The Approximate GCD Problem

A post-quantum problem that is easier to understand than RSA

Martin R. Albrecht

28 March 2018

Outline

Greatest Common Divisors

Euclidean algorithm

Given two integers $a, b < N = 2^\kappa$ the Euclidean algorithm computes their greatest common divisor $\gcd(a, b)$.

```
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)
```

The Euclidean algorithm runs in time $\mathcal{O}\left(\kappa^2\right)$.

Best known algorithm runs in time $\mathcal{O}\left(\kappa \log^2 \kappa \log\log \kappa\right)$.[1]

For comparison, integer multiplication costs $\mathcal{O}\left(\kappa \log \kappa \log\log \kappa\right)$ using the Schönhage–Strassen algorithm.

[1] Damien Stehlé and Paul Zimmermann. A Binary Recursive Gcd Algorithm. In: *Algorithmic Number Theory, 6th International Symposium, ANTS-VI, Burlington, VT, USA, June 13-18, 2004, Proceedings.* Ed. by Duncan A. Buell. Vol. 3076. Lecture Notes in Computer Science. Springer, 2004, pp. 411–425. DOI: 10.1007/978-3-540-24847-7_31. URL: http://dx.doi.org/10.1007/978-3-540-24847-7_31.

RSA

Public key encryption

KeyGen Bob generates a key pair (sk, pk) and publishes pk.

Enc Alice uses pk to encrypt message m for Bob as c.

Dec Bob uses sk to decrypt c to recover m.

Naive RSA

KeyGen The public key is (N, e) and the private key is d, with

- $N = p \cdot q$ where p and q prime,
- e coprime to $\phi(N) = (p-1)(q-1)$ and
- d such that $e \cdot d \equiv 1 \mod \phi(N)$.

Enc $c \equiv m^e \bmod N$

Dec $m \equiv c^d \equiv m^{e \cdot d} \equiv m^1 \bmod N$

Caution

This naive version of RSA only achieves a very weak form of security — OW-CPA — even against classical adversaries: it is hard to recover random messages.

Classical attacks on RSA

- An adversary who can factor large integers can break RSA.
- The best known classical algorithm for factoring is the Number Field Sieve (NFS)
- It has a super-polynomial but sub-exponential (in $\log N$) complexity of

$$\mathcal{O}\left(e^{1.9(\log^{1/3} N)(\log\log^{2/3} N)}\right)$$

operations.

Classical attacks on RSA

- An adversary who can factor large integers can break RSA.
- The best known classical algorithm for factoring is the Number Field Sieve (NFS)
- It has a super-polynomial but sub-exponential (in $\log N$) complexity of

$$\mathcal{O}\left(e^{1.9(\log^{1/3} N)(\log\log^{2/3} N)}\right)$$

operations.

Caution

This does not mean an adversary **has** to factor to solve RSA.

Shared factors

What if two users generate moduli $N_0 = q_0 \cdot p$ and $N_1 = q_1 \cdot p$, i.e. moduli with shared factors?

- We assume that factoring each of N_0 or N_1 is hard.
- On the other hand, computing $\gcd(N_0, N_1)$ reveals p but costs only $\mathcal{O}\left(\kappa \log^2 \kappa \log\log \kappa\right)$ operations when $N_i \approx 2^\kappa$.

Quantum attacks on RSA

An adversary with access to a quantum computer with

$$\mathcal{O}\left(\log^2(N)\log\log(N)\log\log\log(N)\right)$$

gates can factor N using Shor's algorithm.[2]

[2] Peter W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In: *35th FOCS*. IEEE Computer Society Press, Nov. 1994, pp. 124–134.

The Approximate GCD problem

The Approximate GCD problem

The Approximate GCD problem is the problem of distinguishing

$$x_i = q_i \cdot p + r_i$$

from uniform $\mathbb{Z} \cap [0, X)$ with $x_i < X$ (q_i, r_i and p are secret).

The Approximate GCD problem

$$x_i = q_i \cdot p + r_i$$

If λ is our security parameter (think $\lambda = 128$), then

name	sizeof	DGHV10[3]	CheSte15[4]
γ	x_i	λ^5	$\lambda \log \lambda$
η	p	λ^2	$\lambda + \log \lambda$
ρ	r_i	λ	λ

[3] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In: *EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Springer, Heidelberg, May 2010, pp. 24–43.

[4] Jung Hee Cheon and Damien Stehlé. Fully Homomophic Encryption over the Integers Revisited. In: *EUROCRYPT 2015, Part I*. ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 513–536. DOI: `10.1007/978-3-662-46800-5_20`.

Naive encryption

KeyGen The public key is $\{x_i = q_i \cdot p + 2\, r_i\}_{0 \leq i < t}$ and the private key is p.

Enc For $m \in \{0, 1\}$ output $c = m + \sum b_i \cdot x_i$ with $b_i \leftarrow_{\$} \{0, 1\}$.

Dec $m = (c \bmod p) \bmod 2$.

Naive encryption

KeyGen The public key is $\{x_i = q_i \cdot p + 2\, r_i\}_{0 \leq i < t}$ and the private key is p.

Enc For $m \in \{0, 1\}$ output $c = m + \sum b_i \cdot x_i$ with $b_i \leftarrow_{\$} \{0, 1\}$.

Dec $m = (c \bmod p) \bmod 2$.

Note

This encryption scheme is not IND-CCA secure but it is IND-CPA secure if the AGCD problem is hard.

Attacks on the Approximate GCD problem

Exhaustive search

Given $x_0 = q_0 \cdot p + r_0$ and $x_1 = q_1 \cdot p + r_1$ we know that

$$p \mid \gcd\left((x_0 - r_0), (x_1 - r_1)\right)$$

Guess r_0 and r_1!

Cost
$2^{2\rho}$ GCDs

Exhaustive search + multiplication

Compute

$$\gcd\left(x_0', \prod_{i=0}^{2^\rho-1}(x_1 - i) \bmod x_0'\right)$$

for all $x_0' = x_0 - j$ with $0 \leq j < 2^{\rho-1}$.

Cost

2^ρ GCDs, $2^{2\rho}$ multiplications

Lemma

Assume that we have τ samples $x_0, \ldots, x_{\tau-1}$ of a given prime p, of the hidden form $x_i = q_i \cdot p + r_i$, then p can then be recovered with overwhelming probability in time $\tilde{\mathcal{O}}(2^{\frac{\tau+1}{\tau-1}\rho})$.[5]

[5] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers. In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 446–464.

Lattice attacks

Given $x_0 = q_0 p + r_0$ and $x_1 = q_1 p + r_1$, consider

$$\begin{aligned} q_0 x_1 - q_1 x_0 &= q_0(q_1 p + r_1) - q_1(q_0 p + r_0) \\ &= q_0 q_1 p + q_0 r_1 - q_1 q_0 p - q_1 r_0 \\ &= q_0 r_1 - q_1 r_0 \end{aligned}$$

and note that

$$q_0 x_1 - q_1 x_0 \ll x_i$$

Given $x_0 = q_0 p + r_0$ and $x_1 = q_1 p + r_1$, consider

$$\begin{aligned} q_0 x_1 - q_1 x_0 &= q_0(q_1 p + r_1) - q_1(q_0 p + r_0) \\ &= q_0 q_1 p + q_0 r_1 - q_1 q_0 p - q_1 r_0 \\ &= q_0 r_1 - q_1 r_0 \end{aligned}$$

and note that

$$q_0 x_1 - q_1 x_0 \ll x_i$$

Non-starter?

We don't know q_i!

Lattice attacks

Consider the matrix

$$\mathbf{B} = \begin{pmatrix} 2^{\rho+1} & x_1 & x_2 & \cdots & x_t \\ & -x_0 & & & \\ & & -x_0 & & \\ & & & \ddots & \\ & & & & -x_0 \end{pmatrix}$$

multiplying on the left by the vector $\mathbf{q} = (q_0, q_1, q_2, \cdots, q_t)$ gives

$$\begin{aligned} \mathbf{v} &= (q_0, q_1, \cdots, q_t) \cdot \mathbf{B} \\ &= (q_0\, 2^{\rho+1}, q_0 x_1 - q_1 x_0, \cdots, q_0 x_t - q_t x_0) \\ &= (q_0\, 2^{\rho+1}, q_0 r_1 - q_1 r_0, \cdots, q_0 r_t - q_t r_0) \end{aligned}$$

which is a vector with small coefficients compared to x_i.

The set of all integer-linear combinations of the rows of $\mathbf{B}$ the lattice spanned by (the rows of) $\mathbf{B}$.

SVP finding a shortest non-zero vector on general lattices is NP-hard.

Gap-SVP$_\gamma$ Differentiating between instances of SVP in which the answer is at most 1 or larger than γ on general lattices is a well-known and presumed quantum-hard problem for γ polynomial in lattice dimension.

Easy SVP

GCD is SVP on $\mathbb{Z}^2$. For example, $\mathbf{B} = [21, 14]^T$, $\mathbf{v} = (-1, 1)$, $\mathbf{v} \cdot \mathbf{B} = 7$.

Reduction to presumed hard lattice problem

We can show that an adversary has to solve Gap-SVP.

AGCD $\rightarrow$ LWE

If there is an algorithm efficiently solving the AGCD problem then there exists an algorithm which solves the **Learning with Errors** (LWE) problem with essentially the same performance.[6]

LWE $\rightarrow$ Gap-SVP

If there is an algorithm efficiently solving the LWE problem then there exists a quantum algorithm which solves worst-case Gap-SVP instances.[7]

---

[6] Jung Hee Cheon and Damien Stehlé. Fully Homomophic Encryption over the Integers Revisited. In: *EUROCRYPT 2015, Part I*. ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 513–536. DOI: 10.1007/978-3-662-46800-5_20.

[7] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In: *37th ACM STOC*. ed. by Harold N. Gabow and Ronald Fagin. ACM Press, May 2005, pp. 84–93.

Learning with Errors (in normal form)

Given $(\mathbf{A}, \mathbf{c})$ with $\mathbf{c} \in \mathbb{Z}_q^m$, $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, small $\mathbf{s} \in \mathbb{Z}^n$ and small $\mathbf{e} \in \mathbb{Z}^m$ is

$$\left(\begin{array}{c} \\ \\ \mathbf{c} \\ \\ \\ \end{array}\right) = \left(\begin{array}{ccc} \leftarrow & n & \rightarrow \\ & & \\ & \mathbf{A} & \\ & & \\ & & \end{array}\right) \times \left(\begin{array}{c} \\ \mathbf{s} \\ \\ \end{array}\right) + \left(\begin{array}{c} \\ \\ \mathbf{e} \\ \\ \\ \end{array}\right)$$

or $\mathbf{c} \leftarrow_{\$} \mathcal{U}(\mathbb{Z}_q^m)$.

From vectors to scalars

LWE with modulus q^n and dimension 1 is as hard as LWE with modulus q and dimension 1.

$$q^{d-1} \cdot \langle \mathbf{a}, \mathbf{s} \rangle \approx \left(\sum_{i=0}^{n-1} q^i \cdot a_i \right) \cdot \left(\sum_{i=0}^{d-1} q^{d-i-1} \cdot s_i \right) \bmod q^d = \tilde{a} \cdot \tilde{s} \bmod q^d.$$

Example

$$\begin{aligned}
(a_0 + q \cdot a_1) \cdot (q \cdot s_0 + \cdot s_1) &= q(a_0 \cdot s_0 + a_1 \cdot s_1) + (a_1 \cdot s_1) + q^2 (a_1 \cdot s_0) \\
&\equiv q(a_0 \cdot s_0 + a_1 \cdot s_1) + (a_1 \cdot s_1) \bmod q^2 \\
&\approx q(a_0 \cdot s_0 + a_1 \cdot s_1) \bmod q^2
\end{aligned}$$

Fin

Questions?

Bonus

Given $c_i = q_i \cdot p + m'_i$ with $m'_i = 2\,r_i + m_i$.

- We can compute

$$c' = c_0 \cdot c_1 = q_0\,q_1\,p^2 + q_0\,m'_1\,p + q_1\,m'_0\,p + m'_0 \cdot m'_1$$

  to get $c' \bmod p = m'_0 \cdot m'_1$ and $m'_0 \cdot m'_1 \bmod 2 = m_0 \cdot m_1$.

- We can also compute

$$c' = c_0 + c_1 = (q_0 + q_1)p + (m'_0 + m'_1)$$

  to get $c' \bmod p \bmod 2 = m_0 \oplus m_1$.

We can compute with encrypted data.[8]

[8] https://crypto.stanford.edu/craig/easy-fhe.pdf