Chapter 5

Hard Number Theoretic Problems

The following table gives an overview over cryptologically relevant number theoretic computational problems. "Efficient" means "computable with polynomial cost", "ERH" means "if the extended RIEMANN hypothesis holds", "prob." means "by a probabilistic algorithm".

Computational problem	**efficient?**	**treated in**
Primality test	yes	3.1–3.8
For a prime number p		
Finding a primitive element	yes (ERH or prob.)	A.2, A.9
Finding a quadratic nonresidue	yes (ERH or prob.)	A.8
Quadratic residuosity	yes	A.6
Taking a square root	yes (ERH or prob.)	follows in 5.3
Discrete logarithm	? (probably no)	4.1, 4.6
For a composite integer n		
Factoring	? (probably no)	2.2, 2.4
RSA inversion (e-th root)	? (probably no)	2.2
Computation of EULER function	? (probably no)	2.2
Computation of CARMICHAEL f.	? (probably no)	2.2
Finding a primitive element	? (probably no)	A.4
Quadratic residuosity	? (probably no)	A.11
Taking a square root	? (probably no)	follows in 5.5
Discrete logarithm	? (probably no)	follows in 5.1

Figure 5.1 shows the connection between computational problems for a composite integer n. An arrow from A to B indicates that problem B reduces to Problem A by an efficient (maybe probabilistic) algorithm. For an unidirectional arrow the reverse direction is unknown. Reductions indicated by red arrows will be proved in this chapter (sometimes only in the case

where n is a product of two primes). [The task denoted by "Pol. fact." means factoring polynomials in one variable over the residue class ring $\mathbb{Z}/n\mathbb{Z}$. We wont treat it in these lecture notes.]

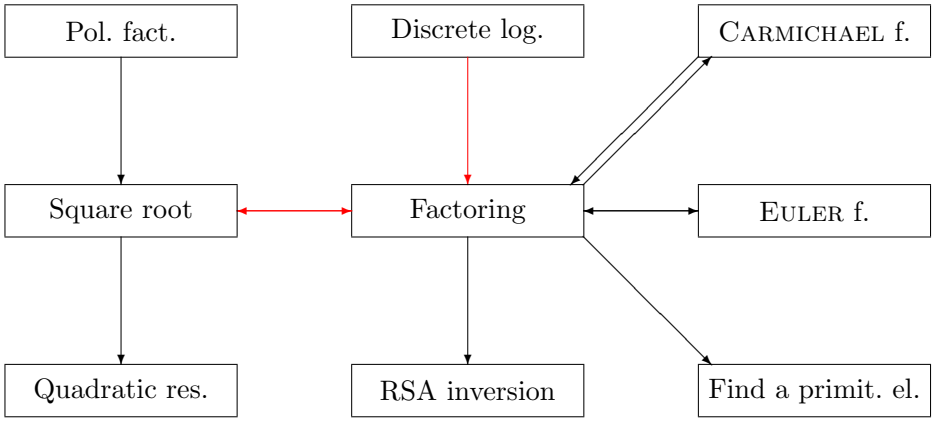

Figure 5.1: Connection between computational problems for a composite module

5.1 Discrete Logarithm and Factorization

Let $a \in \mathbb{M}_n$, $\operatorname{ord} a = s$, and consider the exponential function

$$\exp_a : \mathbb{Z} \longrightarrow \mathbb{M}_n$$

The problem of computing discrete logarithms $\bmod n$ is to find an algorithm that for each $y \in \mathbb{M}_n$

- outputs "no" if $y \notin \langle a \rangle$,
- else outputs an $r \in \mathbb{Z}$ with $0 \leq r < s$ and $y = a^r \bmod n$.

Proposition 15 (E. Bach) *Let $n = pq$ with different primes $p, q \geq 3$. Then factoring n admits a probabilistic efficient reduction to the computation of discrete logarithms $\bmod n$.*

Proof. We have $\varphi(n) = (p-1)(q-1)$. For a randomly chosen $x \in \mathbb{M}_n$ always $x^{\varphi(n)} \equiv 1 \pmod{n}$. Let $y := x^n \bmod n$, thus

$$y \equiv x^n \equiv x^{n-\varphi(n)} = x^{pq-(p-1)(q-1)} = x^{p+q-1} \pmod{n}.$$

The discrete logarithm yields an r with $0 \leq r < \operatorname{ord} x \leq \lambda(n)$ and $y = x^r \bmod n$. Hence

$$x^{r-(p+q-1)} \equiv 1 \pmod{n}, \quad \operatorname{ord} x \mid r - (p+q-1).$$

Since $|r - (p+q-1)| < \lambda(n)$ the probability is high that $r = p+q-1$. This happens for example if $\operatorname{ord} x = \lambda(n)$. Otherwise choose another x.

From the two equations

$$\begin{aligned} p + q &= r + 1 \\ p \cdot q &= n \end{aligned}$$

we easily compute the factors p and q. $\diamond$

5.2 Square Roots and Factorization

Proposition 16 (M. Rabin) *Let $n = pq$ with different primes $p, q \geq 3$. Then factoring n admits a probabilistic efficient reduction to taking square roots* mod n.

Proof. $\mathbb{Z}/n\mathbb{Z}$ contains four different roots of unity, hence also four different square roots of each square in $\mathbb{M}_n$.

For a random choice of $x \in M_n$ the square root algorithm provides a root $y \in \mathbb{M}_n$ of x^2, thus

$$y^2 \equiv x^2 \pmod{n}.$$

The probability that $y \not\equiv \pm x \pmod{n}$ is $\frac{1}{2}$. Since

$$n \mid (x^2 - y^2) = (x + y)(x - y), \quad n \nmid (x \pm y),$$

$\gcd(n, x + y)$ is a proper divisor of n. $\diamond$

Therefore an attacker who knows how to take square roots mod n also can factorize n.

The reverse direction follows in Section 5.5.

5.3 Square Roots in Finite Prime Fields

In many cases taking square roots is a trivial task as the following simple consideration shows:

Lemma 9 *Let G be a finite group of odd order m. Then for each $a \in G$ there is exactly one $x \in G$ with $x^2 = a$, and it is given by $x = a^{\frac{m+1}{2}}$.*

Proof. Since $a^m = 1$ we have $x^2 = a^{m+1} = a$. We conclude that the squaring map $x \mapsto x^2$ is surjective, hence a bijection $G \longrightarrow G$. $\diamond$

We search methods for taking square roots in a finite prime field $\mathbb{F}_p$ as efficiently as possible. The case $p \equiv 3 \pmod 4$ is extremely simple by the foregoing consideration: If $p = 4k + 3$, then the group $\mathbb{M}_p^2$ of quadratic residues has odd order $\frac{p-1}{2} = 2k + 1$. Hence for a quadratic residue $z \in \mathbb{M}_p^2$ the unique square root is $x = z^{k+1} \bmod p$ [Lagrange 1769]. The cost of taking this square root is at most $2 \cdot \log_2(p)$ congruence multiplications.

Examples

1. For $p = 7 = 4 \cdot 1 + 3$ we have $k + 1 = 2$. By A.8 2 is a quadratic residue. A square root is $2^2 = 4$. Check: $4^2 = 16 \equiv 2$.

2. For $p = 23 = 4 \cdot 5 + 3$ we have $k + 1 = 6$. By A.8 again 2 is a quadratic residue. A square root is $2^6 = 64 \equiv 18$. Check: $18^2 \equiv (-5)^2 = 25 \equiv 2$.

Unfortunately for $p \equiv 1 \pmod 4$ we cannot hope for such a simple procedure. For example -1 is a quadratic residue, but no power of -1 can be a square root of -1 since always $[(-1)^m]^2 = (-1)^{2m} = 1 \neq -1$.

Fortunately there are general procedures, for example one that is baptized AMM after Adleman, Manders, and Miller, but was described already by Cipolla in 1903. It starts by decomposing $p - 1$ into $p - 1 = 2^e \cdot u$ with odd u. Furthermore we choose (once and for all) an arbitrary quadratic nonresidue $b \in \mathbb{F}_p^\times - \mathbb{M}_p^2$—this is the only nondeterministic step in the algorithm, see Section A.8. (Assuming ERH the procedure is even deterministic, as it is in the many cases where a quadratic nonresidue is known anyway.)

Now we consider a quadratic residue $z \in \mathbb{M}_p^2$ and want to find a square root of it. Since $z \in \mathbb{M}_p^2$, we have $\operatorname{ord}(z) \mid \frac{p-1}{2}$, hence the 2-order $r = \nu_2(\operatorname{ord}(z))$ of $\operatorname{ord}(z)$ is bounded by $\leq e - 1$, and r is minimal with $z^{u2^r} \equiv 1$.

We recursively define a sequence $z_1, z_2, \ldots$ beginning with

$$z_1 = z \quad \text{with } r_1 = \nu_2(\operatorname{ord}(z_1)).$$

If $z_i \in \mathbb{M}_p^2$ is chosen, and r_i is the 2-order of $\mathrm{ord}(z_i)$, then the sequence terminates if $r_i = 0$. Otherwise we set

$$z_{i+1} = z_i \cdot b^{2^{e-r_i}}.$$

Then $z_{i+1} \in \mathbb{M}_p^2$. Furthermore

$$z_{i+1}^{u \cdot 2^{r_i - 1}} \equiv z_i^{u \cdot 2^{r_i - 1}} \cdot b^{u \cdot 2^{e-1}} \equiv 1,$$

since the first factor is $\equiv -1$ due to the minimality of r_i, and the second factor is $\equiv (\frac{b}{p}) = -1$, for $u \cdot 2^{e-1} = \frac{p-1}{2}$. Hence $r_{i+1} < r_i$. The terminating condition $r_n = 0$ is reached after at most e steps with $n \leq e \leq \log_2(p)$.

Then we compute reversely:

$$x_n = z_n^{\frac{u+1}{2}} \bmod p$$

with $x_n^2 \equiv z_n^{u+1} \equiv z_n$ (since $\mathrm{ord}(z_n) \mid u$ by its odd parity). Recursively

$$x_i = x_{i+1} / b^{2^{e-r_i-1}} \bmod p$$

that by induction satisfies

$$x_i^2 \equiv x_{i+1}^2 / b^{2^{e-r_i}} \equiv z_{i+1} / b^{2^{e-r_i}} \equiv z_i.$$

Hence $x = x_1$ is a square root of z.

In addition to the cost of finding b we count the following steps:

- Computing the powers $b^2, \ldots, b^{2^{e-1}}$, costing $(e-1)$ modular squares.
- Computing the powers $b^u, b^{2u}, \ldots, b^{2^{e-1}u}$, taking at most $2 \cdot \log_2(u) + e - 1$ congruence multiplications.
- Computing z^u, taking at most $2 \cdot \log_2(u)$ congruence multiplications.
- Furthermore we compute for each $i = 1, \ldots, n \leq e$:
  - z_i by one congruence multiplication,
  - z_i^u from z_{i-1}^u by one congruence multiplication,
  - $z_i^{u2^r}$ from $z_{i-1}^{u2^r}$ by one congruence multiplication,
  - and then r_i.

  This makes a total of at most $3 \cdot (e-1)$ congruence multiplications.
- x_n as a power by at most $2 \cdot \log_2(u)$ congruence multiplications.
- x_i from x_{i+1} each by one congruence division with cost $\mathrm{O}(\log(p)^2)$.

Summing up we get costs of size about $\mathrm{O}(\log(p)^3)$ with a rather small constant coefficient.

Example Let $p = 29$ and $z = 5$. Then $p - 1 = 4 \cdot 7$, hence $e = 2$ and $u = 7$. By the remarks above $b = 2$ is a quadratic nonresidue. We compute the powers

$$b^2 = 4, \quad b^u \equiv 128 \equiv 12, \quad b^{2u} \equiv 144 \equiv -1,$$

$$z^2 \equiv 25 \equiv -4, \quad z^4 \equiv 16, \quad z^6 \equiv -64 \equiv -6, \quad z^7 \equiv -30 \equiv -1.$$

Now

$$z_1 = 5, \quad z_1^u \equiv -1, \quad z_1^{2u} \equiv 1, \quad r_1 = 1,$$

$$z_2 \equiv z_1 b^2 \equiv 5 \cdot 4 = 20, \quad z_2^u \equiv z_1^u b^{2u} \equiv (-1)(-1) = 1, \quad r_2 = 0.$$

Now we go backwards:

$$x_2 \equiv z_2^{\frac{u+1}{2}} = z_2^4 = (z_2^2)^2 \equiv 400^2 \equiv (-6)^2 = 36 \equiv 7,$$

$$x_1 = x_2/b \bmod p = 7/2 \bmod 29 = 18.$$

Hence $x = 18$ is the wanted root. Check: $18^2 = 324 \equiv 34 \equiv 5$.

Exercises Find deterministic algorithms (= simple formulas) for taking square roots in the fields

- $\mathbb{F}_p$ with $p \equiv 5 \pmod 8$
- $\mathbb{F}_{2^m}$ with $m \geq 2$ [Hints: 1. Consider the order of the radicand in the multiplicative group. 2. Invert the linear map $x \mapsto x^2$.]
- $\mathbb{F}_q$ for $q = p^m$

Alternative algorithms: Almost all known efficient algorithms that completely cover the case $p \equiv 1 \pmod 4$ are probabilistic and have a deterministic variant whose cost is polynomial assuming ERH. The book by FORSTER (*Algorithmische Zahlentheorie*) has a variant of the CIPOLLA/AMM algorithm that uses the quadratic extension $\mathbb{F}_{p^2} \supseteq \mathbb{F}_p$ and is conceptionally quite simple. The *Handbook of Applied Cryptography* (MENEZES/VAN OORSCHOT/VANSTONE) contains an algorithm by TONELLI 1891 that admits a concise formulation, but cost $O(\log(p)^4)$. Another method is a special case of the CANTOR/ZASSENHAUS algorithm for factoring polynomials over finite fields, see VON ZUR GATHEN/GERHARD: *Modern Computer Algebra.* Yet another procedure by LEHMER uses the LUCAS sequence (a_n) with $a_1 = b$, $a_2 = b^2 - 2z$, where $b^2 - 4z$ is a quadratic nonresidue. The only known deterministic algorithm with proven polynomial cost was given by SCHOOF. It uses elliptic curves, and costs $O(\log(p)^9)$, so it is of theoretical interest only.

For overviews see:

- E. Bach/ J. Shallit: *Algorithmic Number Theory*. MIT Press, Cambridge Mass. 1996.
- D. J. Bernstein: Faster square roots in annoying finite fields. Preprint (siehe die Homepage des Autors `http://cr.yp.to/`).

5.4 Square Roots for Prime Power Modules

A simple procedure (implicitly using HENSEL's lifting) allows to extend the square root algorithms from prime modules to prime powers. Let p be a prime $\neq 2$, and let $e \geq 2$. Let z be a quadratic residue $\bmod p^e$. We want to find a square root of z.

Of course z is also a quadratic residue $\bmod p^{e-1}$. Assume we already have found a root for it, that is a y with $y^2 \equiv z \pmod{p^{e-1}}$. Let

$$a = 1/(2y) \bmod p$$

and $y^2 - z = p^{e-1} \cdot u$. We set

$$x := y - a \cdot (y^2 - z) \bmod p^e.$$

Then we have

$$\begin{aligned} x^2 &\equiv y^2 - 2ay(y^2 - z) + a^2(y^2 - z)^2 \equiv y^2 - 2ayp^{e-1}u \\ &\equiv y^2 - p^{e-1}u = z \pmod{p^e}. \end{aligned}$$

Hence x is a square root of $z \bmod p^e$.

We wont explicit this algorithm but illustrate it with two examples:

Examples

1. $n = 25$, $z = 19$. We have $p = 5$, $19 \bmod 5 = 4$. Hence we can take $y = 2$ and $a = 1/4 \bmod 5 = 4$. Then $y^2 - z = -15$ and

   $$x = 2 + 15 \cdot 4 \bmod 25 = 62 \bmod 25 = 12.$$

   Check: $12^2 = 144 = 125 + 19$.

2. $n = 27$, $z = 19$. We have $p = 3$, $19 \bmod 3 = 1$. Hence in the first step we can take $y = 1$ and $a = 1/2 \bmod 3 = 2$. Then $y^2 - z = -18$ and

   $$x = 1 + 2 \cdot 18 \bmod 9 = 37 \bmod 9 = 1.$$

   For the second step (from 9 to 27) again $y = 1$, $y^2 - z = -18$, and

   $$x = 37 \bmod 27 = 10.$$

   Check: $10^2 = 100 = 81 + 19$.

The costs consist of two contributions:

1. One square root $\bmod p$ and one division. (The quotient a needs to be computed only once since $x \equiv y \pmod p$.)

2. Each time the exponent is incremented we execute two congruence multiplications and two subtractions.

Hence the total cost is $O(\log(n)^3)$ for the module n.

Finally we have to consider the case where $n = 2^e$ is a power of two.

For $e \leq 3$ the only quadratic residue is 1, its square root is 1.

For larger exponents e we have again a recurrence to $e-1$: Let z be an odd integer (all invertible elements are odd). Assume we already found a y with $y^2 \equiv z \pmod{2^{e-1}}$. Then $y^2 - z = 2^{e-1} \cdot t$. If t is even, then $y^2 \equiv z \pmod{2^e}$. Otherwise we set $x = y + 2^{e-2}$. Then

$$x^2 \equiv y^2 + 2^{e-1}y + 2^{2e-4} \equiv z + 2^{e-1} \cdot (t + y) \equiv z \pmod{2^e},$$

since $t + y$ is even. Hence $x = y$ or $y + 2^{e-2}$ is a square root of z. Here the cost is even smaller than $O(\log(n)^3)$.

By the way we have shown that z is a quadratic residue $\bmod 2^e$ (for $e \geq 3$) if and only if $z \equiv 1 \pmod 8$.

5.5 Square Roots for Composite Modules

If we know the prime decomposition of the module n, then we can efficiently compute square roots in $\mathbb{M}_n$. The two tasks "factoring" and "computing square roots" are equivalent with respect to their complexity.

For an execution of the procedure we successively decompose n into coprime factors (down to the prime powers).

So let $n = rs$ with coprime factors r and s. First we compute coefficients a and b such that $ar + bs = 1$ using the extended Euclidean algorithm.

We want to find a square root of z. Let u be a square root $\bmod\ r$ and v be a square root $\bmod\ s$. Then $x := arv + bsu \bmod n$ satisfies the congruences:

$$\begin{aligned} x &\equiv bsu \equiv u \pmod{r}, & x &\equiv arv \equiv v \pmod{s}, \\ x^2 &\equiv u^2 \equiv z \pmod{r}, & x^2 &\equiv v^2 \equiv z \pmod{s}, \end{aligned}$$

hence $x^2 \equiv z \pmod{n}$.

The cost for this procedure is two square roots modulo the factors, one Euclidean algorithm, and four congruence multiplications (+ 1 congruence addition). Hence it is $O(\log(n)^3)$.

For BLUM integers (see Appendix A.11) we even have a simpler algorithm, namely an explicit formula:

Corollary 1 *Let $n = pq$ with primes $p, q \equiv 3 \pmod{4}$. Then*

(i) *$d = \frac{(p-1)(q-1)+4}{8}$ is an integer.*

(ii) *For each quadratic residue $x \in \mathbb{M}_n^2$ the power x^d is the (unique) square root of x in $\mathbb{M}_n^2$.*

Proof. (i) If $p = 4k + 3$, $q = 4l + 3$, then $(p-1)(q-1) = 16kl + 8k + 8l + 4$, hence $d = 2kl + k + l + 1$.

(ii) The exponent of the multiplicative group $\mathbb{M}_n$,

$$\lambda(n) = \mathrm{kgV}(p-1, q-1) = 2 \cdot \mathrm{kgV}(2k+1, 2l+1)$$

is a divisor of $2 \cdot (2k+1) \cdot (2l+1)$, The exponent of the subgroup $\mathbb{M}_n^2$ of squares is $\frac{\lambda(n)}{2}$, hence a divisor of $(2k+1)\cdot(2l+1) = 4kl+2k+2l+1 = 2d-1$. Thus $x^{2d} \equiv x \pmod{n}$ for all $x \in \mathbb{M}_n^2$, thus the square of x^d is x. $\diamond$

This simple formula has the effect that the RABIN cipher is especially easy to handle for BLUM integer modules.