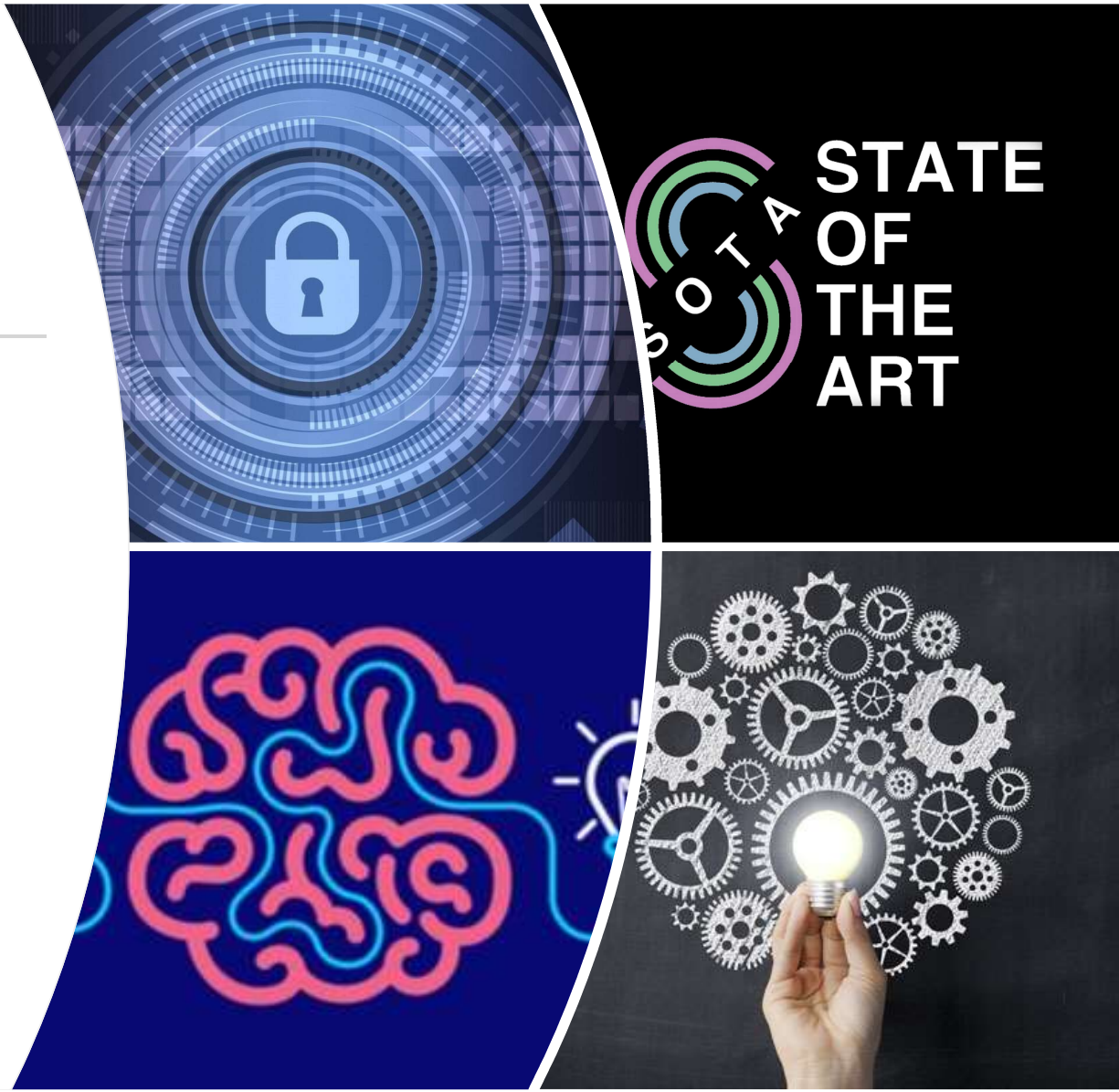
A Third is All You Need: Extended Partial Key Exposure Attack on CRT-RSA with Additive Exponent Blinding

Asiacrypt | Yuanyuan Zhou, Joop van de Pol, Yu Yu, and François-Xavier Standaert | 2022.12.09

Contents

- SOTA PKE Attack
- EPKE Principle
- EPKE Results
- Conclusion



(CRT)-RSA & Additive Exponent Blinding & PKE

- (CRT)-RSA key components: N, d, e, (p, q, dp, dq)
- Additive exponent blinding: private exponent adds a multiple of its group order (because of Euler's theorem)
  - $d' = d + r*phi(N)$
  - $d_p' = d_p + r_p*(p-1)$, $d_q' = d_q + r_q*(q-1)$
- PKE makes use of redundant information among those components
  - $ed = 1 + k(p-1)(q-1)$
  - $ed_p = 1 + k(p-1)$, $ed_q = 1 + l(p-1)$
  - $ed_p' = 1 + k'(p-1)$, $ed_q' = 1 + l'(p-1)$
  - Using partial leakage (consecutive MSB or LSB bits) of p, or d, or d_p, or (d_p, d_q), or d_p', or (d_p', d_q')

SOTA PKE Attack

Required Components	Constraint	Required bits
p	-	n/4
d	$e = \mathcal{O}(\log N)$	n/4
d_p	$e = \mathcal{O}(\log N)$	n/4
d	$d < N^{0.44}$	< n/4
d	$d < N^{0.36}$	< n/8
d	$d < N^{0.29}$	0
d_p, d_q	$d_p, d_q < N^{0.29}$	< 2 x n/4
d_p, d_q	$d_p, d_q < N^{0.19}$	< 2 x n/8
d_p, d_q	$d_p, d_q < N^{0.12}$	0
d_p, d_q	$e \leq N^{1/8}$	≤ 2 x n/4
d_p, d_q	$e \approx N^{1/12}$	2 x n/6

- No Exponent Blinding
  - EC MNS'22, $2\times\frac{n}{6}$

[FKS+, CHES'06]: Only for RSA, small e = 3 or 65537 and Sliding Window Exponentiations, blinding factor shorter than 32 bits, require $50\times(\frac{n}{64} \sim \frac{n}{16})$ partial key leakage

[MT, ISPEC'12]: Only for RSA, at least require $> \frac{n}{2}$ partial key leakage

[WA, JCEN'17]: for CRT-RSA, blinding factor shorter than 96-bit, require $2^{32}\times n$ partial key leakage (tolerate 10% error rate)

- Additive Exponent Blinding
  - This work, $2\times\frac{n}{6}$

Recap EC MNS'22 Work

Step 1: Solve CRT key constant k ($ed_p = 1 + k(p-1)$, $ed_q = 1 + l(q-1)$)

Step 2: Factor N using the estimated kp to get p (N = pq, $ed_p^{MSB} \approx kp \rightarrow p = AGCD(N, ed_p^{MSB})$)

Solve a quadratic polynomial equation to obtain k

Solve the root of univariate polynomial to get the unknown LSB of d_p and then factor N to obtain p

Polynomial time

- $2\times\frac{n}{6}$ MSB (d_p, d_q)
- $e \approx N^{1/12}$

Find the roots of bivariate polynomial using Coppersmith's method to obtain k

Solve the root of univariate polynomial to get the unknown MSB of d_p and then factor N to obtain p

Polynomial time

- $2\times\frac{n}{6}$ LSB (d_p, d_q)
- $e \approx N^{1/12}$

EPKE Attack on CRT Principle (I)

Step 1: Solve CRT key constant k' ($ed_p' = 1 + k'(p-1)$, $ed_q' = 1 + l'(q-1)$)

Factor k'l' to obtain k' (1/6 len(N))	Compute the GCD of two (or more) k'l' to obtain k'	Find the roots of bivariate polynomial using Coppersmith's method to obtain k'
Sub-exponential time	Probabilistic polynomial time	Polynomial time
▪ $2\times\frac{n}{6}$ MSB (d_p', d_q')	▪ $\geq 3\times\frac{n}{6}$ MSB (d_p', d_q')	▪ $2\times\frac{n}{6}$ LSB (d_p', d_q')
▪ $r_p^{2/3}e \approx N^{1/12}$	▪ $r_p^{2/3}e \approx N^{1/12}$	▪ $r_p e \approx N^{1/12}$

EPKE Attack on CRT Principle (II)

Step 2: Factor N using the estimated k'p to obtain p (MNS'22 Common Divisor Method)

Solve the root of univariate polynomial to get the unknown LSB of d_p' and then factor N to obtain p

Polynomial time

- $\geq 2 \times \frac{n}{6}$ MSB (d_p', d_q')
- $r_p^{2/3} e \approx N^{1/12}$

Solve the root of univariate polynomial to get the unknown MSB of d_p' and then factor N to obtain p

Polynomial time

- $2 \times \frac{n}{6}$ LSB (d_p', d_q')
- $r_p e \approx N^{1/12}$

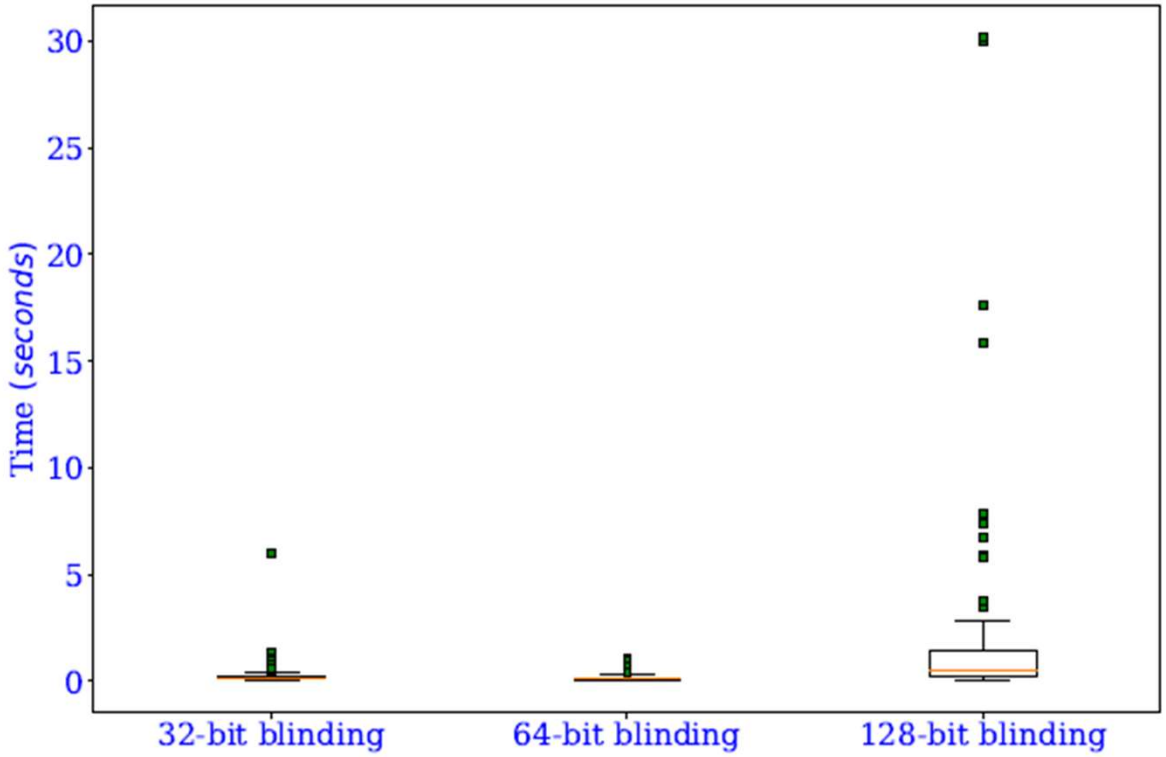
EPKE Attack on CRT Results (I) – MSB with d_q'

- Key Length: 1024, 2048 and 3072 bits; blinding factor: 32, 64 and 128 bits
- Sagemath v9.5, YAFU v2.08, Ubuntu 20.04.4, Intel Core i5-7500 3.4GHz

Len(N)	Len(e)	Len(r_p, r_q)	Len(UnknownLSB)	Step 1a Factoring time	Step 2 Lattice Dim.	LLL time
1024	64	32	336/**352**	<1 s	21	1s
1024	43	64	347/**362**	<1 s	21	1s
1024	17*	128	347/**401**	2s	21	2s
2048	149	32	665/**693**	42s	21	4s
2048	128	64	677/**704**	175s	21	4s
2048	85	128	697/**725**	340s	21	4s
3072	235	32	1008/**1034**	1787s	31	60s
3072	213	64	1014/**1045**	5993s	31	60s
3072	171	128	1032/**1066**	6651s	31	60s

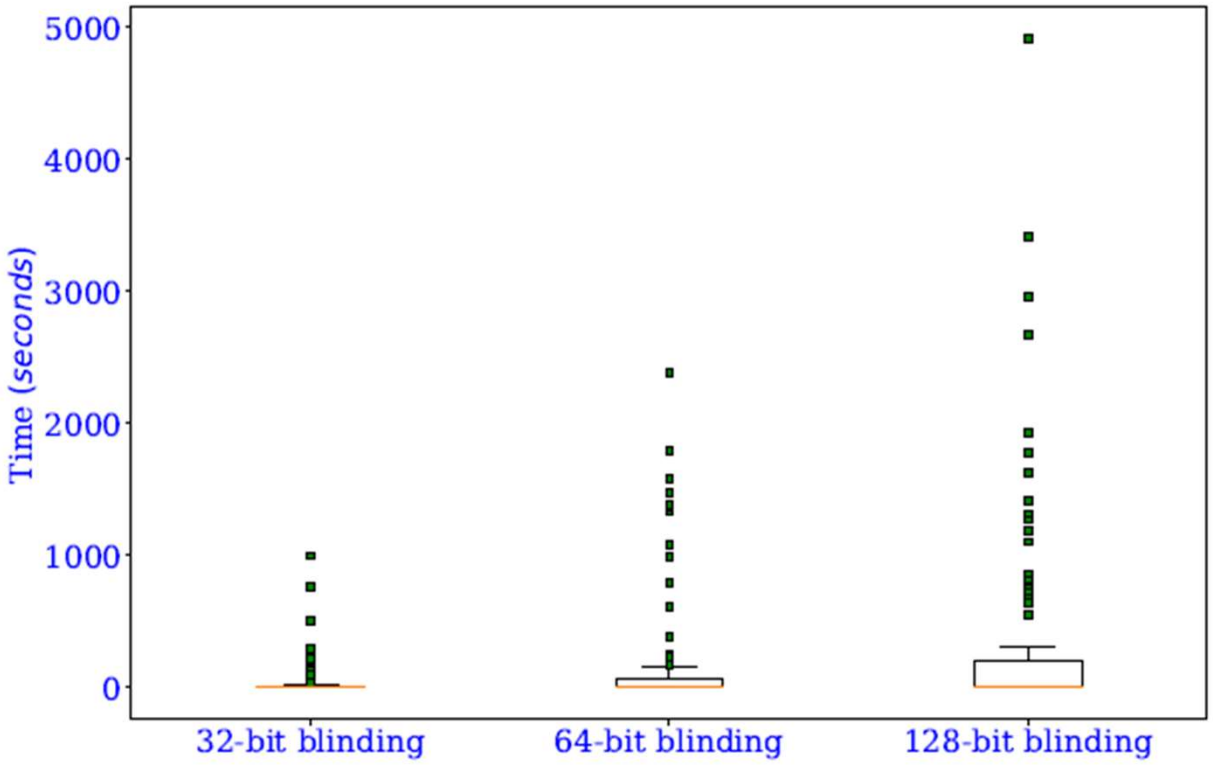
EPKE Attack on CRT Results (I) – Factoring Time

- Key Length: **1024 bits;** blinding factor: **32, 64 and 128 bits**
- Sagemath v9.5, YAFU v2.08, Ubuntu 20.04.4, Intel Core i5-7500 3.4GHz



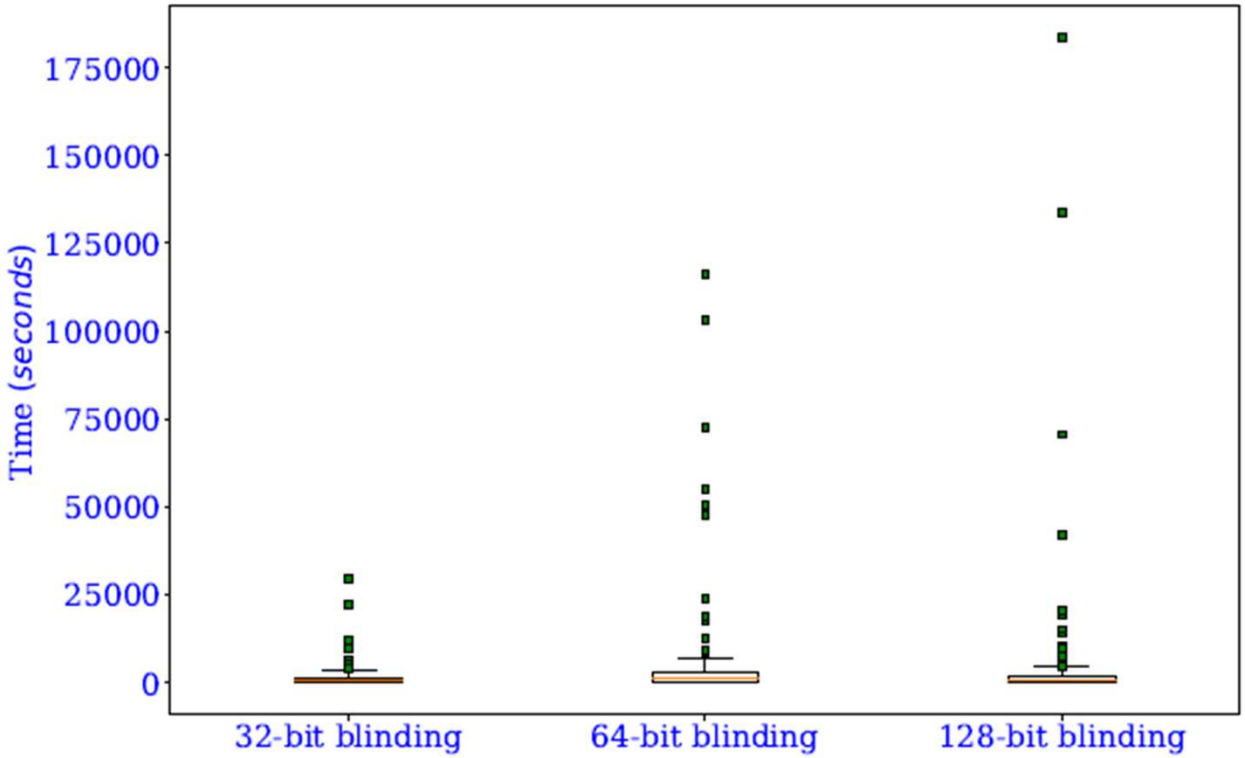
EPKE Attack on CRT Results (I) – Factoring Time

- Key Length: **2048 bits;** blinding factor: **32, 64 and 128 bits**
- Sagemath v9.5, YAFU v2.08, Ubuntu 20.04.4, Intel Core i5-7500 3.4GHz



EPKE Attack on CRT Results (I) – Factoring Time

- Key Length: **3072 bits;** blinding factor: **32, 64 and 128 bits**
- Sagemath v9.5, YAFU v2.08, Ubuntu 20.04.4, Intel Core i5-7500 3.4GHz



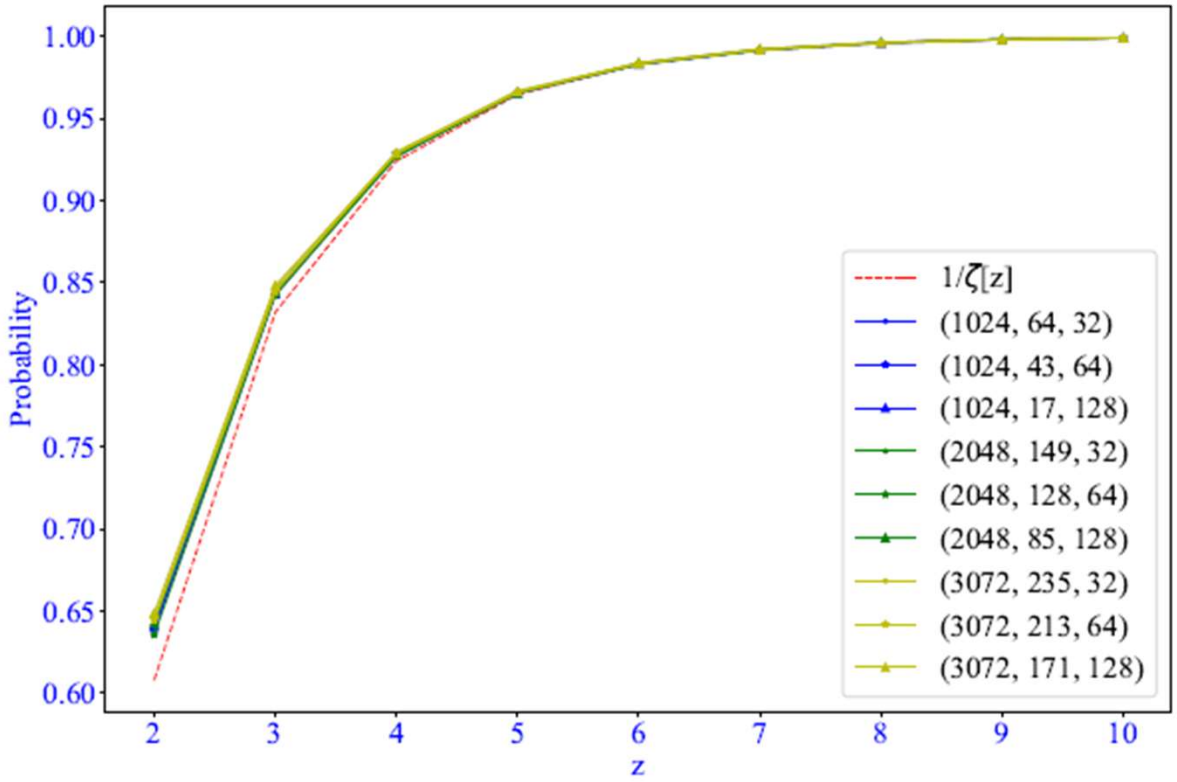
EPKE Attack on CRT Results (II) – MSB with $d_{q,i}$'

- Key Length: 1024, 2048 and 3072 bits； blinding factor: 32, 64 and 128 bits
- Sagemath v9.5, YAFU v2.08, Ubuntu 20.04.4, Intel Core i5-7500 3.4GHz

Len(N)	Len(e)	Len(r_p, r_q)	Len(UnknownLSB)	Step 1b Success Prob.
1024	64	32	336/**352**	0.67/0.89/0.93/0.97/0.98/0.99/0.98/0.98/1.00
1024	43	64	347/**362**	0.65/0.89/0.91/0.98/0.97/0.99/0.99/1.00/0.99
1024	17*	128	347/**401**	0.65/0.78/0.94/0.99/0.96/0.99/0.99/1.00/1.00
2048	149	32	665/**693**	0.67/0.79/0.89/0.95/0.99/1.00/0.98/0.99/1.00
2048	128	64	677/**704**	0.73/0.86/0.92/0.93/0.98/1.00/1.00/1.00/1.00
2048	85	128	697/**725**	0.73/0.86/0.92/0.95/0.99/1.00/0.99/1.00/1.00
3072	235	32	1008/**1034**	0.69/0.81/0.93/0.98/0.98/0.99/1.00/1.00/1.00
3072	213	64	1014/**1045**	0.72/0.82/0.94/0.98/0.98/0.99/1.00/1.00/1.00
3072	171	128	1032/**1066**	0.67/0.81/0.93/1.00/0.99/1.00/1.00/1.00/1.00

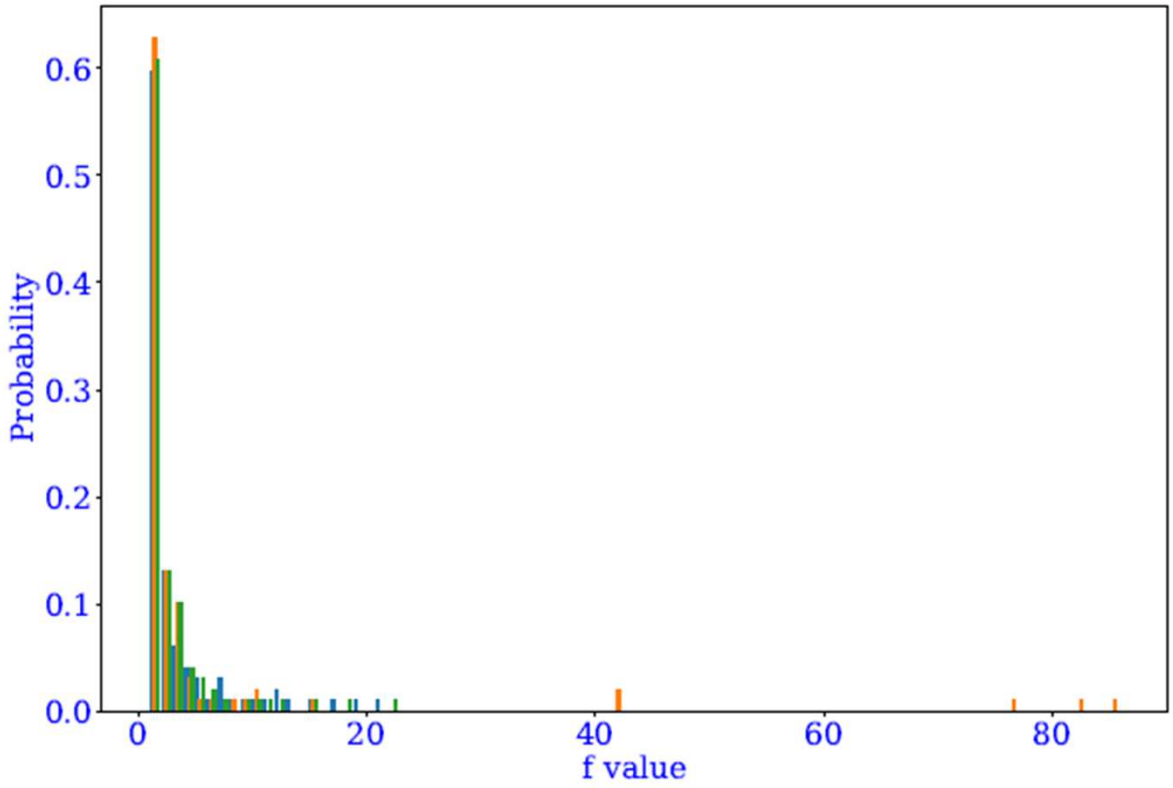
EPKE Attack on CRT Results (II) – Success Prob.

- Key Length: 1024, 2048 and 3072 bits; blinding factor: 32, 64 and 128 bits
- Sagemath v9.5, YAFU v2.08, Ubuntu 20.04.4, Intel Core i5-7500 3.4GHz



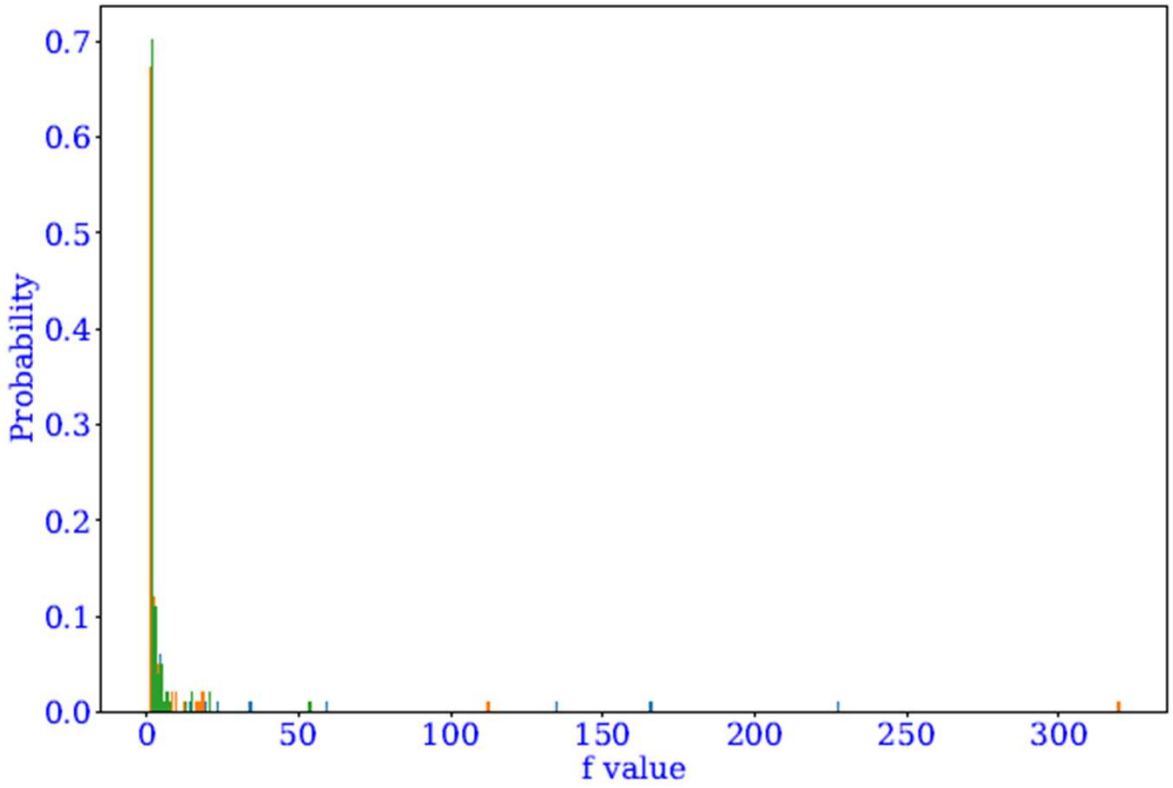
EPKE Attack on CRT Results (II) – f-value Distrib.

- Key Length: **1024 bits;** blinding factor: **32, 64 and 128 bits;** f = GCD(l_1', l_2')
- Sagemath v9.5, YAFU v2.08, Ubuntu 20.04.4, Intel Core i5-7500 3.4GHz



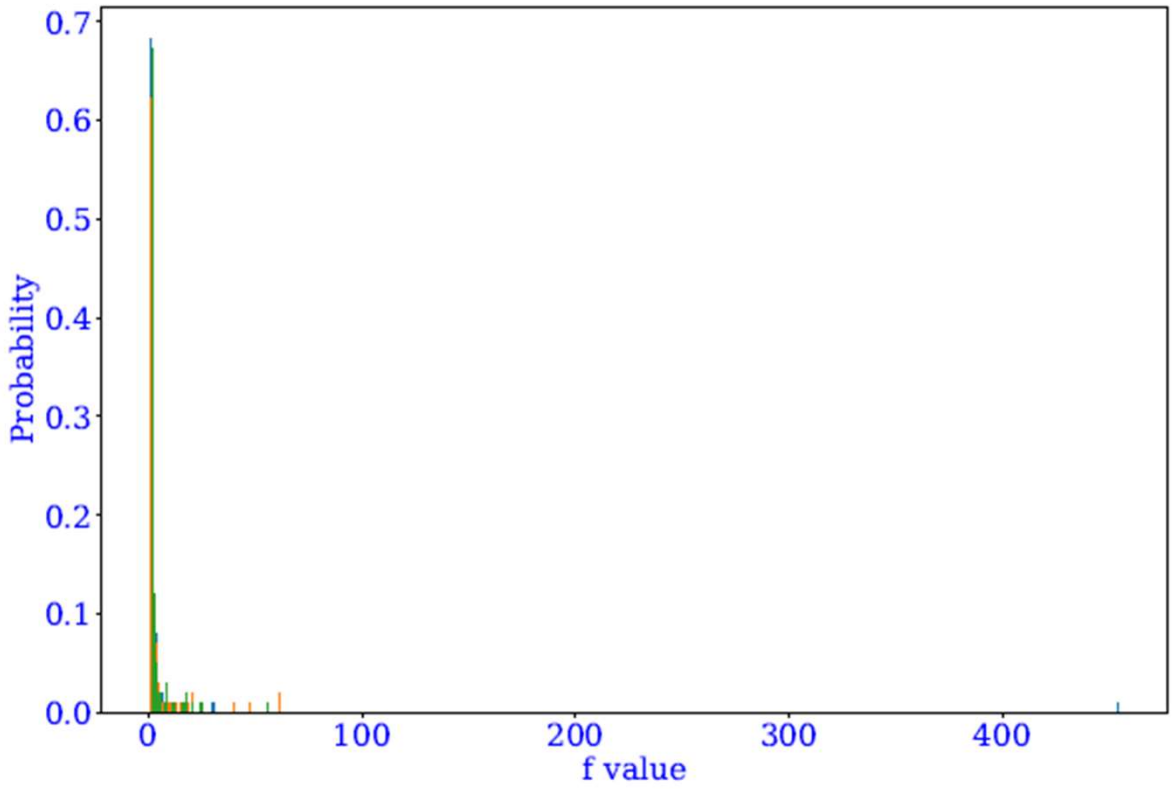
EPKE Attack on CRT Results (II) – f-value Distrib.

- Key Length: 2048 bits; blinding factor: 32, 64 and 128 bits; f = GCD(l_1', l_2')
- Sagemath v9.5, YAFU v2.08, Ubuntu 20.04.4, Intel Core i5-7500 3.4GHz



EPKE Attack on CRT Results (II) – f-value Distrib.

- Key Length: 3072 bits; blinding factor: 32, 64 and 128 bits; f = GCD(l_1', l_2')
- Sagemath v9.5, YAFU v2.08, Ubuntu 20.04.4, Intel Core i5-7500 3.4GHz



EPKE Attack on CRT Results (III) – LSB with d_q'

- Key Length: 1024, 2048 and 3072 bits； blinding factor: 32, 64 and 128 bits
- Sagemath v9.5, YAFU v2.08, Ubuntu 20.04.4, Intel Core i5-7500 3.4GHz

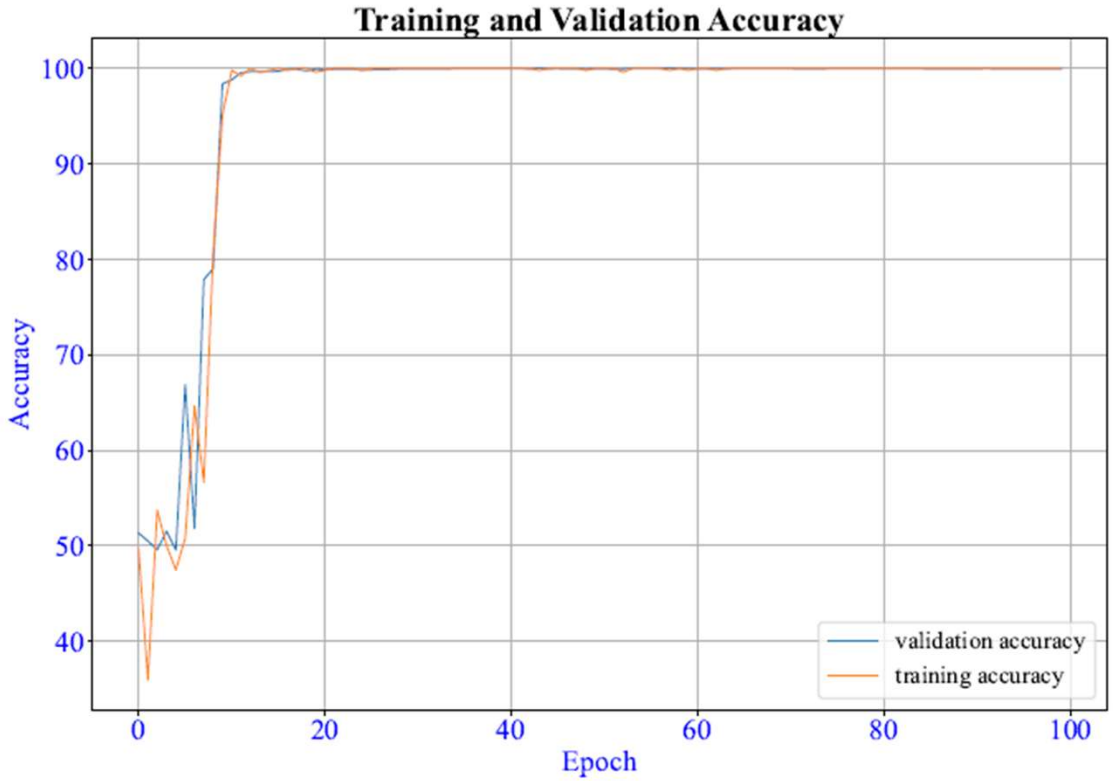
Len(N)	Len(e)	Len(r_p, r_q)	Len(UnknownMSB)	Step 1c Lattice Dim.	LLL time	Step 2 Lattice Dim.	LLL time
1024	53	32	320/**341**	121	216s	21	<1 s
1024	21	64	320/**341**	121	202s	21	<1 s
1024	17*	128	191/**222**	121	461s	21	4s
2048	139	32	648/**682**	121	487s	21	4s
2048	107	64	647/**682**	121	470s	21	4s
2048	43	128	649/**682**	121	419s	21	4s
3072	224	32	978/**1024**	121	1104s	31	90s
3072	192	64	978/**1024**	121	1110s	31	90s
3072	128	128	976/**1024**	121	991s	31	87s

Real-World EPKE Attack on CRT

- Potential partial key leakage sources: side-channel attacks (SCA); cold-boot attacks; micro-architectural attacks
- SCA background
  - Different side channels: timing; power consumption; electromagnetic emission…
  - SCA stages: measurement; pre-processing; analysis; post-analysis
  - SCA methods: SPA/SEMA; DPA/DEMA; TA; SA; DLSCA
  - Profiling & non-profiling SCA
- Potential impact on our real life
  - Often embedded devices use CRT (for performance) + Additive exponent blinding (against SCA attacks)
  - EMVCo alone circulates ~12 Billion chip cards internationally (end of 2021) and (CRT)-RSA is used by default
  - What if we obtain 1/3 MSB or LSB blinded d_p' and d_q' via SCA?

SCA Application of EPKE Attack on CRT

- CRT-2048 + 64-bit additive blinding + Montgomery Ladder modular exponentiation
- DLSCA to recover 411-bit MSB or 441-bit LSB blinded exponents (10 attack keys per case)



SCA Application of EPKE Attack on CRT

- CRT-2048 + 64-bit additive blinding + Montgomery Ladder modular exponentiation
- DLSCA to recover 411-bit MSB or 441-bit LSB blinded exponents (10 attack keys per case)

Attack Key	Success Nr.	Step 1a Factoring time	Step 2 Lattice Dim.	LLL time
K_{MSB1}	26	110s ([0.93s, 599.7s])	21	6s
K_{MSB2}	35	90s ([0.5s, 1379.7s])	21	6s
K_{MSB3}	15	176s ([1.0s, 1817.7s])	21	6s
K_{MSB4}	3	10s ([6.1s, 16.4s])	21	6s
K_{MSB5}	9	202s ([1.5s, 1211.8s])	21	6s
K_{MSB6}	4	251s ([59.7s, 477.9s])	21	6s
K_{MSB7}	4	65s ([2.1s, 235.7s])	21	6s
K_{MSB8}	4	212s ([1.6s, 775.2s])	21	6s
K_{MSB9}	4	969s ([764.4s, 1373.7s])	21	6s
K_{MSB10}	12	21s ([2.6s, 60.7s])	21	6s

SCA Application of EPKE Attack on CRT

- CRT-2048 + 64-bit additive blinding + Montgomery Ladder modular exponentiation
- DLSCA to recover 411-bit MSB or 441-bit LSB blinded exponents (10 attack keys per case)

Attack Key	Success Nr.	Step 1c Lattice Dim.	LLL time	Step 2 Lattice Dim.	LLL time
K_{LSB1}	6	121	388s	21	2s
K_{LSB2}	2	121	404s	21	2s
K_{LSB3}	4	121	407s	21	3s
K_{LSB4}	3	121	421s	21	3s
K_{LSB5}	8	121	410s	21	3s
K_{LSB6}	5	121	407s	21	3s
K_{LSB7}	9	121	399s	21	3s
K_{LSB8}	12	121	405s	21	3s
K_{LSB9}	16	121	442s	21	3s
K_{LSB10}	9	121	433s	21	3s

Takeaways

- 1/3 MSB or LSB of additively blinded d_p' and d_q' to recover the CRT key
  - Similar to MNS'22 results without blinding, don't need "more" bits
  - Price to pay: factoring an integer $\approx N^{1/6}$ or calculating GCD using multiple recovered 1/3 MSBs of $d_{q,i}$' (GCD has success probability, feasible for a small brute force with only two $d_{q,i}$')
- Encountered hurdles to apply MNS'22 to additive blinding
  - Due to blinding: non-trivial to derive k'+l' from k'+l' mod e; non-trivial to derive kl from kl mod e = k'l' mod e; GCD(2^ie, k'N) = 1 doesn't always hold
- Splicing two different measurements to make the PKE attack on larger blinding factor easier
  - Especially useful for embedded devices: measuring only 1/3 exponentiation greatly speeds up acquisition
  - Verified in practice through our experiments

- Future work: How to deal with erroneous bits of d_p' and d_q'?

Thanks for your attention