The RSA cryptosystem and its pitfalls

Marco Vitturi

School of Mathematics

PG Colloquium - September 24th 2014

The RSA cryptosystem and its pitfalls

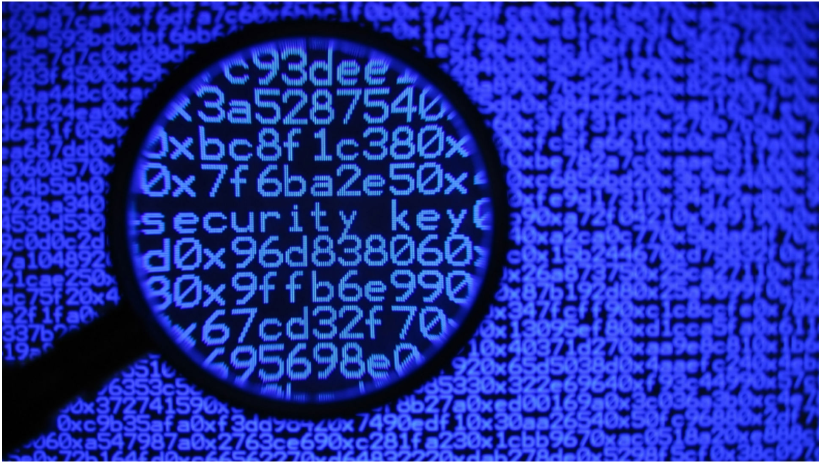

Figure: Pointless stock photo returned by a Google Images query for the word "cryptography"

Private Key vs. Public key

Private Key: both parties know the shared secret key (same for both), and they use to encrypt and decrypt

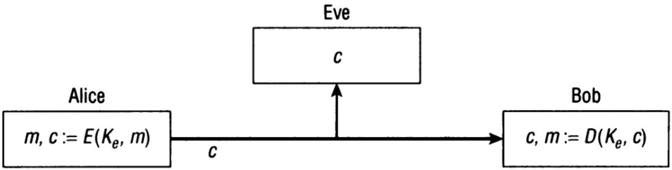

Figure 2.2: Generic setting for encryption

Figure: Ferguson, Schneier, Kohno ©

Communication is only secret as long as the password stays secret. The parties have to exchange the key safely at an earlier time.

Private Key vs. Public key

Public Key: Bob's *public key* is known to everyone (you can find it on a phonebook, so to say), and allows encryption of messages to Bob

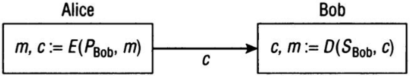

Figure 2.5: Generic setting for public-key encryption

Figure: Ferguson, Schneier, Kohno ©

Decryption requires a *different, private key*, known to Bob alone. No exchange of keys is required.

Fermat's Little Theorem

Fermat's Little Theorem

Let N be a natural number, and $\phi(N) := \#\{n < N \text{ s.t. } \gcd(n, N) = 1\}$ the Euler's totient function. Then, for every a coprime with N, it is

$$a^{\phi(N)} \equiv 1 \mod N.$$

Fermat's Little Theorem

Fermat's Little Theorem

Let N be a natural number, and $\phi(N) := \#\{n < N \text{ s.t. } \gcd(n, N) = 1\}$ the Euler's totient function. Then, for every a coprime with N, it is

$$a^{\phi(N)} \equiv 1 \mod N.$$

It's easy to calculate $\phi(N)$ from the prime factorization $N = p_1^{\alpha_1} \cdots p_r^{\alpha_r}$:

$$\phi(N) = N \left(\frac{p_1 - 1}{p_1}\right) \cdots \left(\frac{p_r - 1}{p_r}\right)$$

Fermat's Little Theorem

Fermat's Little Theorem

Let N be a natural number, and $\phi(N) := \#\{n < N \text{ s.t. } \gcd(n, N) = 1\}$ the Euler's totient function. Then, for every a coprime with N, it is

$$a^{\phi(N)} \equiv 1 \mod N.$$

It's easy to calculate $\phi(N)$ from the prime factorization $N = p_1^{\alpha_1} \cdots p_r^{\alpha_r}$:

$$\phi(N) = N\left(\frac{p_1 - 1}{p_1}\right) \cdots \left(\frac{p_r - 1}{p_r}\right)$$

$\Rightarrow$ if $N = pq$ then $\phi(N) = (p-1)(q-1)$.

RSA algorithm

- Pick primes p, q and form $N = pq$ (2048 or 4096 bits)

RSA algorithm

- Pick primes p, q and form $N = pq$ (2048 or 4096 bits)
- Pick e and d s.t. $e \cdot d \equiv 1 \mod (p-1)(q-1)$

RSA algorithm

- Pick primes p, q and form $N = pq$ (2048 or 4096 bits)
- Pick e and d s.t. $e \cdot d \equiv 1 \mod (p-1)(q-1)$
- **Public key** $= (N, e)$

RSA algorithm

- Pick primes p, q and form $N = pq$ (2048 or 4096 bits)
- Pick e and d s.t. $e \cdot d \equiv 1 \mod (p-1)(q-1)$
- **Public key** $= (N, e)$
  Private key $= (N, d)$

RSA algorithm

- Pick primes p, q and form $N = pq$ (2048 or 4096 bits)
- Pick e and d s.t. $e \cdot d \equiv 1 \mod (p-1)(q-1)$
- **Public key** $= (N, e)$
  Private key $= (N, d)$
- **Encryption**: message m is encrypted by Alice to

$$c := m^e \mod N$$

RSA algorithm

- Pick primes p, q and form $N = pq$ (2048 or 4096 bits)
- Pick e and d s.t. $e \cdot d \equiv 1 \mod (p-1)(q-1)$
- **Public key** $= (N, e)$
  Private key $= (N, d)$
- **Encryption**: message m is encrypted by Alice to

$$c := m^e \mod N$$

- **Decryption**: ciphertext c is decrypted by Bob by performing

$$c^d \mod N,$$

thus recovering the message m

RSA algorithm: the math

- Since e and d are inverses $\mod (p-1)(q-1) = \phi(N)$, one has

$$ed = k\phi(N) + 1$$

RSA algorithm: the math

- Since e and d are inverses $\mod (p-1)(q-1) = \phi(N)$, one has

$$ed = k\phi(N) + 1$$

- by Fermat's Little theorem

$$c^d \equiv m^{ed} \equiv m^{k\phi(N)+1} \equiv 1 \cdot m = m \mod N$$

RSA algorithm: the math

- Since e and d are inverses $\mod (p-1)(q-1) = \phi(N)$, one has

$$ed = k\phi(N) + 1$$

- by Fermat's Little theorem

$$c^d \equiv m^{ed} \equiv m^{k\phi(N)+1} \equiv 1 \cdot m = m \mod N$$

Efficiency

Exponentiations $\mod N$ are efficient: $x^n \mod N$ only requires $O(\log n)$ exponentiations altogether, by repeated squaring.

Strength of RSA: decryption strategies

By factoring N you get p, q, from which you calculate $\phi(N)$ and then calculate $d = e^{-1} \mod \phi(N)$ (by Euclid's algorithm)

▷ but *integer factorization is a hard problem*

Strength of RSA: decryption strategies

By factoring N you get p, q, from which you calculate $\phi(N)$ and then calculate $d = e^{-1} \mod \phi(N)$ (by Euclid's algorithm)
▷ but *integer factorization is a hard problem*

Another possibility is to extract the discrete e-th root $\mod N$ of c, thus recovering m directly. This is believed to be just as hard as factoring, at least.

Strength of RSA: decryption strategies

By factoring N you get p, q, from which you calculate $\phi(N)$ and then calculate $d = e^{-1} \mod \phi(N)$ (by Euclid's algorithm)
▷ but *integer factorization is a hard problem*

Another possibility is to extract the discrete e-th root $\mod N$ of c, thus recovering m directly. This is believed to be just as hard as factoring, at least.

Still, these aren't the only attacks. RSA can easily be broken if used naively.

There are plenty of non-mathematical attacks on RSA.

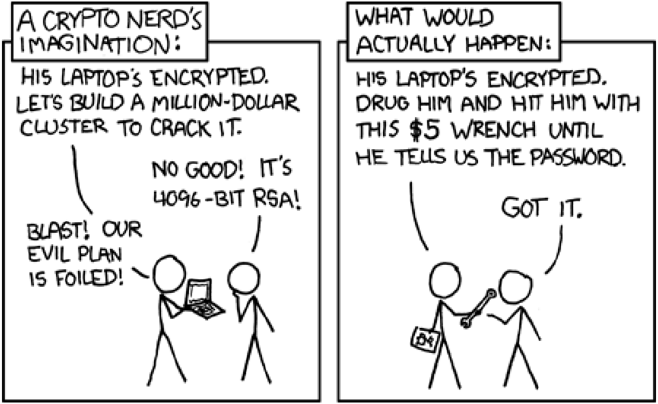

Figure: ©Xkcd

There are plenty of non-mathematical attacks on RSA.

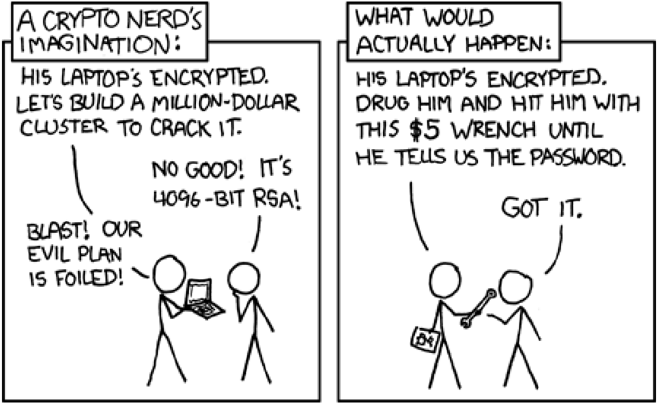

Figure: ©Xkcd

We concentrate on mathematical attacks, though. Specifically, those that do not attempt to solve the factorization problem.

Common N

N should be unique for every user.

Common N

N **should be unique for every user.**

This is because, given e, d it is easy to factor N, and thus find $\phi(N)$ and calculate other users' secret keys.

Common N

N **should be unique for every user.**
This is because, given e, d it is easy to factor N, and thus find $\phi(N)$ and calculate other users' secret keys.

We know $ed - 1 = k\phi(N) =: M$, then for a coprime with N

$$a^M \equiv 1 \mod N.$$

Common N

N should be unique for every user.
This is because, given e, d it is easy to factor N, and thus find $\phi(N)$ and calculate other users' secret keys.

We know $ed - 1 = k\phi(N) =: M$, then for a coprime with N

$$a^M \equiv 1 \mod N.$$

$\phi(N)$ is even, thus $M = 2^n t$, and therefore $a^{M/2}$ is a square root of $1 \mod N$. There are 4 such square roots, $\pm 1, \pm x$, where

$$\begin{cases} x \equiv 1 \mod p, \\ x \equiv -1 \mod q. \end{cases}$$

Common N

N **should be unique for every user.**
This is because, given e, d it is easy to factor N, and thus find $\phi(N)$ and calculate other users' secret keys.

We know $ed - 1 = k\phi(N) =: M$, then for a coprime with N

$$a^M \equiv 1 \mod N.$$

$\phi(N)$ is even, thus $M = 2^n t$, and therefore $a^{M/2}$ is a square root of $1 \mod N$. There are 4 such square roots, $\pm 1, \pm x$, where

$$\begin{cases} x \equiv 1 \mod p, \\ x \equiv -1 \mod q. \end{cases}$$

For a chosen at random, with high probability one amongst $a^{M/2}, a^{M/4}, \ldots$ is $\pm x$, and $\gcd(x \pm 1, N)$ gives a factor of N.

Small private key d

If d is too small, we run into troubles: d can be recovered

Theorem 1 (M. J. Wiener).

Let $N = pq$, $q < p < 2q$, *and suppose* $d < \frac{1}{3}N^{1/4}$. *Then, given public key* (N, e), *one can recover* d *in* $O(\log N)$ *time.*

Small private key d

If d is too small, we run into troubles: d can be recovered

Theorem 1 (M. J. Wiener).

Let $N = pq$, $q < p < 2q$, *and suppose* $d < \frac{1}{3}N^{1/4}$. *Then, given public key* (N, e), *one can recover* d *in* $O(\log N)$ *time.*

By comparison, factoring N takes

$$O\left((\log N)^{c_0\left(\frac{\log N}{\log\log N}\right)^{1/3}}\right)$$

with the fastest known algorithm.

Small private key d

If d is too small, we run into troubles: d can be recovered

Theorem 1 (M. J. Wiener).

Let $N = pq$, $q < p < 2q$, *and suppose* $d < \frac{1}{3}N^{1/4}$. *Then, given public key* (N, e), *one can recover* d *in* $O(\log N)$ *time.*

By comparison, factoring N takes

$$O\left((\log N)^{c_0\left(\frac{\log N}{\log\log N}\right)^{1/3}}\right)$$

with the fastest known algorithm.

▷ The vulnerability here is the existence of good approximations to rationals with *continued fractions*

Small private key d: proof

- Since $ed = k\phi(N) + 1$, we have $\left|\frac{e}{\phi(N)} - \frac{k}{d}\right| = \frac{1}{d\phi(N)}$.

Small private key d: proof

- Since $ed = k\phi(N) + 1$, we have $\left|\frac{e}{\phi(N)} - \frac{k}{d}\right| = \frac{1}{d\phi(N)}$.
- Given the magnitude, $N^{-1} \approx \phi(N)^{-1}$

Small private key d: proof

- Since $ed = k\phi(N) + 1$, we have $\left|\frac{e}{\phi(N)} - \frac{k}{d}\right| = \frac{1}{d\phi(N)}$.
- Given the magnitude, $N^{-1} \approx \phi(N)^{-1}$
- Consider then e/N as an approximation instead:

$$\left|\frac{e}{N} - \frac{k}{d}\right| = \left|\frac{ed}{Nd} - \frac{Nk}{Nd} + \frac{k\phi(N)}{Nd} - \frac{k\phi(N)}{Nd}\right|$$

Small private key d: proof

- Since $ed = k\phi(N) + 1$, we have $\left|\frac{e}{\phi(N)} - \frac{k}{d}\right| = \frac{1}{d\phi(N)}$.
- Given the magnitude, $N^{-1} \approx \phi(N)^{-1}$
- Consider then e/N as an approximation instead:

$$\left|\frac{e}{N} - \frac{k}{d}\right| = \left|\frac{ed}{Nd} - \frac{Nk}{Nd} + \frac{k\phi(N)}{Nd} - \frac{k\phi(N)}{Nd}\right|$$

$$= \left|\frac{1}{Nd} - k\frac{N - \phi(N)}{Nd}\right|$$

Small private key d: proof

- Since $ed = k\phi(N) + 1$, we have $\left|\frac{e}{\phi(N)} - \frac{k}{d}\right| = \frac{1}{d\phi(N)}$.
- Given the magnitude, $N^{-1} \approx \phi(N)^{-1}$
- Consider then e/N as an approximation instead:

$$\left|\frac{e}{N} - \frac{k}{d}\right| = \left|\frac{ed}{Nd} - \frac{Nk}{Nd} + \frac{k\phi(N)}{Nd} - \frac{k\phi(N)}{Nd}\right|$$

$$= \left|\frac{1}{Nd} - k\frac{N - \phi(N)}{Nd}\right|$$

- but $\phi(N) = (p-1)(q-1) = N - p - q + 1$

Small private key d: proof

- Since $ed = k\phi(N) + 1$, we have $\left|\frac{e}{\phi(N)} - \frac{k}{d}\right| = \frac{1}{d\phi(N)}$.
- Given the magnitude, $N^{-1} \approx \phi(N)^{-1}$
- Consider then e/N as an approximation instead:

$$\left|\frac{e}{N} - \frac{k}{d}\right| = \left|\frac{ed}{Nd} - \frac{Nk}{Nd} + \frac{k\phi(N)}{Nd} - \frac{k\phi(N)}{Nd}\right|$$

$$= \left|\frac{1}{Nd} - k\frac{N - \phi(N)}{Nd}\right|$$

- but $\phi(N) = (p-1)(q-1) = N - p - q + 1$
- then we can bound $N - \phi(N) = p + q - 1 < 3N^{1/2}$

Small private key d: proof

- Thus we have

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{3kN^{1/2}}{Nd} = \frac{3k}{N^{1/2}d}$$

Small private key d: proof

- Thus we have

$$\left|\frac{e}{N}-\frac{k}{d}\right| < \frac{3kN^{1/2}}{Nd} = \frac{3k}{N^{1/2}d}$$

- Since $k\phi(N) < ed$, and $e < \phi(N)$, it must be $k < d < \frac{1}{3}N^{1/4}$

Small private key d: proof

- Thus we have

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{3kN^{1/2}}{Nd} = \frac{3k}{N^{1/2}d}$$

- Since $k\phi(N) < ed$, and $e < \phi(N)$, it must be $k < d < \frac{1}{3}N^{1/4}$
- $\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{1}{N^{1/4}d} < \frac{1}{2d^2}$

Small private key d: proof

- Thus we have

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{3kN^{1/2}}{Nd} = \frac{3k}{N^{1/2}d}$$

- Since $k\phi(N) < ed$, and $e < \phi(N)$, it must be $k < d < \frac{1}{3}N^{1/4}$
- $\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{1}{N^{1/4}d} < \frac{1}{2d^2}$
- (from the theory of continued fractions) there are at most $\log_2 N$ fractions with denominator $D < N$ that approximate e/N within error $1/2D^2$.

Small private key d: proof

- Thus we have

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{3kN^{1/2}}{Nd} = \frac{3k}{N^{1/2}d}$$

- Since $k\phi(N) < ed$, and $e < \phi(N)$, it must be $k < d < \frac{1}{3}N^{1/4}$
- $\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{1}{N^{1/4}d} < \frac{1}{2d^2}$
- (from the theory of continued fractions) there are at most $\log_2 N$ fractions with denominator $D < N$ that approximate e/N within error $1/2D^2$.
- they are the convergents of the continued fraction expansion of e/N; one of them will be k/d.

Small public key e

A small public key e isn't good either: it allows partial breacking

Small public key e

A small public key e isn't good either: it allows partial breacking

Theorem 2 (Coppersmith).

Let $N \in \mathbb{N}$ *and* $P \in \mathbb{Z}[X]$ *be a monic polynomial of degree* d. *Fix* $1/d > \varepsilon > 0$.
Then the roots z_0 *of* $P(X) \mod N$ *s.t.* $|z_0| < N^{1/d-\varepsilon}$ *can be calculated in* $O(\min\left(\frac{1}{\varepsilon}, \log N\right)^2)$ *time.*

Small public key e

A small public key e isn't good either: it allows partial breacking

Theorem 2 (Coppersmith).

Let $N \in \mathbb{N}$ *and* $P \in \mathbb{Z}[X]$ *be a monic polynomial of degree* d. *Fix* $1/d > \varepsilon > 0$.
Then the roots z_0 *of* $P(X) \mod N$ *s.t.* $|z_0| < N^{1/d-\varepsilon}$ *can be calculated in* $O(\min\left(\frac{1}{\varepsilon}, \log N\right)^2)$ *time.*

- In our case N is part of the public key and $P(X) = X^e - c$.

Small public key e

A small public key e isn't good either: it allows partial breacking

Theorem 2 (Coppersmith).

Let $N \in \mathbb{N}$ *and* $P \in \mathbb{Z}[X]$ *be a monic polynomial of degree* d. *Fix* $1/d > \varepsilon > 0$.
Then the roots z_0 *of* $P(X) \mod N$ *s.t.* $|z_0| < N^{1/d-\varepsilon}$ *can be calculated in* $O(\min\left(\frac{1}{\varepsilon}, \log N\right)^2)$ *time.*

- In our case N is part of the public key and $P(X) = X^e - c$.
- Allows broadcasting attacks (a message m broadcasted to a high number of users can be decoded by a non-recipient)

Small public key e: Coppersmith's theorem

Sketch of the idea: for $P(X) = \sum a_i X^i \in \mathbb{Z}[X]$ define norm $\|P\|^2 = \sum |a_i|^2$

Small public key e: Coppersmith's theorem

Sketch of the idea: for $P(X) = \sum a_i X^i \in \mathbb{Z}[X]$ define norm $\|P\|^2 = \sum |a_i|^2$

- if $\|P\|$ is small, then small roots of P mod N are real roots:

Small public key e: Coppersmith's theorem

Sketch of the idea: for $P(X) = \sum a_i X^i \in \mathbb{Z}[X]$ define norm $\|P\|^2 = \sum |a_i|^2$

- if $\|P\|$ is small, then small roots of P mod N are real roots: assume $P(z_0) \equiv 0 \mod N$ and $|z_0| < R$; if $\|P(R\cdot)\| < N(\deg P)^{-1/2}$, then $P(z_0) = 0$.

Small public key e: Coppersmith's theorem

Sketch of the idea: for $P(X) = \sum a_i X^i \in \mathbb{Z}[X]$ define norm $\|P\|^2 = \sum |a_i|^2$

- if $\|P\|$ is small, then small roots of P mod N are real roots: assume $P(z_0) \equiv 0 \mod N$ and $|z_0| < R$; if $\|P(R\cdot)\| < N(\deg P)^{-1/2}$, then $P(z_0) = 0$.
- real roots can be approximated efficiently (e.g. Newton's method)

Small public key e: Coppersmith's theorem

Sketch of the idea: for $P(X) = \sum a_i X^i \in \mathbb{Z}[X]$ define norm $\|P\|^2 = \sum |a_i|^2$

- if $\|P\|$ is small, then small roots of P mod N are real roots: assume $P(z_0) \equiv 0 \mod N$ and $|z_0| < R$; if $\|P(R\cdot)\| < N(\deg P)^{-1/2}$, then $P(z_0) = 0$.
- real roots can be approximated efficiently (e.g. Newton's method)
- need to modify P to reduce its norm, while preserving the roots:

Small public key e: Coppersmith's theorem

Sketch of the idea: for $P(X) = \sum a_i X^i \in \mathbb{Z}[X]$ define norm $\|P\|^2 = \sum |a_i|^2$

- if $\|P\|$ is small, then small roots of P mod N are real roots: assume $P(z_0) \equiv 0 \mod N$ and $|z_0| < R$; if $\|P(R\cdot)\| < N(\deg P)^{-1/2}$, then $P(z_0) = 0$.
- real roots can be approximated efficiently (e.g. Newton's method)
- need to modify P to reduce its norm, while preserving the roots: e.g. $Q(X)P(X)$ and linear combinations.

Small public key e: Coppersmith's theorem

Sketch of the idea: for $P(X) = \sum a_i X^i \in \mathbb{Z}[X]$ define norm $\|P\|^2 = \sum |a_i|^2$

- if $\|P\|$ is small, then small roots of P mod N are real roots: assume $P(z_0) \equiv 0 \mod N$ and $|z_0| < R$; if $\|P(R\cdot)\| < N(\deg P)^{-1/2}$, then $P(z_0) = 0$.
- real roots can be approximated efficiently (e.g. Newton's method)
- need to modify P to reduce its norm, while preserving the roots: e.g. $Q(X)P(X)$ and linear combinations.
- Since $P(z_0)^\ell \equiv 0 \mod N^\ell$, consider $\tilde{P}$ in basis $g_{j,k}(X) = N^{\ell-k} X^j P(X)^k$, then you need $\|\tilde{P}(N^{1/d-\varepsilon}\cdot)\| < N^\ell$

Small public key e: Coppersmith's theorem

Sketch of the idea: for $P(X) = \sum a_i X^i \in \mathbb{Z}[X]$ define norm $\|P\|^2 = \sum |a_i|^2$

- if $\|P\|$ is small, then small roots of P mod N are real roots: assume $P(z_0) \equiv 0 \mod N$ and $|z_0| < R$; if $\|P(R\cdot)\| < N(\deg P)^{-1/2}$, then $P(z_0) = 0$.
- real roots can be approximated efficiently (e.g. Newton's method)
- need to modify P to reduce its norm, while preserving the roots: e.g. $Q(X)P(X)$ and linear combinations.
- Since $P(z_0)^\ell \equiv 0 \mod N^\ell$, consider $\tilde{P}$ in basis $g_{j,k}(X) = N^{\ell-k} X^j P(X)^k$, then you need $\|\tilde{P}(N^{1/d-\varepsilon}\cdot)\| < N^\ell$
- for ℓ big enough you can do this; can be calculated efficiently using the LLL Algorithm

Small public key e: key exposure

A small key e can expose up to 50% of the most significant bits of d

Small public key e: key exposure

A small key e can expose up to 50% of the most significant bits of d

As before,

$$ed = k\phi(N) - 1 \Rightarrow d \approx \frac{kN}{e}, \quad k \leqslant e$$

Small public key e: key exposure

A small key e can expose up to 50% of the most significant bits of d

As before,

$$ed = k\phi(N) - 1 \Rightarrow d \approx \frac{kN}{e}, \quad k \leqslant e$$

Indeed,

$$\left|\frac{kN}{e} - d\right| < \left|\frac{k(p+q)}{e}\right| < \frac{3kN^{1/2}}{e} \leqslant 3N^{1/2}$$

Small public key e: key exposure

A small key e can expose up to 50% of the most significant bits of d

As before,

$$ed = k\phi(N) - 1 \Rightarrow d \approx \frac{kN}{e}, \quad k \leqslant e$$

Indeed,

$$\left|\frac{kN}{e} - d\right| < \left|\frac{k(p+q)}{e}\right| < \frac{3kN^{1/2}}{e} \leqslant 3N^{1/2}$$

$\triangleright$ for some k, half the most significant bits of $\frac{kN}{e}$ coincide with d. There are at most e possibilities for k.

Small public key e: key exposure

A small key e can expose up to 50% of the most significant bits of d

As before,

$$ed = k\phi(N) - 1 \Rightarrow d \approx \frac{kN}{e}, \quad k \leqslant e$$

Indeed,

$$\left|\frac{kN}{e} - d\right| < \left|\frac{k(p+q)}{e}\right| < \frac{3kN^{1/2}}{e} \leqslant 3N^{1/2}$$

▷ for some k, half the most significant bits of $\frac{kN}{e}$ coincide with d.
There are at most e possibilities for k.
Information leakage: $\approx \frac{\log N}{2} - \log e$ bits.

p and q too close

If p are q are too close, then factoring N is easy

p and q too close

If p are q are too close, then factoring N is easy

Theorem 3 (Fermat's factorization method).

If $|p-q| < cN^{1/4}$, *then* N *can be factored in* $O_c(1)$ *time.*

p and q too close

If p are q are too close, then factoring N is easy

Theorem 3 (Fermat's factorization method).

If $|p-q| < cN^{1/4}$, then N can be factored in $O_c(1)$ time.

How the factorization works:

- If $x \neq y$ are s.t. $x^2 \equiv y^2 \mod N$, then $N \mid (x+y)(x-y)$ and thus N shares a factor with $x \pm y$.

p and q too close

If p are q are too close, then factoring N is easy

Theorem 3 (Fermat's factorization method).

If $|p - q| < cN^{1/4}$, *then* N *can be factored in* $O_c(1)$ *time.*

How the factorization works:

- If $x \neq y$ are s.t. $x^2 \equiv y^2 \mod N$, then $N \mid (x+y)(x-y)$ and thus N shares a factor with $x \pm y$.
- starting from $\lceil \sqrt{N} \rceil$ and incrementing by 1, you need to test at most $O_c(1)$ numbers to find $x^2 \equiv y^2 \mod N$ s.t. $\gcd(N, x \pm y)$ is a factor of N.

p and q too close

If p are q are too close, then factoring N is easy

Theorem 3 (Fermat's factorization method).

If $|p-q| < cN^{1/4}$, then N can be factored in $O_c(1)$ time.

How the factorization works:

- If $x \neq y$ are s.t. $x^2 \equiv y^2 \mod N$, then $N \mid (x+y)(x-y)$ and thus N shares a factor with $x \pm y$.
- starting from $\lceil \sqrt{N} \rceil$ and incrementing by 1, you need to test at most $O_c(1)$ numbers to find $x^2 \equiv y^2 \mod N$ s.t. $\gcd(N, x \pm y)$ is a factor of N.
- method takes exponential time if instead p and q are not close

$p - 1$ having small factors

Tricky: **If** $p - 1$ **has only small factors,** $N = pq$ **can be factored quickly**

$p-1$ having small factors

Tricky: **If** $p-1$ **has only small factors,** $N = pq$ **can be factored quickly**

Pollard's $p-1$ method

Suppose we have the prime factorization

$$p - 1 = \prod_{q_i | p-1} q_i^{\alpha_i}$$

with $q_i < B$.

$p-1$ having small factors

Tricky: **If $p-1$ has only small factors, $N = pq$ can be factored quickly**

Pollard's $p-1$ method

Suppose we have the prime factorization

$$p - 1 = \prod_{q_i | p-1} q_i^{\alpha_i}$$

with $q_i < B$.
If we choose β_i s.t. $q_i^{\beta_i} \leqslant N < q_i^{\beta_i+1}$, then $\beta_i \geqslant \alpha_i$

$p-1$ having small factors

Tricky: **If $p-1$ has only small factors, $N = pq$ can be factored quickly**

Pollard's $p-1$ method

Suppose we have the prime factorization

$$p - 1 = \prod_{q_i | p-1} q_i^{\alpha_i}$$

with $q_i < B$.
If we choose β_i s.t. $q_i^{\beta_i} \leqslant N < q_i^{\beta_i+1}$, then $\beta_i \geqslant \alpha_i$
$\Rightarrow p-1 \mid \prod q_i^{\beta_i} =: R$.

$p-1$ having small factors

Tricky: **If $p-1$ has only small factors, $N = pq$ can be factored quickly**

Pollard's $p-1$ method

Suppose we have the prime factorization

$$p - 1 = \prod_{q_i | p-1} q_i^{\alpha_i}$$

with $q_i < B$.
If we choose β_i s.t. $q_i^{\beta_i} \leqslant N < q_i^{\beta_i+1}$, then $\beta_i \geqslant \alpha_i$
$\Rightarrow p-1 \mid \prod q_i^{\beta_i} =: R$.
$\Rightarrow$ for random a, $a^R \equiv 1 \mod N$, and $\gcd(a^R - 1, N)$ is a factor.

$p-1$ having small factors

Tricky: **If $p-1$ has only small factors, $N=pq$ can be factored quickly**

Pollard's $p-1$ method

Suppose we have the prime factorization

$$p-1=\prod_{q_i|p-1} q_i^{\alpha_i}$$

with $q_i < B$.
If we choose β_i s.t. $q_i^{\beta_i} \leqslant N < q_i^{\beta_i+1}$, then $\beta_i \geqslant \alpha_i$
$\Rightarrow p-1 \mid \prod q_i^{\beta_i} =: R$.
$\Rightarrow$ for random a, $a^R \equiv 1 \mod N$, and $\gcd(a^R-1, N)$ is a factor.
Therefore, given the list of primes up to B, we only have to keep multiplying $R' = \prod_{q_i \text{ prime} \leqslant r} q_i^{\beta_i}$ until $\gcd(a^{R'}-1, N)$ returns a divisor.

Blinding

You can be tricked into signing something that you don't want to

Blinding

You can be tricked into signing something that you don't want to

- Signing m means returning $s := m^d \mod N$ (nobody else has access to d); $s^e \equiv m \mod N$, proving to everyone it was us who signed.

Blinding

You can be tricked into signing something that you don't want to

- Signing m means returning $s := m^d \mod N$ (nobody else has access to d); $s^e \equiv m \mod N$, proving to everyone it was us who signed.
- A malicious attacker (Eve) wants us to sign m, which we won't agree to.

Blinding

You can be tricked into signing something that you don't want to

- Signing m means returning $s := m^d \mod N$ (nobody else has access to d); $s^e \equiv m \mod N$, proving to everyone it was us who signed.
- A malicious attacker (Eve) wants us to sign m, which we won't agree to.
- Instead, Eve picks r coprime with N and asks us to sign $m' := r^e m \mod N$, which looks innocent

Blinding

You can be tricked into signing something that you don't want to

- Signing m means returning $s := m^d \mod N$ (nobody else has access to d); $s^e \equiv m \mod N$, proving to everyone it was us who signed.
- A malicious attacker (Eve) wants us to sign m, which we won't agree to.
- Instead, Eve picks r coprime with N and asks us to sign $m' := r^e m \mod N$, which looks innocent
- we sign m' by returning Eve $s := (m')^d \mod N$.

Blinding

You can be tricked into signing something that you don't want to

- Signing m means returning $s := m^d \mod N$ (nobody else has access to d); $s^e \equiv m \mod N$, proving to everyone it was us who signed.
- A malicious attacker (Eve) wants us to sign m, which we won't agree to.
- Instead, Eve picks r coprime with N and asks us to sign $m' := r^e m \mod N$, which looks innocent
- we sign m' by returning Eve $s := (m')^d \mod N$.
- Eve calculates a signature on m by $sr^{-1} \mod N$

Blinding

You can be tricked into signing something that you don't want to

- Signing m means returning $s := m^d \mod N$ (nobody else has access to d); $s^e \equiv m \mod N$, proving to everyone it was us who signed.
- A malicious attacker (Eve) wants us to sign m, which we won't agree to.
- Instead, Eve picks r coprime with N and asks us to sign $m' := r^e m \mod N$, which looks innocent
- we sign m' by returning Eve $s := (m')^d \mod N$.
- Eve calculates a signature on m by $sr^{-1} \mod N$
- It is indeed a valid signature: $s \equiv r^{ed} m^d \equiv rm^d \mod N$.

Questions?