RSA Algorithm

Alice

(n, e)

plaintext message:
$m \pmod{n}$

Encryption:

$c \equiv m^e \pmod{n}$

c

Bob

Key Generation:

choose secret primes p, q
choose secret d invertible
mod $\varphi(pq) = (p-1)(q-1)$
and its inverse e.

(n, e)

Public Key: $(n = pq, e)$ "encryption exponent"
Private Key: p, q, d "decryption exponent"

Decryption:

$c^d \pmod{n}$
$\equiv m^{ed} \equiv m^1 \equiv m$

Collision/Birthday Attack

① List 1:

$$cx^{-e} \pmod{n} \quad \text{various } x$$

② List 2:

$$y^e \pmod{n} \quad \text{various } y$$

Look for a collision:

$$cx^{-e} \equiv y^e \pmod{n}$$

$$c \equiv \underbrace{(xy)}_{m}{}^e \pmod{n}$$

Ⓐ Random x, y

$\rightarrow O(\sqrt{n})$ attack

Ⓑ Let $1 < x < \sqrt{B}$
$1 < y < \sqrt{B}$

Then if $m < B$,
likely(?) $m = xy$, x, y in that range.

Lesson: don't send <u>small</u> messages.

Fix: pad the messages with random digits

Timing Attacks

secret

Sps you could watch power/time used for Bob's decryptions. ($c^d \pmod{n}$ for various c)

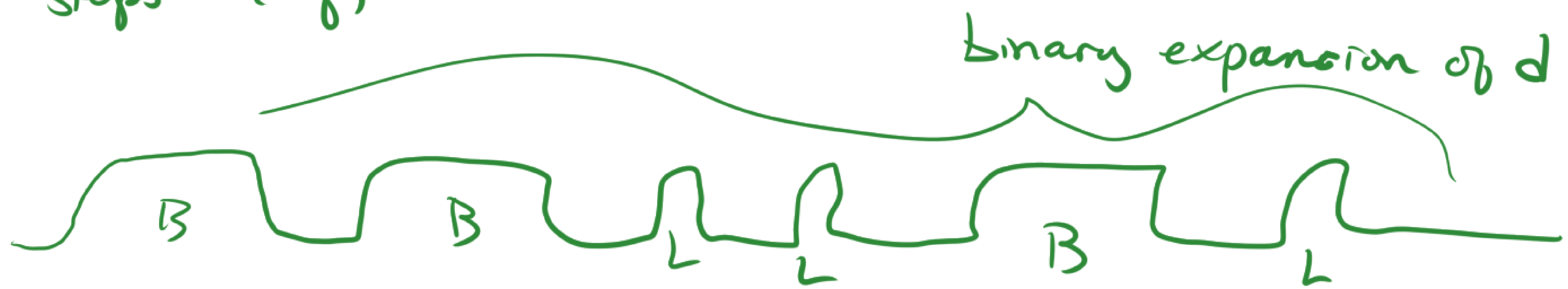
know

Bob does exponentiation by double-n-add.

→ big steps (sq-and-mult.)

→ little steps (sq)

power signature:



What if you just have overall timing?

1. lower overall time = lower hamming weight ← # of 1's in binary expansion of d.

2. use variances

Variances for a timing attack.

<u>Input:</u> ① collect overall times t for each ciphertext c. (Bob's time to decrypt.)

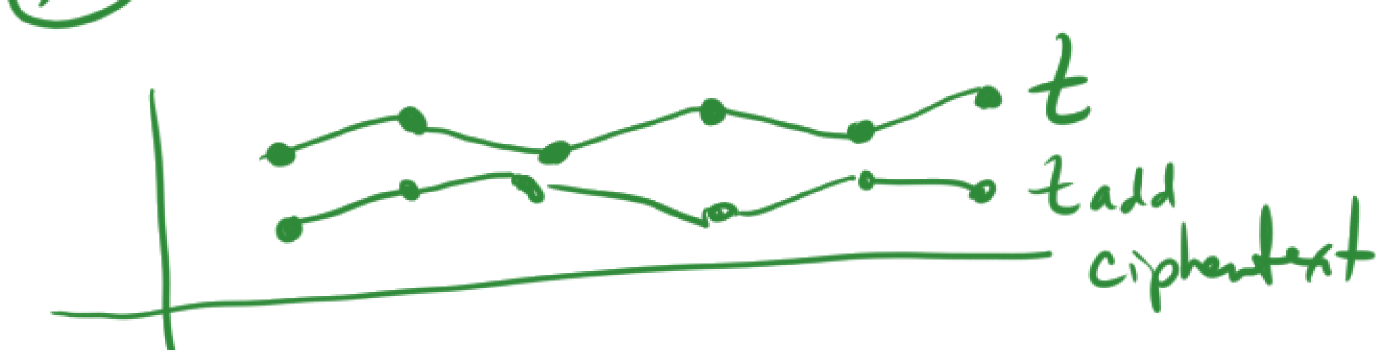
② for the same c, run experiments to get t_{add}, time to do the 1st addition (double-n-add).

(don't know if Bob includes this)

<u>Key:</u> Compare variance $(\overbrace{t - t_{add}})$ and variance (t)

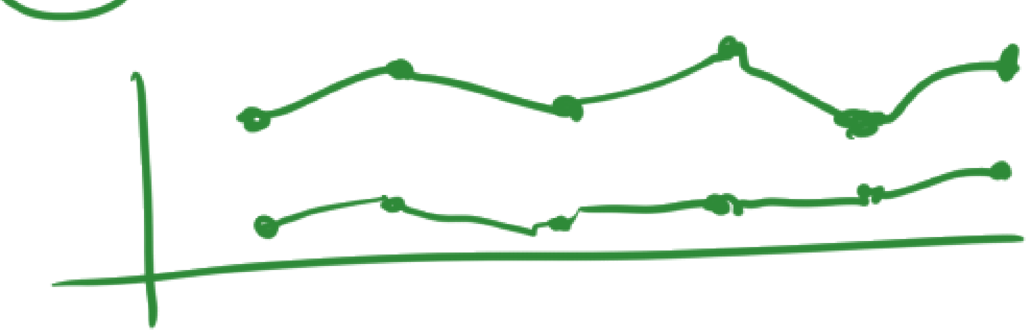
<u>Analysis:</u>

Ⓐ If t_{add} <u>not</u> included in t then t_{add} & t are independent.



$$\mathrm{var}(t - t_{add}) \approx \mathrm{var}(t) + \mathrm{var}(t_{add}) > \mathrm{var}(t).$$

Ⓑ If t_{add} <u>is</u> included in t then $t - t_{add}$ & t_{add} are independent.



$$\mathrm{var}(t) \approx \mathrm{var}(t - t_{add}) + \mathrm{var}(t_{add}) > \mathrm{var}(t - t_{add}).$$

$\Longrightarrow$ Guess the 1st digit of d. (Repeat for more digits.)

Those are 2 cautionary tales on implementation.

Factoring!

p−1 Factoring.

Pick a (say $a=2$).

Compute a chain (mod n)

$$a \xrightarrow{\text{sq}} a^2 \xrightarrow{\text{cube}} a^{3\cdot 2} \xrightarrow[\text{pow}]{\text{4th}} a^{4\cdot 3\cdot 2} \longrightarrow \cdots \longrightarrow a^{B!} =: b$$

Try $d = \gcd(b-1, n)$. (Hope it is a proper factor of n).

Why is there a good chance?

If $p \mid n$ and $p-1$ = product of small primes ("smooth").

then $p-1 \mid B!$ (probably)

Fermat's little theorem
$a^{p-1} \equiv 1 \pmod{p}$

So $b = a^{B!} = a^{(p-1)\cdot k} \equiv 1^k \equiv 1 \pmod{p}$. $\Longrightarrow$ $p \mid b-1$.

So $p \mid d$ and probably $n \nmid d$.

Lesson: RSA $n = pq$ is vulnerable if $p-1$ or $q-1$ is smooth.

Fix: Find p, q by taking $p = kp_0 + 1$ and primality testing, for various k, big prime p_0.