Applications of LLL: Breaking RSA

Phong Nguyễn

Summary

- RSA
- Lattice Attacks on RSA
  - « Linear » Attacks
    - Wiener's Attack
    - Bleichenbacher's Attack
  - Small-Root Attacks

The RSA Cryptosystem

Remember RSA

- N=pq product of two large random primes.
- $ed \equiv 1 \pmod{\phi(N)}$ where $\phi(N)=(p-1)(q-1)$.
  - e is the public exponent
  - d is the secret exponent
- Then $m \to m^e$ is a trapdoor one-way permutation over $\mathbf{Z}/N\mathbf{Z}$, whose inverse is $c \to c^d$.

Wiener's Attack (1989)

Short-Secret RSA

- To speed-up RSA secret operations, we may want to select a short d.
- Assume that d ≪ N, can we recover d from (e,N)?
- $ed = 1+k\phi(N)$

  where $\phi(N)=(p-1)(q-1)=N+O(\sqrt{N})$

- So $k=O(d)$ and $ed \approx kN$, namely $ed-kN=O(d\sqrt{N})$.

Lattices and Short-Secret RSA

- Consider the 2-dim lattice L spanned by:

e	√N
N	0

- It contains the vector t=dx(1st row)-kx(2nd row).

Lattices and Short-Secret RSA

- How short is $t=dx(1st\ row)-kx(2nd\ row)$?
  - Its 1st coordinate is $ed-kN=O(d\sqrt{N})$.
  - Its 2nd coordinate is $d\sqrt{N}$.
- So $||t||=O(d\sqrt{N})$.
- This is unusually short if $||t|| \le vol(L)^{1/2}=N^{3/4}$ i.e. $d \le O(N^{1/4})$, then t is ``likely'' to be a shortest vector of L.

Lattice Attack on Short-Secret RSA

- Compute a shortest vector of the 2-dim lattice L: this only takes polynomial-time, less than 1s for 2048-bit RSA.
- If it is ±t, recover (k,d): how?
- Check that (k,d) is correct: how?

Lattice Attack on Short-Secret RSA

- If it is $\pm t$, recover (k,d): how?
  - Divide the 2nd coordinate by $\sqrt{N}$.
- Check that (k,d) is correct: how?
  - $ed-kN=1-k(p+q-1)$.
  - Derive $p+q$.
  - Recover p and q by solving $X^2-(p+q)X+N=0$.

Wiener's Attack (1989)

- Using continued fractions instead of lattices, Wiener showed:
  - Th: If $q<p<2q$ and $1 \le d \le N^{1/4}/3$, one can recover p and q in polynomial time from (N,e).
- [BonehDurfee1999]: There is a heuristic (lattice) attack recovering p and q in polynomial time from (N,e) if $d \le N^{0.292\ldots}$

Bleichenbacher's Attack (1998)

Security of RSA

- It is well-known that recovering the secret key d is as hard as factoring N=pq.
- But inverting the permutation $m \rightarrow m^e$ might be easier than

  factoring: how hard is it to compute $c \rightarrow c^d$ without knowing d?

Textbook RSA Encryption

- Just use the RSA permutation $m \rightarrow m^e$ over $\mathbb{Z}/N\mathbb{Z}$ directly to encrypt and decrypt.
- But this is insecure in theory and in many practical settings.

Short-Message Attack

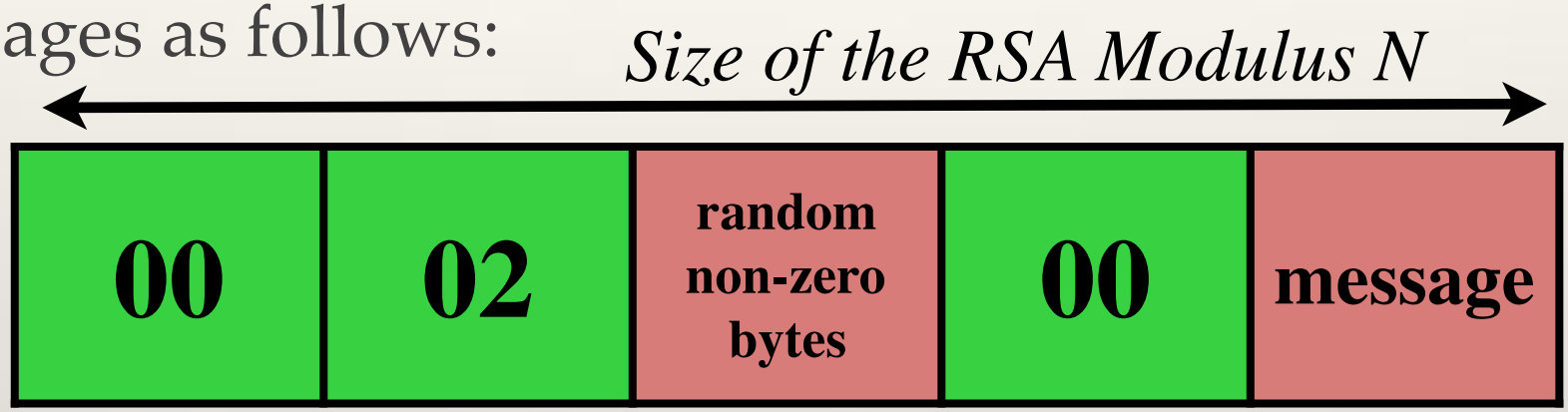
- Assume that e=3, N is 2048-bit, and that we encrypt a 128-bit AES key m.
  - $c=m^e \pmod N$.
  - What is the problem?

Securing RSA Encryption

- Encryption needs to be randomized.
- Concretely, one (randomly) transforms the plaintext, before applying the RSA permutation.
- To decrypt, the formatting must be checked and inverted.

Bleichenbacher's Attack (1998)

- The PKCS standard used by SSLv3.0 preprocessed messages as follows:



- The server checked if decrypted messages had this shape: if not, an error message was sent.
- Thus, one could know if the decryption of a ciphertext started with 00 02.

Bleichenbacher's Attack (1998)

- Given a public key (N,e) and a ciphertext $c=m^e \pmod N$.
  - Choose a random r mod N and let $c' = c\, r^e \pmod N$.
  - Ask the server if c' is a valid ciphertext.
    - If not, pick another r.
    - If yes, we know that $c'^d = mr \pmod N$ starts with 00 02.
  - Repeat until enough r's have been collected: recover m using a special algorithm.

Enter Lattices

- Recall that $c=m^e \pmod N$.
- We know many integers r_i s.t. $(mr_i) \bmod N$ starts with 0002.
- Let s=00020...0 with the same bit-length as N.
- Then $0 \le ((mr_i) \bmod N) - s < N/2^{16}$

Enter Lattices

- Let L be the lattice spanned by:

$$\begin{pmatrix} 1 & 2^{16}r_1 & 2^{16}r_2 & \dots & 2^{16}r_n \\ 0 & 2^{16}N & 0 & \dots & 0 \\ \vdots & 0 & 2^{16}N & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & 2^{16}N \end{pmatrix}$$

- Then $u=(m, 2^{16}(mr_1 \bmod N), \dots, 2^{16}(mr_n \bmod N)) \in L$

Enter Lattices

- Let L be the lattice spanned by:

$$\begin{pmatrix} 1 & 2^{16}r_1 & 2^{16}r_2 & \dots & 2^{16}r_n \\ 0 & 2^{16}N & 0 & \dots & 0 \\ \vdots & 0 & 2^{16}N & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & 2^{16}N \end{pmatrix}$$

- Let $u=(m, 2^{16}(mr_1 \bmod N), \dots, 2^{16}(mr_n \bmod N)) \in L$ and $t=(0,s,s \dots s)$, then $\|u-t\| \approx N\sqrt{n}$.

Enter Lattices

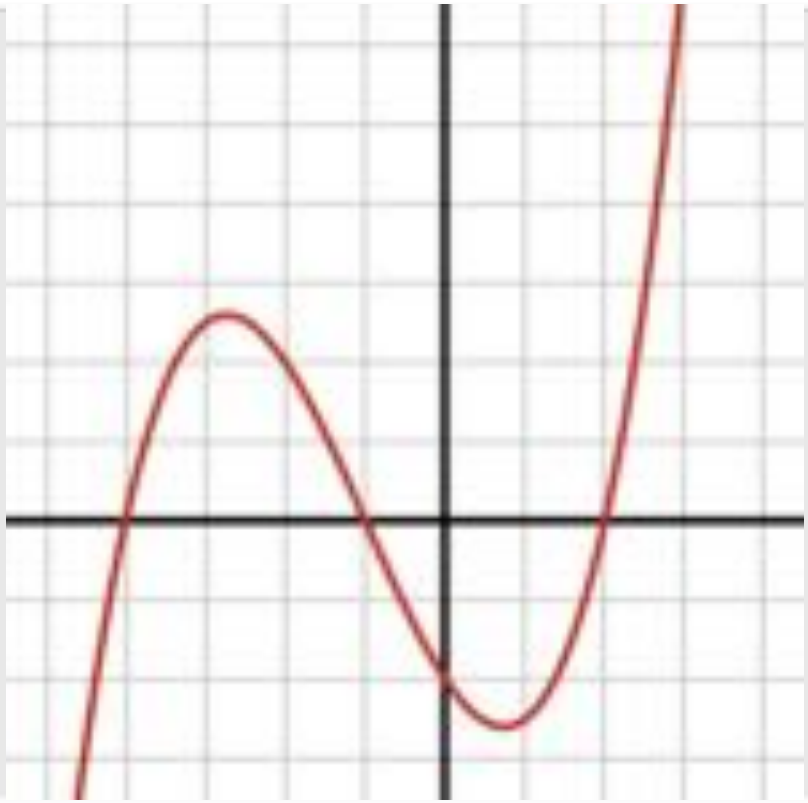
- Let L be the lattice spanned by:

$$\begin{pmatrix} 1 & 2^{16}r_1 & 2^{16}r_2 & \dots & 2^{16}r_n \\ 0 & 2^{16}N & 0 & \dots & 0 \\ \vdots & 0 & 2^{16}N & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & 2^{16}N \end{pmatrix}$$

- This distance $\|u-t\| \approx N\sqrt{n}$ is much smaller than $\sqrt{n}\,\mathrm{vol}(L)^{1/(n+1)} \approx 2^{16}N\sqrt{n}$ so u is heuristically the closest lattice vector to t.

Recap

- Build the lattice L.
- Compute the closest lattice vector to the target vector t.
- Derive the plaintext m as the 1st coordinate.
- If N is 1024-bit, n=80 works in practice.



Small-Roots Attacks

Breaking RSA without Factoring

- In 1996, Coppersmith showed how to solve two problems in polynomial time using lattices:
  - Given a monic polynomial P in **Z**[X] and an integer N, find all "small" integers x s.t. $P(x) \equiv 0 \pmod{N}$.
  - Given an irreducible polynomial P in **Z**[X,Y], find all "small" integers x and y s.t. $P(x,y) = 0$.

Applications to RSA

- This and generalizations lead to breaking many special cases of RSA
  - When the secret exponent d is too small.
  - When half of the bits of p are known.
  - When the public exponent e is small, and only a fraction of the plaintext is unknown.

Stereotyped Attack

- Assume that e=3, N is 2048-bit, and that we encrypt a 128-bit AES key m by padding a known constant like « Today's key is ».
  - $c=(m+b)^e \pmod N$.
  - What is the problem?

Factoring with a hint [Cop96]

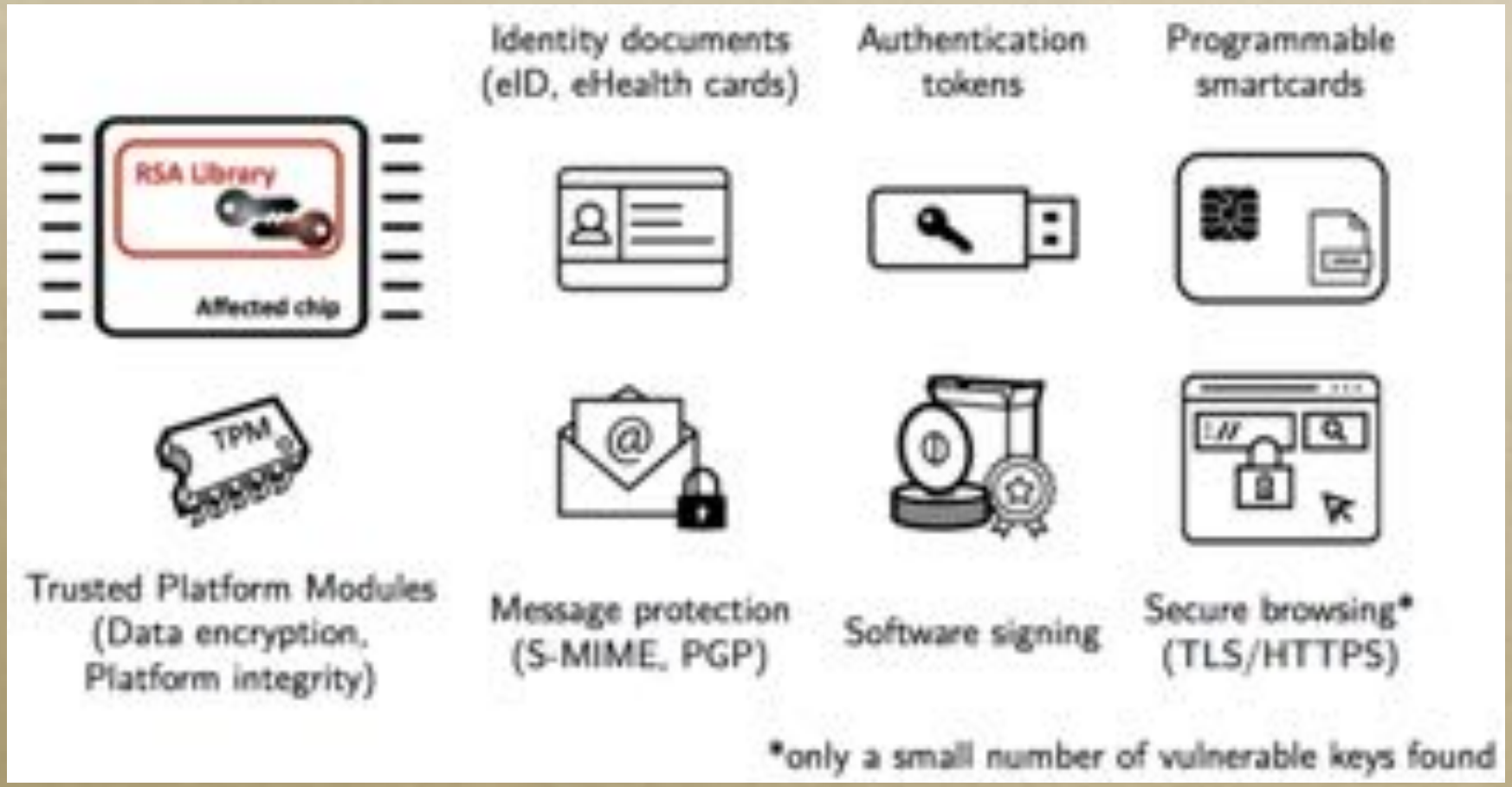
- N=pq where p=p_0+ ε for some small ε .
- Let f(x)=p_0+x.
- Then gcd(f(ε),N)=p is large.
- Can recover ε and p if | ε |≤$N^{1/4}$

Another Real-World Attack

- Attack on Infineon RSA keys.

- See **ACM CCS '17: The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli by Matus Nemec, Marek Sys, Petr Svenda, Dusan Klinec, Vashek Matyas (Masaryk University).**

Impact



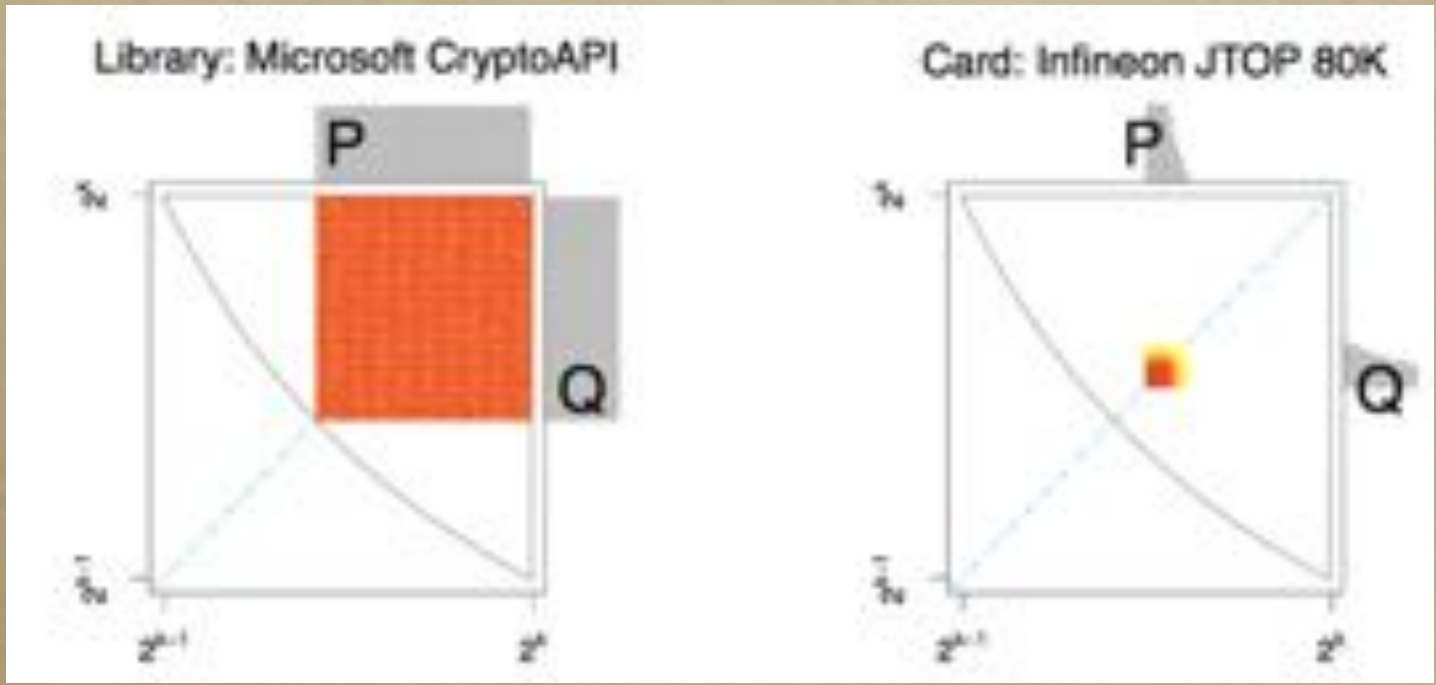
Ex: Estonia's 750,000 ID cards.

Identifying RSA keys

- Svenda et al. analyzed 60 millions fresh keys produced by 22 libraries and 16 smartcards from 6 manufacturers.
- Most distributions of N=pq and/or p were different and could be identified!

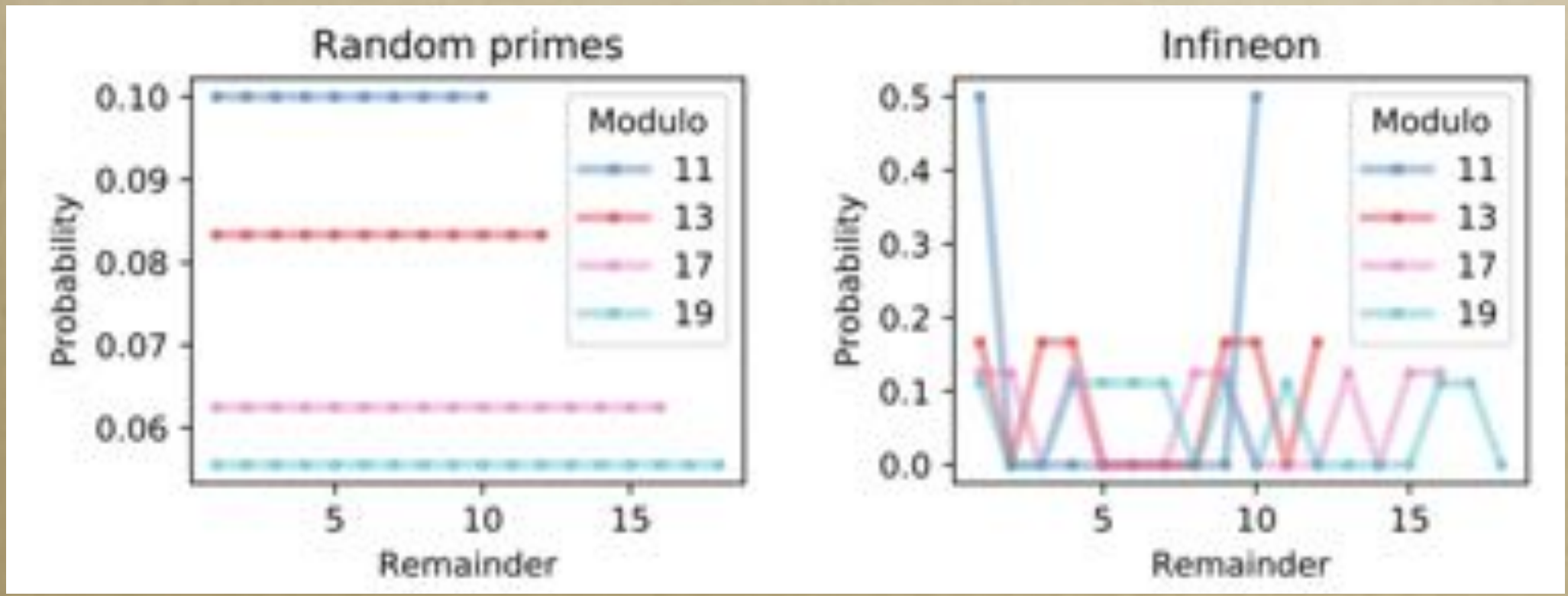
Why?

- If p and q are random primes, then (p-1)(q-1) may not be coprime with e, and N=pq will not have a fixed bit-length.
- Each manufacturer/library typically has their own distribution.



Ex: Infineon

- Infineon primes are « not random »



What is going on?

- If p_i is a small prime then p mod p_i is not uniform over $\{1,\ldots, p_i-1\}$.
- It seems to be uniform over some **small subgroup** of $(\mathbb{Z}/p_i\mathbb{Z})^*$.

Why?

- Typically, one generates primes as:
  - Repeat
    - Generate a large random number p
  - Until p is prime
- In practice, primality testing is a few modular exponentiations. One can increase the probability by making p not divisible by all small p_i.

Generation

- The subgroup of $(\mathbb{Z}/p_i\mathbb{Z})^*$ is the one generated by 65537.
- p and q are of the form:
  - $p=kM+(65537^a \bmod M)$, where M is the product of the first n primes: 2x3x5x...
  - n depends on the size of N.
- Hence, N mod M is a power of 65537, which can easily be checked.

Breaking Infineon-RSA

- $p = kM + (65537^a \bmod M)$
- If one can guess the exponent a, then $p \bmod M$ is known.
- From Coppersmith's 1996 work: if $M \geq N^{0.25}$, lattice attacks recover p in poly-time from N.

N	512-bit	1024-bit	2048-bit	3072-bit	4096-bit
$(\log_2 M)/(\log_2 N)$	0.43	0.46	0.47	0.32	0.48

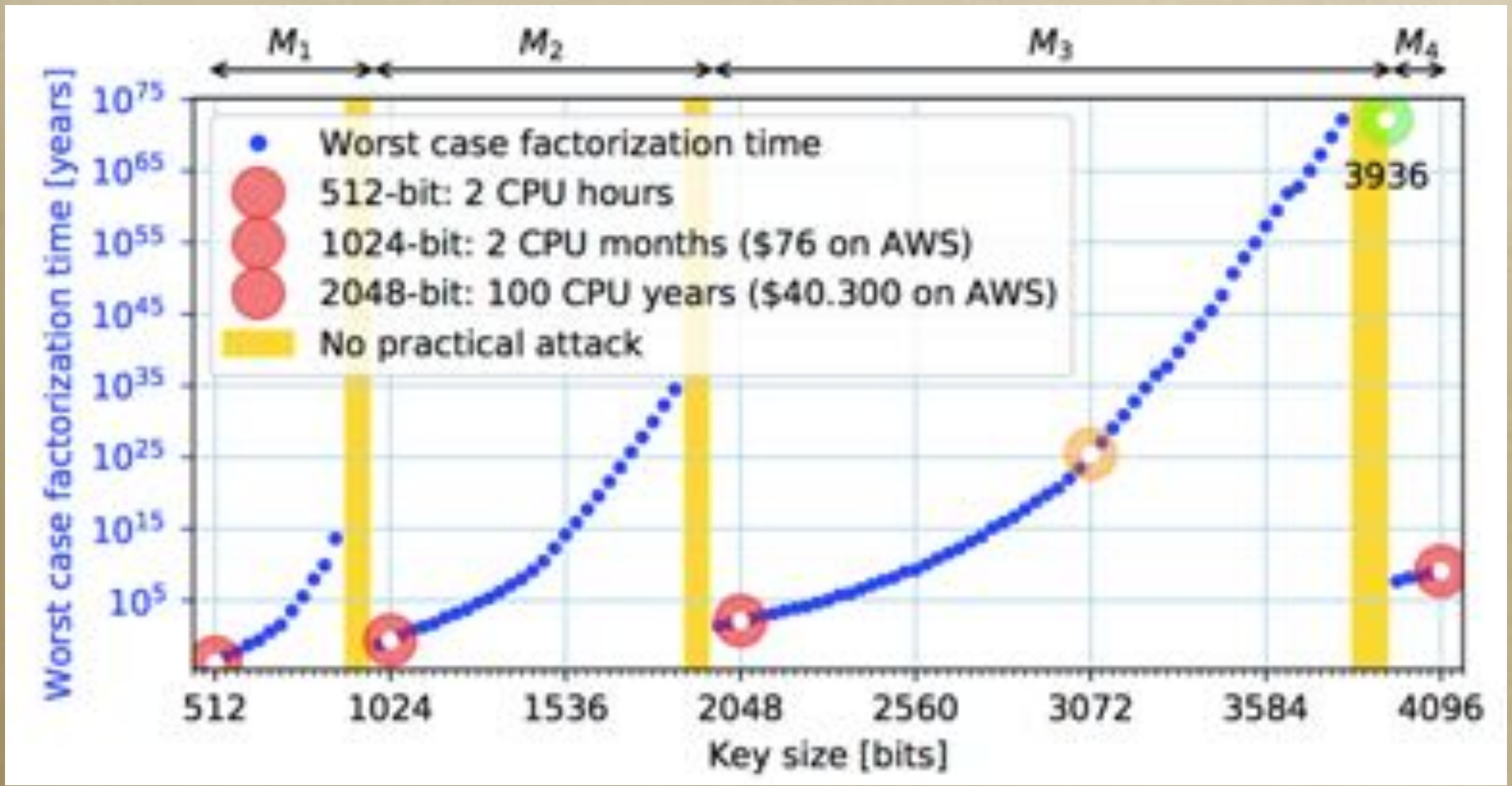
Lattice Attacks

- If p mod M is known, one knows a linear polynomial $f(X) \in \mathbf{Z}[X]$ s.t. $\gcd(f(x_0),N)=p$ is large, where x_0 is a small integer: it is small if M is large.
- This can be solved by lattice techniques [Cop1996].

The Trick

- Guessing a depends on the order of 65537 in $(\mathbb{Z}/M\mathbb{Z})^*$, which might be as big as $M \geq N^{0.4}$: exhaustive search too expensive!
- However, no need to take M: take any divisor M′ of M s.t. $M' \geq N^{1/4}$ and the order of 65537 in $(\mathbb{Z}/M'\mathbb{Z})^*$ is small.
  - Ex: 20-bit order for 512-bit N, 30-bit order for 1024-bit.

Implementation



Non-linear!