Lecture 10:
Real-World Cryptanalysis

CS 355 - Spring 2019

May 1, 2019

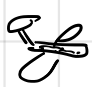
Henry Corrigan-Gibbs

Logistics

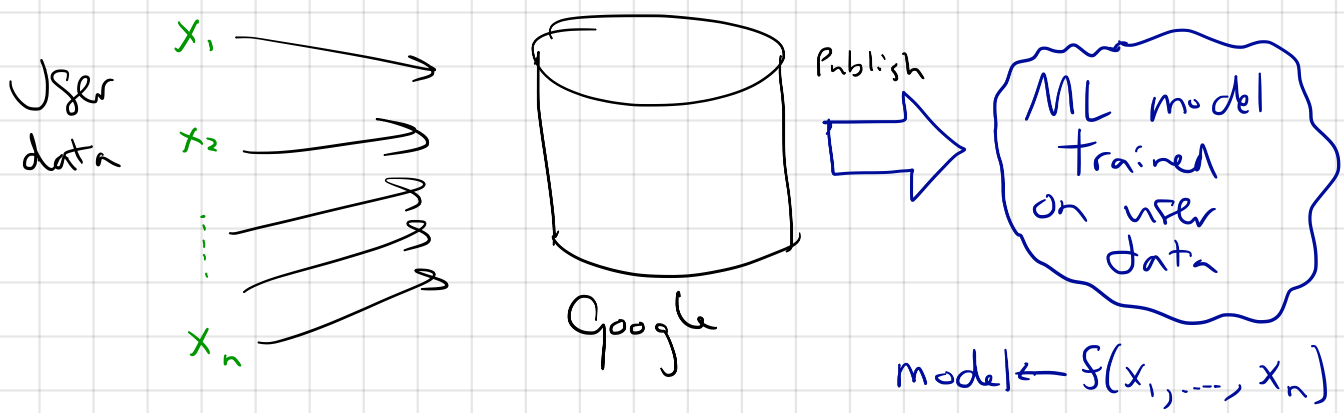
* HW3 out now! Due May 10 at 5pm
* HW2 graded by early next week!
* Bay Area Crypto Day. - May 10 (all day) On campus. Free!
* Ask us about grad school, etc.

See latest results from Stanford, Berkeley, Visa ...

Today

- Recap: Differential privacy
- GCD attack on RSA (2012)
- Infineon Bug (2017)
- Airport! 

Recap: Differential Privacy



Q: Does publishing model violate user privacy?

→ What does this Q even mean formally?

Differential Privacy gives one answer

A: Publishing model is okay as long as training alg satisfies ε-D.P. for some "reasonable" ε.

Q: What ε is reasonable?

Apple ε = 4-8 per day

Intuition: $\left\{ \begin{array}{c} \text{model} \\ \text{trained on} \\ (x_1, \ldots, x_n) \end{array} \right\} \underset{\approx}{\text{close}} \left\{ \begin{array}{c} \text{model} \\ \text{trained on} \\ (x_1, \ldots, x_{n-1}) \end{array} \right\}$

<u>Def'n</u> (ε-DP). Training alg ("mechanism") is ε-DP if $\forall\ D, D' \in \mathcal{X}^n$ s.t. $\Delta(D, D') = 1$, $\forall$ model $\in \mathrm{Im}(\mathrm{Train})$

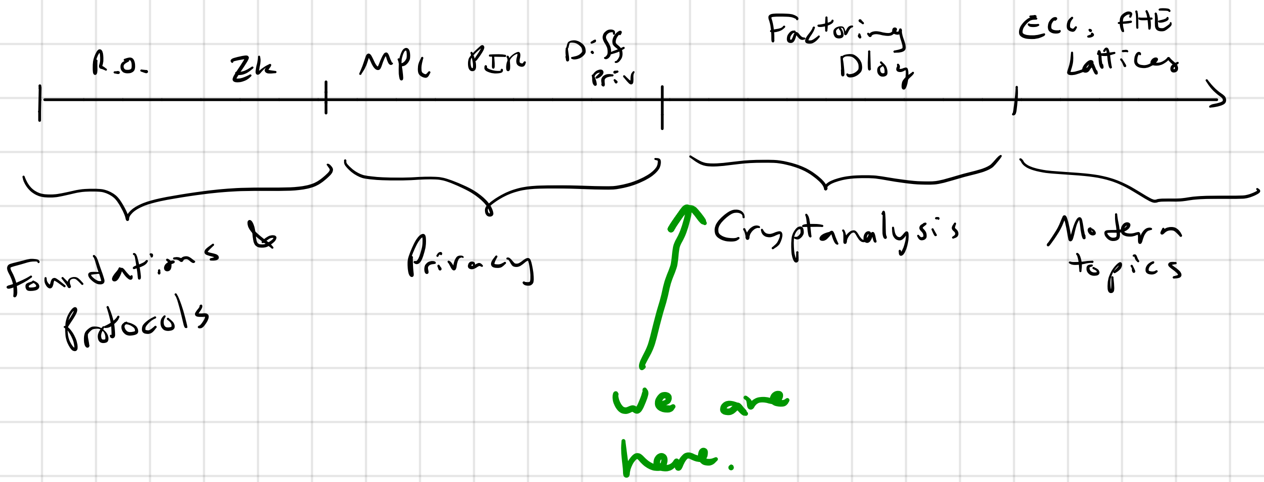
$$\Pr\left[\mathrm{Train}(D) = \text{model}\right] \leq e^{\varepsilon} \cdot \Pr\left[\mathrm{Train}(D') = \text{model}\right].$$

① To be ε-DP, Train(·) must be randomized, if it's non-trivial.

② Intuition: ε-DP ensures "not much additional harm". If publishing model w/o your data caused harm w.p. p, including your data increases harm to prob $e^{\varepsilon} p$.

Real-World Cryptanalysis

So far in this course



Today, we'll talk about two recent developments in real-world cryptanalysis.

↳ Theme: Cryptosystems often break b/c of misuse

- * Poor API design
- * Cross-stack confusion
- * Bugs in implementation
- * Unsafe optimizations
- ⋮

Neither attack we'll see today comes from a "direct" attack on the cryptosystem (e.g. factoring)

↳ Both come from indirect misuse / failure

GCD Attack

As far as I can tell, discovered independently by
Lenstra et al. (2012)
Heninger et al. (2012) ← [Zakir D. (now Stanford CS prof) one of main authors on this paper.]

Background

* RSA used all over the place for encryption & signatures
  SSH, TLS, IPSec, PGP, ...
  (IPSec ← VPNs)

* If you connect to an SSH or TLS host, it will send you its public key

$$pk = (N, e)$$

- e ← encryption exponent. (often $e=3$)
- N ← $N = pq$ for large random primes p, q

Idea: Scan entire IPv4 address space, collect all pks

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

→ With fast net connection, takes 5 mins
(connection ← 10 gigE)

[Useful life fact
$2^{10} \approx 1$ thousand
$2^{20} \approx 1$ million
$2^{30} \approx 1$ billion]

Software to do this is called zmap

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

GCD Attack

What happened? Found many RSA moduli sharing exactly <u>one</u> common factor

$N = p \cdot q$ $\qquad\qquad$ $N' = p \cdot q'$ $\qquad$ [for distinct primes p, q, q']

Both N and N' are hard to factor on their own, but given both, can compute

$$\gcd(N, N') \longrightarrow p \quad \text{in time } \mathrm{poly}(\underbrace{\log N}_{\text{length of modulus } N \text{ in bits}})$$

↳ in fact, $\log^2 N$ time

[Uses Euclid's Algorithm... essentially the oldest known algorithm. See Knuth 4.5.2 for lots of details.]
(300 B.C.E)

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

<u>Problem:</u> Researchers downloaded $\approx 2^{24}$ keys. Computing pairwise gcd among k keys

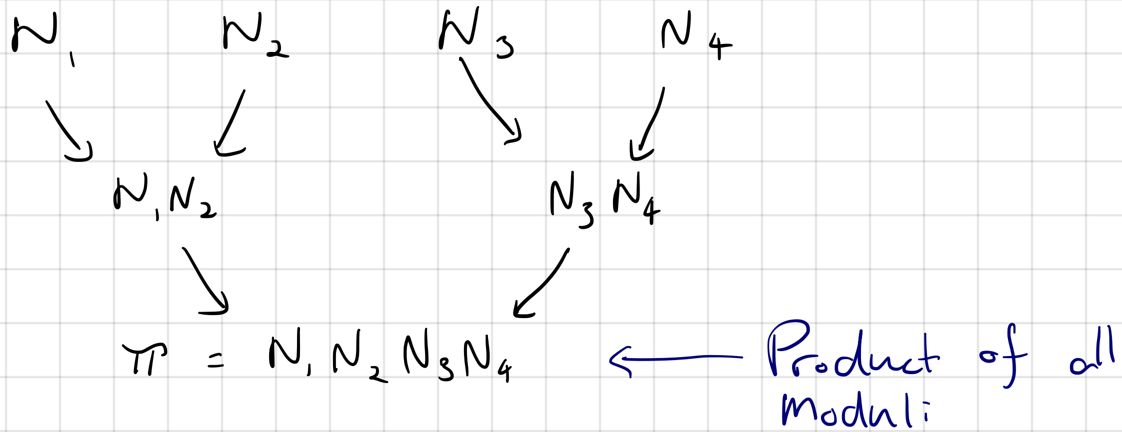
$\Omega(k^2)$ gcd computations $\Rightarrow 2^{50}$ gcds!

↳ This is a feasible – but very large – amount of work. Entire Bitcoin network does $\approx 2^{65}$ hashes per second. But that takes as much energy as a small country ($\approx$ Ireland) uses in a year.

↑ Not in reach for an academic group...

Better idea: GCD Tree (Bernstein)

Compute all pairwise gcds at once



Runs in time $\tilde{O}(k \lg N)$ ← k input keys

Compute $r = \pi \bmod N_1^2$ then $\gcd(r/N_1, N_1)$ yields common factor of N_1 with some other modulus

$$\pi = N_1 N_2 N_3 N_4$$
$$r = p^2 q q' \underbrace{N_3 N_4}_{} \mod N_1^2$$

$$r = p^2 q q' (\text{junk}) \mod N_1^2$$
$$r/N_1 = p q' (\text{junk}) \mod N_1^2$$

$$\gcd(pq'(\text{junk}), pq) = p$$

Need to compute
$\pi \bmod N_1^2$
$\pi \bmod N_2^2$
$\pi \bmod N_3^2$
⋮
} Use another tree!

Then compute remainders

$$N_1 N_2 N_3 N_4$$

$$(\bmod N_1^2 N_2^2) \qquad (\bmod N_3^2 N_4^2)$$

$$(\bmod N_1^2) \qquad (\bmod N_2^2) \qquad (\bmod N_3^2) \qquad (\bmod N_4^2)$$

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

→ Found lots of factors!

How did RSA moduli end up sharing prime factors?

(Chance of two people picking same prime is tiny)

I. Very bad implementation (e.g. IBM mgmt interfaces)

```
KeyGen() {
    // Hardcoded values - 9 primes
    P = {p_1, ..., p_9}

    chose p_i, p_j from P
    output N <- p_i p_j as modulus
}
```

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

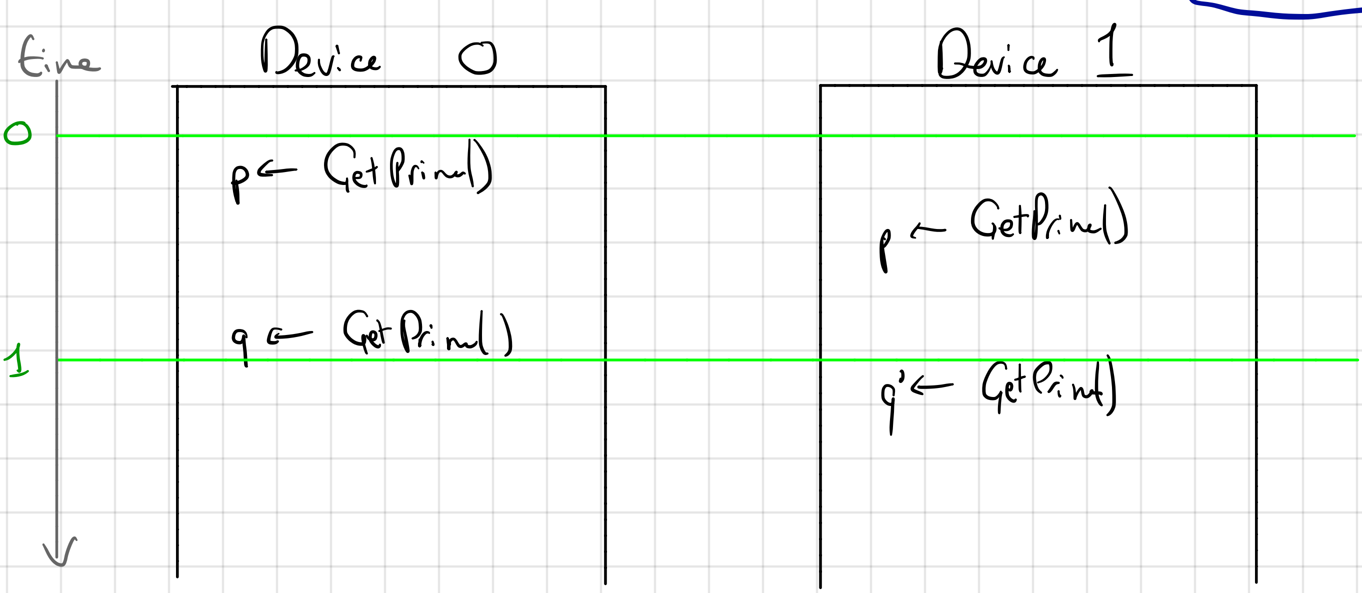
II. Unfortunate confluence of unlucky events.

* Many embedded devices (e.g. net routers) speak TLS/SSH
* Linux gathers "random" values from I/O devices
  ↳ keyboard, mouse, HD, ...
* But, embedded devices have no keyboard, no mouse, no HD, etc.
* On first boot, two devices of the same model have the same state
* On first boot, these devices gen RSA keys (e.g. for SSH server)

  ↳ Two devices can end up w/ same key!

KeyGen()
  p ← GetPrime()
  q ← GetPrime() — { x ← Hash(state, time()); return smallest prime > x }
  output N = p · q as pub key

Q: How would you fix this?



Infineon Attack (2017)

- One of the most shocking cryptographic attacks in recent memory.
  ↳ Tens of millions of smartcards recalled
- Paper is amazing. Really impressive piece of work
  (linked online)

Background

* RSA is surprisingly fragile
* Many many variants of RSA are insecure ← See HW!
* Optimizations can easily break security.

Standard RSA KeyGen

$p, q \xleftarrow{R} \{\lambda\text{-bit primes}\}$ for $\lambda \approx 1024$

output $N \leftarrow p \cdot q$

Infineon smartcards used an "optimized" Keygen:

$p \leftarrow k \cdot M + (65537^{a} \bmod M)$
$q \leftarrow k' M + (65537^{a'} \bmod M)$
output $N \leftarrow p \cdot q$

} For random k, k', a, a' to make p, q prime

M is public constant ≈ 970 bits

↳ Each prime is sampled from a funny distribution (not uniform)

Flawed logic: $\geq 2^{128}$ choices for p, $\geq 2^{128}$ " " q } $\geq 2^{256}$ choices for N

↳ Ok! Right? Wrong!

⇝ Infineon primes are easy to factor.

Don't have time for the full attack... will give a simplified version.

Coppersmith: If you know 1/2 of the bits of p for $N = pq$, then can factor N in poly time.

Using primes w/ many random bits is not sufficient!

Core Cryptanalytic Tool (Coppersmith Howegrave-Graham):

> **Thm** Let $N = pq$ be an RSA modulus of unknown factorization. Let $f \in \mathbb{Z}[x]$ have degree δ. Then can find all solns x_0 of
>
> $$f(x) = 0 \mod p \quad \text{s.t.} \quad |x_0| \leq N^{1/4\delta}$$
>
> in time poly$(\log N, \delta)$

This version of thm comes from nice survey by Alexander May. Proof uses lattices, LLL.

Attack Strategy

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

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

Know that $p = \underbrace{kM}_{\text{known}} + \underbrace{(65537^a \mod M)}_{\text{known}}$

$$p = C_1 x + C_0$$

We want x_0!

We know that for the special value k we seek...

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

(2) $\deg(f) = 1$ – f is linear

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

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

$p = kM + (\bmod M) \leq N^{1/2}$

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

$\Rightarrow k \leq N^{1/6} \leq N^{1/4}$.

So, if we guess "a" correctly, we can apply the theorem to recover k. $\Rightarrow$ factor.

↳ Gives all solutions in poly time $\Rightarrow$ Can only be poly many.

↳ Try all possible k's to factor N.

Back to attack step 1: How do we guess "a"?

If $M \approx N^{1/3}$ then there could be $\approx N^{1/3}$ possible values of a! Too many to guess... 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 prime ⇒ Could be ≈ M

If M is product of many small primes ⇒ Could be very small

Guess which M Infineon used...

If 65537 generates order-A subgroup in $\mathbb{Z}_M^*$, then there are A values of "a" to try.

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

↳ Even faster attack!

Be wary of optimizations. If you can't prove that opt. is safe, maybe it's not...