Lecture 6: Deterministic Primality Testing

Topics in Pseudorandomness and Complexity (Spring 2018)
Rutgers University
Swastik Kopparty

Scribe: Justin Semonsen, Nikolas Melissaris

1 Introduction

The AKS (Agrawal-Kayal-Saxena) algorithm, found in 2002, is the first ever deterministic polynomial-time primality testing algorithm. The algorithm is based on a generalization of Fermat's Little Theorem to polynomial rings over finite fields: if a number a is co-prime to n, $n > 1$, then:

$$n \text{ is prime iff } (x+a)^n \equiv x^n + a \pmod{n} \tag{1}$$

2 The AKS Algorithm

$A = \log^{10} n$
$R = \log^6 n$

1. If n is a perfect power, output `composite`.
2. If n has a factor smaller than R, output `composite`.
3. For each $a \in [A]$:
   For each $r \in [R]$:
   Check that $(x+a)^n \equiv x^n + a \pmod{(n, x^r - 1)}$

We can check efficiently this identity by repeated squaring.

3 Proof (AKS)

Claim 1. $\exists r_0 < R$ s.t. $\gcd(r,n) = 1$ and $c = \text{ord}(n) \pmod{r} > \log^2 n$

The last inequality means that all $n, n^2, \cdots, n^c$ are distinct $\pmod{r}$.

Proof. We look at $M = n(n-1)(n^2-1)\cdots(n^{\log^2 n}-1)$.
This means that $M \leq n^{\log^4 n} = 2^{\log^5 n}$ which in turn implies that there is some prime $r < R$ that doesn't divide M. That is the r we are looking for because $\forall a < \log^2 n, r \nmid n^a - 1 \Rightarrow$
$\text{ord}(n) \pmod{r} \geq \log^2 n$ $\square$

Suppose n is composite and not a power and not divisible by any prime less that R. We want to show that:

$$\exists a \leq A \text{ s.t. } (x+a)^n \not\equiv x^n + a \pmod{(n, x^{r_0}-1)} \tag{2}$$

Suppose not. This means that:

$$(x+a)^n \equiv x^n + a \pmod{(n, x^{r_0}-1)} \qquad \forall a \in [A] \tag{3}$$

Take p s.t. $p \mid n,\ \ r_0 \nmid p-1$. Since $n \not\equiv 1 \pmod{r_0}$, some prime factor of n has $p \not\equiv 1 \pmod{r_0}$.

Now we work over $\mathbb{F}_p[x]$. We know that:

$$(x+a)^n \equiv x^n + a \pmod{x^{r_0}-1} \tag{4}$$

Let $H = \{a \in \overline{\mathbb{F}}_p : a^{r_0} = 1\}$.

From (4) we get that $\forall \alpha \in H \quad (\alpha+a)^n = \alpha^n + a \quad \forall a \in A$.

Definition 2. *We say that $Q(x)$ and m commute if* $\forall \alpha \in H \quad Q(\alpha^m) = (Q(\alpha))^m$

From (4) we get that $\forall a \in [A]$, $(x+a)$ and n commute: $\forall \alpha \in H \quad (\alpha+a)^n = \alpha^n + a$.

Claim 3. *$\forall Q(x) \in \mathbb{F}_p$, $Q(x)$ and p commute: $\forall \alpha \in H$, $Q(\alpha)^p = Q(\alpha^p)$.*

Proof. $Q(\alpha)^p = \left(\sum a_i \alpha^i\right)^p = \sum a_i^p \alpha^{ip} = \sum a_i \alpha^{ip} = Q(\alpha^p)$ □

The proof is based on the fact that in $\mathbb{F}_p$, $(a+b)^p = a^p + b^p$. For $a \in \mathbb{F}_p$, $a^p = a$.

Lemma 4. *If Q_1, Q_2 both commute with m then so does $Q_1 Q_2$.*

Proof. Given: $\forall \alpha \in H$, $Q_1(\alpha^m) = Q_1(\alpha)^m$, $Q_2(\alpha^m) = Q_2(\alpha)^m$ then
$\forall \alpha \in H$, $Q_1 Q_2(\alpha^m) = Q_1(\alpha^m) Q_2(\alpha^m) = Q_1(\alpha)^m Q_2(\alpha)^m = (Q_1 Q_2(\alpha))^m$ □

Lemma 5. *If Q commutes with m_1, m_2 then Q commutes with $m_1 m_2$.*

Proof. Given: $\forall \alpha \in H$, and

(i). $Q(\alpha^{m_1}) = Q(\alpha)^{m_1}$

(ii). $Q(\alpha^{m_2}) = Q(\alpha)^{m_2}$

$$Q(\alpha^{m_1 m_2}) = Q\left(\left((\alpha)^{m_1}\right)^{m_2}\right) \overset{\substack{\alpha^{m_1} \in H \\ (ii)}}{=} Q\left(\alpha^{m_1}\right)^{m_2} \overset{(i)}{=} Q\left(\alpha\right)^{m_1 m_2}$$ □

We define $S = \{x + a : a \in [A] \subseteq \mathbb{F}_p[x]\}$ and $T = \{n, p\} \subseteq \mathbb{Z}$. Every element in S commutes with every element in T. Moreover we have $\overline{S}$ the multiplicative closure of S, which is the set of products of $(x+a)$'s, and $\overline{T}$ the multiplicative closure of T with is the set $\{n^i p^j : i, j > 0\}$.

Then, by the two lemmas 3 and 4, every element of $\overline{S}$ commutes with every element of $\overline{T}$.

Let $G = \overline{T} \pmod{r}_0$. G is a group and a subgroup of $\left(\mathbb{Z}^*_{r_0}, \times\right)$. Let $t = |G|$

4 Continuation of proof

Note that H is a cyclic group, and thus $\exists\, \alpha_0 \in H$ such that $H = \{1, \alpha_0, \alpha_0^2, \ldots, \alpha_0^{r_0-1}\}$. This is a known result from algebra.

Lemma 6. *$\forall\, Q_1 \neq Q_2 \in \overline{S}$ with $\deg(Q_1), \deg(Q_2) < t$, then $Q_1(\alpha_0) \neq Q_2(\alpha_0)$.*

Proof. Assume that $Q_1(\alpha_0) = Q_2(\alpha_0)$. Then $Q_1(\alpha_0^m) = Q_1(\alpha_0)^m \neq Q_2(\alpha_0)^m = Q_2(\alpha_0^m)$ for any $m \in \overline{T}$.

However, since $\overline{T}$ is a multiplicative group and $\alpha_0^{r_0} = 1$, any $\alpha \in G$ can be written as a α_0^m for some $m \in \overline{T}$. This means that $Q_1(\alpha) = Q_2(\alpha)$ for every $\alpha \in G$.

This means that Q_1 agrees with Q_2 on t elements, so since $\deg(Q_1), \deg(Q_2) < t$, this means that $Q_1 = Q_2$. □

This means that we can let $B = \{Q(\alpha_0) : \deg(Q) \leq t-1, Q \in S\}$. Note that B includes $Q(x) = \prod_i = 1^{t-1}(x - a_i)$ for any distinct choices of a_i, so $|B| \geq \binom{A}{t-1} \geq \left(\frac{A}{t-1}\right)^{t-1} \geq 2^t$.

Now we upper bound this set, and get a contradiction out of that.

Note that if $n \neq p^l$ then $n^i p^j = n^{i'} p^{j'}$ only when $(i,j) = (i',j')$. However, since $\alpha_0^{n^i p^j} \in G$ for any i, there are only t distinct exponents mod r_0. Therefore $|\{n^i p^j \mod r_0 : i,j \leq \sqrt{t}+1\}| \leq t$.

This means $\exists\, m_1, m_2$ of the form $n^i p^j \mod r_0 : i,j \leq \sqrt{t}+1$ such that $m_1 \cong m_2 \mod r_0$, but $m_1 \neq m_2$.

Therefore $\alpha_0^{m_1} = \alpha_0^{m_2}$, so $Q(\alpha_0)^{m_1} = Q(\alpha_0^{m_1}) = Q(\alpha_0^{m_2}) = Q(\alpha_0)^{m_2}$ for any $Q \in \overline{S}$. Because we chose $r_0 \neq p-1$, this means that $\alpha_0 \notin F_p$, so $Q(\alpha_0) \neq 0$. This means that for every $Q \in \overline{S}, Q(\alpha_0)^{\overline{m}} = 1$, where $\overline{m} = m_1 - m_2$.

This means that for any $b \in B$, we have that $b^{\overline{m}} = 1$. Since this polynomial has no more than $\overline{m}$ roots, $|B| \leq \overline{m} \leq n^{2(\sqrt{t}+1)} \leq 2^{3\sqrt{t}\log(n)}$.

However, since t is the order of $n \mod r_0$, this means that $t >> \log^2(n)$. This contradiction proves the correctness of the algorithm.

5 Discrete Square Root

The problem consists of a given p prime, and a number $a \in \mathbb{F}_p$. We would like to find $b \in \mathbb{F}_p$ (if one exists) such that $b^2 = a$. We assume that $p > 2$.

Note that we can tell if a is a square by checking that $a^{\frac{p-1}{2}} = 1$.

Claim 7. *a is a square if and only if $a^{\frac{p-1}{2}} = 1$.*

Proof. Since $a^{p-1} = 1$, we have that $\left(a^{\frac{p-1}{2}} - 1\right)\left(a^{\frac{p-1}{2}} + 1\right) = 0$. This means that $a^{\frac{p-1}{2}}$ is either 1 or -1.

If a is a square, then let b be such that $a = b^2$. Now $a^{\frac{p-1}{2}} = b^{p-1} = 1$.

Consider the homomorphism $\mathbb{F}_p^* \to \{\text{squares}\}$ given by $x \to x^2$. Since both -1 and 1 map to 1, the size of the kernel is 2. Therefore there are $\frac{p-1}{2}$ squares, which are all the roots of $x^{\frac{p-1}{2}} - 1$. $\square$

Berlekamp's Algorithm is a probabilistic algorithm for finding the discrete square root. The algorithm proceeds as follows:

1. Choose $c, d \in \mathbb{F}_p$ uniformly at random.
2. Compute $GCD(x^{\frac{p-1}{2}} - 1, (cx+d)^2 - a)$.
3. If this result is degree 1, then the discrete square root is $cx + d$ for x solving the linear equation. Otherwise, the algorithm fails.

Since $x^{\frac{p-1}{2}} - 1$ is sparse, it is fairly simple to use repeated exponentiation to find $x^{\frac{p-1}{2}} - 1 \mod (cx + d)^2 - a$ without keeping track of every coefficient. In this way this algorithm runs efficiently.

Claim 8. *This algorithm succeeds (and finds a proper square root) with probability approximately 1/2.*

Proof. If $a = b^2$, then $(cx+d)^2 - a = [(cx+d)+b][(cx+d)-b]$. This means that if the GCD found is linear, the value for $cx + d$ that is found is a square root of a.

Since $cx + d$ is a random affine transformation, it permutes the two roots of $x^2 - a$ independently at random. The GCD is linear when exactly one of these roots is a root of $x^{\frac{p-1}{2}} - 1$. Since there are exactly $\frac{p-1}{2}$ roots of $x^{\frac{p-1}{2}} - 1$, the probability the GCD is linear is $\frac{1}{2} - \frac{1}{2p^2}$. $\square$

This algorithm relies on the fact that p is prime. In fact, if we could do this for any general n, we could factor n!

Let's assume that $A(x, n)$ is an (efficient) algorithm for finding the square root of $x \mod n$. This lets us factor n as follows:

1. Pick $y \in \mathbb{Z}_n$ uniformly at random.
2. Let $y' = A(y^2, n)$. If $y = y'$ algorithm fails.
3. Compute $GCD(y - y', n)$. If this is non-trivial, we have factored n.

This algorithm will not work properly if $y = y'$ or when n is a prime power. We can check if n is a prime power easily (as well as if $y|n$), and factor n that way if so.

Claim 9. *This algorithm succeeds (when n is not a prime power) with probability at least 1/2.*

Proof. Since $y^2 = (y')^2 \mod n$, this means $n|(y-y')(y+y')$. Since both y and y' are less than n, the only way that $y - y'$ has trivial GCD with n is when $y' = y$ or $y' = -y$.

However, if n is not a prime power, then $n = \prod_i = 1^k p_i^{e_i}$ for some k primes p_i (and exponents e_i). By the Chinese Remainder theorem, y^2 is uniquely defined by its remainder modulo $p_i^{e_i}$ for each i.

Since y^2 is a square, each of these moduli is a square, and so each have two square roots modulo that prime power. Each possible set of choices of these square roots corresponds to a unique square root of y^2, meaning y had 2^k square roots modulo n.

Since A has no knowledge of which of these square roots we chose, it has a $\frac{2}{2^k} \leq \frac{1}{2}$ probability of returning y or $-y$. This means the algorithm fails with probability less than $\frac{1}{2}$. $\square$