Real-World Cryptanalysis

Logistics

* PS4 out now! (due 5/25)
* PS3 graded soon
  ↳ very good results so far!
* Piazza poll on last 2 lectures
* Research! Grad school! Come ask us!
* Events this week

David's defense
Lattices & NIZK
May 18, 1:30pm, Gates 498

Daniel Kohlbrenner (UCSD)
May 16, 4:15 pm, Gates 400
Side-channel attacks in browser

Today

- Recap of Post Quantum Crypto
- GCD Attack on RSA (2012)
- Infineon Attack (2017)

Recap

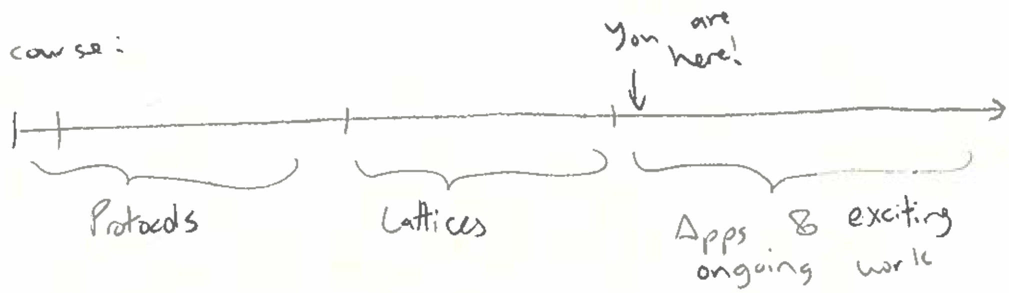
QC break dlog, RSA

Alternatives: Hash-based, lattice based, code-based, isogeny, ...

Problems: * Large key sizes
* Unclear security (new/recent assumptions)
* Q adversaries hard to understand .... how many qubits we need?

Real-World Cryptanalysis

This course:



Today will talk about two recent developments in real-world cryptanalysis

↳ Theme: Crypto often breaks b/c of misuse { Poor API design, Cross-stack confusion, Bugs in impl...

1) GCD Attack
2) Infineon attack

Neither attack came from "direct" cryptanalysis (i.e., new factoring algs)

↳ came from indirect misuse/failure

GCD Attack

As far as I know, discovered independently by
Lenstra et al. (2012)
Heninger et. al. (2012) ← [Zakir D. (future Stanford prof) heavily involved w/ this paper

Background

* RSA used all over the place for encryption / signatures
  SSH, TLS, IPSec, ...

* If you connect to an SSH/TLS host on Internet, it will send you its RSA pk.

$pk = (N, e)$

N: product of large random primes

e: public exponent (often $e = 3$)

Idea: Scan entire IPv4 address space, collect all pks

aaa.bbb.ccc.ddd ⇒ 32-bit address $2^{32} \approx 4$ billion

↳ With fast net connection, takes < 1 hour

Found huge vulnerabilities: 0.50% of TLS private keys recovered (64k)
0.03% of SSH " " " (2.5k)

GCD Attack

What happened?

Found many RSA moduli sharing <u>one</u> common factor

$$N = p \cdot q \qquad N' = pr \quad (\text{for } p, q, r \text{ distinct primes})$$

Both N and N' are hard to factor independently, but given both, can compute

$$\gcd(N, N') \longrightarrow p \quad \text{in poly time}$$

[Uses Euclid's algorithm... possibly oldest known algorithm (300 B.C.)]

Given p, can factor N and N' using division.

Problem: Researchers downloaded $\approx 2^{24}$ keys. Computing pairwise gcd takes

$$\Omega(k^2) \cdot \text{gcd computations} \Longrightarrow \approx 2^{50} \text{ gcds!}$$

This is a huge amount of work.

Better idea: GCD Tree (Bernstein). Compute all gcds at once

$$\begin{tikzcd}[column sep=small]
N_1 \arrow[dr] & & N_2 \arrow[dl] & & N_3 \arrow[dr] & & N_4 \arrow[dl] \\
& N_1 N_2 \arrow[drr] & & & & N_3 N_4 \arrow[dll] & \\
& & & N_1 N_2 N_3 N_4 & & &
\end{tikzcd}$$

$\leftarrow$ Product of all moduli

Takes time (very roughly) $O(k \cdot \text{polylog } k)$ [ignoring size of modulus... see paper for detailed analysis]

Once you have product, can extract out all gcds

$$N_1 N_2 N_3 N_4$$

$$\text{mod } N_1^2 N_2^2 \qquad \text{mod } N_3^2 N_4^2$$

$$(*) \quad \text{mod } N_1^2 \qquad \text{mod } N_2^2 \qquad \text{mod } N_3^2 \qquad \text{Mod } N_4^2$$

$$\gcd(\cdot/N_1, N_1) \qquad \gcd(\cdot/N_2, N_2) \; \ldots$$

Idea: Remainders at level (*) keep duplicate prime factors around.

Say $N_1 = pq \qquad N_3 = pr$

$$N_1 \cdot N_2 N_3 N_4 \quad \text{mod} \quad N_1^2$$

$$= p^2 q r$$

Divide by N_1

$$= pr$$

gcd w/ N_1

$$p$$

How did this happen?!

I. Very bad implementation IBM management devices.

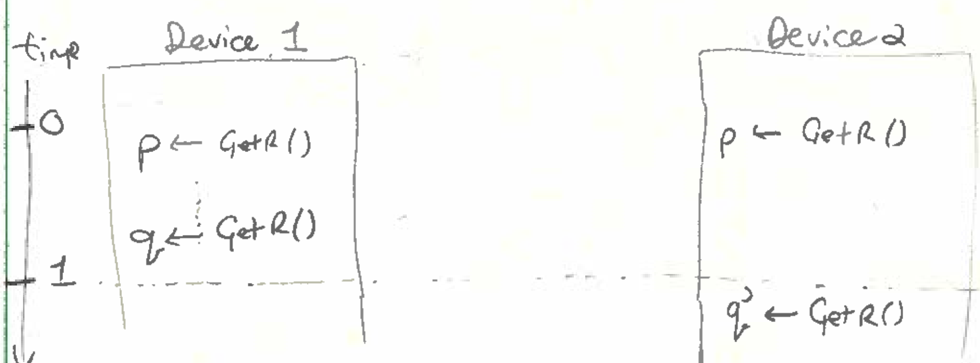
```
KeyGen() {
    // Hardcoded values - 9 primes
    P = {p_1, p_2, ..., p_9}
    choose distinct p_i, p_j from P
    output N = p_i * p_j as pk
}
```

$\Rightarrow$ Only $\binom{9}{2} < 100$ possible public keys!

II. Unfortunate confluence of unlucky events.

* Many embedded devices (e.g. net routers) speak TLS/SSH
* Linux gathers "random" values from I/O devices
  ↳ Keyboard/mouse/HD/net timings, etc.
* Embedded device has few peripherals
* On first boot, two devices have same state
* On first boot, device generates RSA key

```
KeyGen()
    p ← GetRandomPrime()      {  x ← Hash(state, time())
    q ← GetRandomPrime()      {  state ← Hash(state, x)
    output N = p*q as pk      {  return smallest prime > x.
```



.... what would you do to fix this???

Infineon Attack (2017)

- One of the most shocking attacks in recent memory
  - ↳ tens of millions of smartcards affected & recalled
- Paper is amazing. Really impressive piece of work (IMO)

Background
* RSA-Type cryptosystems are surprisingly fragile
* Many many variants of RSA are insecure (see pset)
* Developers trying to be clever/efficient can introduce new vulnerabilities by mistake.

Standard RSA KeyGen

$$p, q \xleftarrow{R} \{\lambda\text{-bit primes}\}$$

output $N = p * q$ as modulus

To generate primes faster on smartcard (probably) Infineon generated them as

$$\left.\begin{aligned} p &\leftarrow k \cdot M + (65537^{a} \bmod M) \\ q &\leftarrow k' \cdot M + (65537^{a'} \bmod M) \end{aligned}\right\} \text{For random } k, k', a, a' \text{ to make } p \text{ and } q \text{ prime}$$

output $N = p * q$

Where M is public constant.

When generating 2048-bit RSA key, Infineon used

$|M| \approx 970$ bits

↳ Each prime is sampled from a very non-uniform distribution over 1024 bit primes

↳ Flawed logic $\left.\begin{aligned} &\geq 2^{64} \text{ choices for } p \\ &\geq 2^{64} \text{ choices for } q \end{aligned}\right\} \geq 2^{128}$ choices for N

↳ OK! Right?

[N.B This alg is faster than real/standard/secure RSA KeyGen]

⇒ Bottom line: Infineon primes are "easy" to factor!

(worst-possible break — recover sk from pk.

Don't have time to sketch the full attack, but will give a simplified version.

Coppersmith showed that if you know ½ of bits of p for $N = pq$, then can factor N in polynomial time.

Using primes with many random bits is not sufficient!

Tool at heart: (Coppersmith, then Howgrave-Graham)

Thm: Let $N = pq$ be an RSA modulus of unknown factorization.
Let $f \in \mathbb{Z}[x]$ have degree δ.
Then can find <u>all</u> solutions x_0 of equation

$$f(x) = 0 \mod p \quad \text{s.t.} \quad |x_0| < N^{1/4\delta}$$

in time $\text{poly}(\delta, \lg N)$.

[This version of the thm comes from nice survey by Alexander May.
Proof uses lattices! LLL...]

Recall that an Infineon prime is

$$p = K \cdot M + (65537^a \bmod M)$$

Strategy
1) Guess a
2) Use Thm to factor N.

Assume for now that we can guess a.
How do we factor?

Know that $p = K \cdot \underbrace{M}_{\text{Known}} + \underbrace{(65537^a \bmod M)}_{\text{Known}}$

$$p = C_1 x + C_0 \quad \leftarrow \text{Want } x$$

$$f(x) = C_1 x + C_0$$

We know that for special value k we seek

(1) $f(k) = p \Rightarrow f(k) \equiv 0 \pmod p$

(2) $\deg(f) = 1$ — f is linear

(3) $|k| < N^{1/4}$

↳ M is $\approx N^{1/3}$ in Infineon case

$p = (k\,M + (\underline{\quad} \bmod M)) \leq N^{1/2}$

$\Rightarrow (k+1)M < N^{1/2}$

$\Rightarrow k \lesssim N^{1/6} < N^{1/4}$

So, if we guess a correctly, can apply thm!

↳ Gives us all solutions in ppt $\Rightarrow$ Can only be poly many

↳ Try all possible k's to factor N.

Back to Step 1: How do we find a?

If $M \approx N^{1/3}$ then there could be $\approx N^{1/3}$ possible values of a! Too many... faster to just factor N directly.

Recall that

$p = kM + (65537^a \bmod M)$

Idea: Look at subgroup of $\mathbb{Z}_M^*$ generated by 65537

$\{65537^0, 65537^1, 65537^2, \ldots\ldots, \}$ (all mod M)

↳ How many distinct elements are there?

↳ It depends! If M is a prime $\Rightarrow$ could be $\approx M$

If M is product of many small primes $\Rightarrow$ Could be very small

Guess which M Infineon used...

If 65537 generates a subgroup of order A mod M, then there are A possible values of a to try.

↳ Extra trick in their paper.... switch M and a to equivalent M' and a' w/o knowing factorization

↳ Even faster attack!