1, \cdots, 2n+2$, $s = \sum_{t=1}^{m} |M_t|$. One can check easily that

$$|M_t| = \frac{(m-t)^{2n+2}}{((n+1)!)^2} + o((m-t)^{2n+2}).$$

Hence $s \approx \sum_{t=1}^{m} \frac{(m-t)^{2n+2}}{(((n+1)!)^2} = \frac{m^{2n+3}}{((n+1)!)^2(2n+3))}$. Also it can be easily checked that

$$s_1 = \cdots = s_{2n+2} \approx \frac{m^{2n+3}}{(n+1)!(n+2)!}.$$

Putting the values of $X_1, X_2, \ldots, X_{2n+2}$, $s_1, \ldots, s_{2n+2}, s$ in the condition

$$X_1^{s_1} X_2^{s_2} \cdots X_{2n+2}^{s_{2n+2}} < N^s,$$

and neglecting the terms of $o(m^{2n+3})$ we get the condition $\sum_{i=1}^{n} \delta_i + \gamma < \frac{n+2}{2(2n+3)}$.

So, when $n \to \infty$, one gets the upper bound 0.25. In the presence of a single fault in this situation the upper bound was 0.153 [5, Section 2.2]. Now, when $n = 1$, we get the upper bound 0.30. Using a lattice of dimension 9, Coron et al. [5, Section 2.3] obtained the upper bound 0.167. Hence we achieve a better upper bound than the work of [5]. We use Coppersmith's idea to obtain better bound than [5].

In Table 5, we present some experimental results. We first consider the case when the hash function is SHA-1. Hence $\gamma \log N = 160$. Coron et al. [5, Table 2] presented the value of $\delta_1 \log N$ to be 13 and 158 for 1024-bit and 2048-bit RSA respectively for one faulty signature. Also using the approach of Coron et al. [5, Section 2.3], for faults of two different factors, we obtain an upper bound of δ_1 as $0.167 - 0.156 = 0.011$ for 1024-bit-RSA and $0.167 - 0.078 = 0.089$ for 2048-bit-RSA. Hence upper bounds of $\delta_1 \log N$ will be 12 and 182 for 1024-bit and 2048-bit RSA respectively. In our experiments, $\delta_1 \log N$ is 71 and 278 for 1024-bit and 2048-bit RSA respectively. We also consider the case when SHA 256 is used for 2048-bit-RSA. In this case N can be factored given two faulty signatures of two different factors containing 180 random bits in less than 2 minutes.

5 Conclusion

So far, all the existing partial key exposure attacks on RSA assume a single contiguous block of unknown bits of the secret exponent when encryption

exponent e is large. However, in practice, it is more likely that the attacker obtains certain bits of the secret exponent in fragmented blocks. In such a case, none of the existing methods work towards any significant cryptanalysis of the ciphers. We address this issue in our current work and generalize the above attacks, when the number of unexposed blocks can be more than one. We also study an ISO/IEC 9796-2 standard signature scheme with two faulty signatures, one modulo p and another modulo q. More than one faulty signature with same modulo was studied by Coron et al. [5]. So, it would be interesting to study the factorization of N with more than 2 signatures, some faulty modulo p and others faulty modulo q.

Acknowledgment. The authors would like to thank Prof. Subhamoy Maitra and the anonymous Indocrypt reviewers for their detailed comments on the technical issues of this paper. The authors are also grateful to Dr. Goutam Kumar Paul and Mr. Sourav Sen Gupta for their editorial contribution towards the presentation of this paper.

References

1. Blömer, J., May, A.: New Partial Key Exposure Attacks on RSA. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 27–43. Springer, Heidelberg (2003)
2. Boneh, D., Durfee, G., Frankel, Y.: An Attack on RSA Given a Small Fraction of the Private Key Bits. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 25–34. Springer, Heidelberg (1998)
3. Boneh, D., Durfee, G.: Cryptanalysis of RSA with Private Key d Less Than $N^{0.292}$. IEEE Transactions on Information Theory 46(4), 1339–1349 (2000)
4. Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of checking cryptographic protocols for faults. Journal of Cryptology 14(2), 101–119 (2001)
5. Coron, J.-S., Joux, A., Kizhvatov, I., Naccache, D., Paillier, P.: Fault Attacks on RSA Signatures with Partially Unknown Messages. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 444–456. Springer, Heidelberg (2009)
6. Coron, J.-S., Naccache, D., Tibouchi, M.: Fault Attacks Against EMV Signatures. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 208–220. Springer, Heidelberg (2010)
7. Ernst, M., Jochemsz, E., May, A., de Weger, B.: Partial Key Exposure Attacks on RSA up to Full Size Exponents. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 371–386. Springer, Heidelberg (2005)
8. Herrmann, M., May, A.: Solving Linear Equations Modulo Divisors: On Factoring Given Any Bits. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 406–424. Springer, Heidelberg (2008)
9. Heninger, N., Shacham, H.: Reconstructing RSA Private Keys from Random Key Bits. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 1–17. Springer, Heidelberg (2009)
10. Howgrave-Graham, N.: Finding small roots of univariate modular equations revisited. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997)
11. ISO/IEC 9796-2, Information technology - Security techniques - Digital signature scheme giving message recovery, Part 2: Mechanisms using a hash-function (1997)

12. Jochemsz, E., May, A.: A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 267–282. Springer, Heidelberg (2006)
13. Kocher, P.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
14. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261(4), 513–534 (1982)
15. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. Communications of ACM 21(2), 158–164 (1978)
16. Sarkar, S., Maitra, S.: Cryptanalysis of RSA with more than one Decryption Exponent. Information Processing Letters 110(8-9), 336–340 (2010)
17. Wiener, M.: Cryptanalysis of Short RSA Secret Exponents. IEEE Transactions on Information Theory 36(3), 553–558 (1990)

Appendix A: Detailed Calculations Related to Theorem 2

Here we present the detailed calculations for s, s_1, s_2, s_{n+2}.

Calculation of s

One may note that s is the number of solutions of $0 \leq i_1 + \cdots + i_{n+1} \leq m$, $0 \leq i_{n+2} \leq i_{n+1} + t$. Thus,

$$\begin{aligned} s &= \sum_{r=0}^{m} \binom{r+n-1}{r} \left(\frac{(m-r)^2}{2} + t(m-r) \right) \\ &\approx \sum_{r=0}^{m} \frac{r^{n-1}}{(n-1)!} \left(\frac{(m-r)^2}{2} + t(m-r) \right) \\ &\quad \text{(neglecting the lower order terms)} \\ &\approx \frac{m^{n+2}}{(n+2)!} + \frac{tm^{n+1}}{(n+1)!} \end{aligned}$$

(using Proposition 1 and neglecting the lower order terms).

Calculation of s_1

$$\begin{aligned} s_1 = &\sum_{i_1=0}^{m+1} \sum_{r=0}^{m-i_1+1} \sum_{i_{n+1}=0}^{m+1-i_1-r} \sum_{i_{n+2}=0}^{i_{n+1}+t} \binom{r+n-2}{r} i_1 \\ &- \sum_{i_1=0}^{m} \sum_{r=0}^{m-i_1} \sum_{i_{n+1}=0}^{m-i_1-r} \sum_{i_{n+2}=0}^{i_{n+1}+t} \binom{r+n-2}{r} i_1. \end{aligned}$$

Now,

$$\sum_{i_1=0}^{m}\sum_{r=0}^{m-i_1}\sum_{i_{n+1}=0}^{m-i_1-r}\sum_{i_{n+2}=0}^{i_{n+1}+t}\binom{r+n-2}{r}i_1$$
$$\approx \sum_{i_1=0}^{m}\sum_{r=0}^{m-i_1}\sum_{i_{n+1}=0}^{m-i_1-r}\frac{r^{n-2}}{(n-2)!}(i_{n+1}+t)i_1$$
(neglecting the lower order terms)
$$\approx \sum_{i_1=0}^{m}\sum_{r=0}^{m-i_1}\sum_{i_{n+1}=0}^{m-i_1-r}\frac{r^{n-2}}{(n-2)!}i_1\left(\frac{(m-i_1-r)^2}{2}+t(m-i_1-r)\right)$$

(using Proposition 1 and neglecting the lower order terms).

Let $r_1 = m - i_1$. Then

$\sum_{i_1=0}^{m}\sum_{r=0}^{m-i_1}\sum_{i_{n+1}=0}^{m-i_1-r}\sum_{i_{n+2}=0}^{i_{n+1}+t}\binom{r+n-2}{r}i_1$

$$\approx \sum_{i_1=0}^{m}\sum_{r=0}^{r_1}\frac{r^{n-2}}{(n-2)!}i_1\left(\frac{(r_1-r)^2}{2}+t(r_1-r)\right)$$
$$\approx \tfrac{1}{(n+1)!}\sum_{i_1=0}^{m} i_1(m-i_1)^{n+1}+\frac{t}{n!}\sum_{i_1=0}^{m} i_1(m-i_1)^n$$
$$\approx \tfrac{m^{n+3}}{(n+3)!}+\tfrac{t}{(n+2)!}m^{n+2}$$

$$\text{So } s_1 \approx \frac{(m+1)^{n+3}}{(n+3)!}+\frac{t}{(n+2)!}(m+1)^{n+2} - \frac{m^{n+3}}{(n+3)!}+\frac{t}{(n+2)!}m^{n+2}$$
$$\approx \frac{m^{n+2}}{(n+2)!}+\frac{tm^{n+1}}{(n+1)!}.$$

From the structure of the polynomial f, we have $s_1 = s_2 = \cdots = s_n$.

Calculation of s_{n+1}

Let us now consider s_{n+1}. We have,

$$s_{n+1} = \sum_{i_{n+1}=0}^{m+1}\sum_{i_{n+2}=0}^{i_{n+1}+t}\sum_{r=0}^{m+1-i_{n+1}} i_{n+1}\binom{n-1+r}{r} - \sum_{i_{n+1}=0}^{m}\sum_{i_{n+2}=0}^{i_{n+1}+t}\sum_{r=0}^{m-i_{n+1}} i_{n+1}\binom{n-1+r}{r}.$$

Now, $\sum_{i_{n+1}=0}^{m}\sum_{i_{n+2}=0}^{i_{n+1}+t}\sum_{r=0}^{m-i_{n+1}} i_{n+1}\binom{n-1+r}{r}$
$\approx \sum_{i_{n+1}=0}^{m}\sum_{i_{n+2}=0}^{i_{n+1}+t}\sum_{r=0}^{m-i_{n+1}} \frac{r^{n-1}}{(n-1)!}i_{n+1}$

$\approx \sum_{i_{n+1}=0}^{m} \frac{(m-i_{n+1})^n}{n!} i_{n+1}(i_{n+1}+t)$
(using Proposition 1 and neglecting lower order terms)
$\approx \frac{2m^{n+3}}{(n+3)!} + \frac{tm^{n+2}}{(n+2)!}$.

Hence, $s_2 \approx \dfrac{2m^{n+2}}{(n+2)!} + \dfrac{tm^{n+1}}{(n+1)!}$.

Calculation of s_{n+2}

$$s_{n+2} = \sum_{i_{n+1}=0}^{m+1} \sum_{i_{n+2}=0}^{i_{n+1}+t} \sum_{r=0}^{m+1-i_{n+1}} i_{n+1} \binom{n-1+r}{r} - \sum_{i_{n+1}=0}^{m} \sum_{i_{n+2}=0}^{i_{n+1}+t} \sum_{r=0}^{m-i_{n+1}} i_{n+1} \binom{n-1+r}{r}$$

Now, $\sum_{i_{n+1}=0}^{m} \sum_{i_{n+2}=0}^{i_{n+1}+t} \sum_{r=0}^{m-i_{n+1}} i_{n+1} \binom{n-1+r}{r}$
$\approx \sum_{i_{n+1}=0}^{m} \frac{(m-i_{n+1})^n}{n!} \frac{(i_{n+1}+t)^2}{2}$
$\approx \frac{m^{n+3}}{(n+3)!} + t\frac{m^{n+2}}{(n+2)!} + t^2 \frac{m^{n+1}}{(n+1)!2}$.

Hence, $s_{n+2} \approx \dfrac{m^{n+2}}{(n+2)!} + t\dfrac{m^{n+1}}{(n+1)!} + t^2 \dfrac{m^n}{n!2}$.