Primality Tests and Factoring with the AKS polynomials

Factoring with funny but exotic algorithms
Robert Erra
LSE WEEK 2015

July 17, 2015

Primality Tests and Factoring with the AKS polynomials

Factoring with funny but exotic algorithms
Robert Erra
LSE WEEK 2015

Introduction

Factoring

Primality Tests

Factoring with AKS polynomial?

Factoring via Modular Square Root

Conclusion

1 Introduction

Introduction

Factoring

Primality Tests

Factoring with AKS polynomial?

Factoring via Modular Square Root

Conclusion

Prime and Composite Numbers

1. *Prime numbers are ubiquitous in modern cryptography and fortunately a lot of probabilistic and deterministic primality tests exist. The most famous is the AKS algorithm that has proved that "Prime is in P", a result that is one of the most important results in the last 30 years in computational number theory*
2. *On the other side, Factoring a large number is a hard problem*
3. *By the way:*
   - *Prime is in P (AKS Theorem)*
   - *Factoring: complexity is still unknown!*

Introduction

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Prime and Composite Numbers

1. *Prime numbers are ubiquitous in modern cryptography and fortunately a lot of probabilistic and deterministic primality tests exist. The most famous is the AKS algorithm that has proved that "Prime is in P", a result that is one of the most important results in the last 30 years in computational number theory*
2. *On the other side, Factoring a large number is a hard problem*
3. *By the way:*
   - *Prime is in P (AKS Theorem)*
   - *Factoring: complexity is still unknown!*

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Prime and Composite Numbers

1. *Prime numbers are ubiquitous in modern cryptography and fortunately a lot of probabilistic and deterministic primality tests exist. The most famous is the AKS algorithm that has proved that "Prime is in P", a result that is one of the most important results in the last 30 years in computational number theory*
2. *On the other side, Factoring a large number is a hard problem*
3. *By the way:*
   - *Prime is in P (AKS Theorem)*
   - *Factoring: complexity is still unknown!*

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Prime and Composite Numbers

1. *Prime numbers are ubiquitous in modern cryptography and fortunately a lot of probabilistic and deterministic primality tests exist. The most famous is the AKS algorithm that has proved that "Prime is in P", a result that is one of the most important results in the last 30 years in computational number theory*
2. *On the other side, Factoring a large number is a hard problem*
3. *By the way:*
   - *Prime is in P (AKS Theorem)*
   - *Factoring: complexity is still unknown!*

Introduction

We propose here to analyze the following question:

If we take a composite number what information can we obtain with (failed) primality tests?

With an objective:

Can we factor a composite number with the results of (some) primality tests?

Introduction

We propose here to analyze the following question:

If we take a composite number what information can we obtain with (failed) primality tests?

With an objective:

Can we factor a composite number with the results of (some) primality tests?

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Our presentation: Work in Progress

1. We will explain how in some cases we can factor a number using some primality tests
2. We will for example explain why Charmichael numbers are easy to factor
3. And we will finish with the presentation of two new (and curious) factorization algorithms
   - One uses the AKS polynomials (but is still folkloric)
   - The other uses the Cipolla polynomials (and we hope it could be efficient for some special numbers)

Introduction

Our presentation: Work in Progress

1. We will explain how in some cases we can factor a number using some primality tests
2. We will for example explain why Charmichael numbers are easy to factor
3. And we will finish with the presentation of two new (and curious) factorization algorithms
   - One uses the AKS polynomials (but is still folkloric)
   - The other uses the Cipolla polynomials (and we hope it could be efficient for some special numbers)

Primality Tests and Factoring with the AKS polynomials

Factoring with funny but exotic algorithms
Robert Erra
LSE WEEK 2015

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Plan

Introduction

Factoring

Primality Tests

Factoring with AKS polynomial?

Factoring via Modular Square Root

Conclusion

2 Factoring

— Some factorisation records: a stagnacy?

Bit-Size	Year	Algorithm
RSA-120 (399 bits)	1993	MQPS
RSA-129 (429 bits)	1994	MPQS
RSA-130 (432 bits)	1996	NFS
RSA-140 (466 bits)	1999	NFS
RSA-155 (512 bits)	1999	NFS
RSA-160 (532 bits)	2003	NFS
RSA-200 (665 bits)	2005	NFS
RSA-768 bits	2010	NFS
RSA-896 bits	2015?	NFS
RSA-1024 bits	2030?	??

Introduction

Factoring

Primality Tests

Factoring with AKS polynomial?

Factoring via Modular Square Root

Conclusion

3. Primality Tests

Deterministic or Probabilistic ?

- Deterministic primality tests:
  1. AKS, Goldwasser-Kilian (GK/ECPP), Atkin-Morain (AM/ECPP)
  2. Expensive!
  3. But we have a *proof* of primality (a prime certificate).
- Probabilistic primality tests:
  1. Fermat, Solovay-Strassen, Miller-Rabin, Muller (*via* Square Modular Roots)
  2. Fast
  3. But they can fail!!!!

Primality Tests and Factoring with the AKS polynomials
Factoring with funny but exotic algorithms
Robert Erra
LSE WEEK 2015
Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Deterministic or Probabilistic ?

- Deterministic primality tests:
  1. AKS, Goldwasser-Kilian (GK/ECPP), Atkin-Morain (AM/ECPP)
  2. Expensive!
  3. But we have a *proof* of primality (a prime certificate).
- Probabilistic primality tests:
  1. Fermat, Solovay-Strassen, Miller-Rabin, Muller (*via* Square Modular Roots)
  2. Fast
  3. But they can fail!!!!

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Deterministic or Probabilistic ?

- Deterministic primality tests:
  1. AKS, Goldwasser-Kilian (GK/ECPP), Atkin-Morain (AM/ECPP)
  2. Expensive!
  3. But we have a *proof* of primality (a prime certificate).
- Probabilistic primality tests:
  1. Fermat, Solovay-Strassen, Miller-Rabin, Muller (*via* Square Modular Roots)
  2. Fast
  3. But they can fail!!!!

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

(Little) Fermat Theorem

Let $n \geq 2$, if for all a coprime with n

$$a^{n-1} \equiv 1 \bmod n$$

then n is prime.

(Little) Fermat Theorem: for polynomials

Let $n \geq 2$, if for all a coprime with n

$$(x+a)^n \equiv x^n + a \bmod n$$

then n is prime.

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

(Little) Fermat Theorem

Let $n \geq 2$, if for all a coprime with n

$$a^{n-1} \equiv 1 \bmod n$$

then n is prime.

(Little) Fermat Theorem: for polynomials

Let $n \geq 2$, if for all a coprime with n

$$(x+a)^n \equiv x^n + a \bmod n$$

then n is prime.

Converse of Little Fermat Theorem

Unfortunately, the converse is false!

$$4^{15} \equiv 1 \bmod 15$$

but 15 is composite.
From [2]: *There exist also infinitely many composite numbers n for which the converse of Fermat's theorem is "as false as possible", for these numbers we have* $a^{n-1} \equiv 1 \bmod n$ *for every a with* $GCD(a, n) = 1$. *Such numbers are called* **Charmichael** *numbers; the smallest is 561.*

Pseudoprimes

Pseudoprimes [2,3,4]

Definition: An odd composite n such that

$$a^{n-1} \equiv 1 \bmod n$$

is called a *pseudoprime* for the base a.

- They are also called Fermat pseudoprimes or liars
- The group of Fermat Liars $F(n)$ is defined as $F(n) = \{a \in \mathbb{Z}_n^* : a^{n-1} \equiv 1 \bmod n\}$
- There are $\Pi_{p|n} GCD(p-1, n-1)$ Fermat liars

Definition: If n is a composite number such that $a^{n-1} \equiv 1 \bmod n$ for all $a \in \mathbb{Z}_n^*$, then n is said to be a Charmichael number.

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Pseudoprimes [2,3,4]

Definition: An odd composite n such that

$$a^{n-1} \equiv 1 \bmod n$$

is called a *pseudoprime* for the base a.

- They are also called Fermat pseudoprimes or liars
- The group of Fermat Liars $F(n)$ is defined as $F(n) = \{a \in \mathbb{Z}_n^* : a^{n-1} \equiv 1 \bmod n\}$
- There are $\Pi_{p|n} GCD(p-1, n-1)$ Fermat liars

Definition: If n is a composite number such that $a^{n-1} \equiv 1 \bmod n$ for all $a \in \mathbb{Z}_n^*$, then n is said to be a Charmichael number.

Pseudoprimes [2,3,4]

Definition: An odd composite n such that

$$a^{n-1} \equiv 1 \bmod n$$

is called a *pseudoprime* for the base a.

- They are also called Fermat pseudoprimes or liars
- The group of Fermat Liars $F(n)$ is defined as $F(n) = \{a \in \mathbb{Z}_n^* : a^{n-1} \equiv 1 \bmod n\}$
- There are $\Pi_{p|n} GCD(p-1, n-1)$ Fermat liars

Definition: If n is a composite number such that $a^{n-1} \equiv 1 \bmod n$ for all $a \in \mathbb{Z}_n^*$, then n is said to be a Charmichael number.

The grandfather of almost all primality tests

Fermat's Test

Algorithm 1 : **Fermat's Test**

Input: n et $T > 0$;
Output: **n prime** or *n composite*;
Begin:
For $i = 1$ **To** T
Choose a_i randomly in $\{2, \cdots, n-1\}$;
If $a_i^{n-1} \neq 1 \bmod n$ **Return** *n composite*;
EndOfFor
Return n prime;
End.

Miller-Rabin Primality Test

- Modular Square Root Problem: let $a \in \mathbb{Z}_p$, solve $x^2 \equiv a \bmod p$
- With $a = 1$ and p prime: 1 et -1 are trivial solutions
- If p is prime, then $x^2 \equiv 1 \bmod p$ can be written $(x-1)(x+1) \equiv 0 \bmod p$
- and so p divides $(x-1)(x+1)$, so $x \equiv \pm 1 \bmod p$
- But, if n is prime >2, with $n - 1 = 2^s d$ (d odd) then:
- $\forall a \in \mathbb{Z}_n^*$
  1. $a^d \equiv 1 \bmod n$
  2. or $a^{2^r d} \equiv -1 \bmod n$ for r such that $0 \leq r \leq s-1$

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Miller-Rabin Primality Test

- If we find an $a \in \mathbb{Z}_n^*$ such that:
  1. $a^d \not\equiv 1 \bmod n$
  2. $a^{2^r d} \not\equiv -1 \bmod n$ for r such that $0 \le r \le s-1$
- Then n is *composite* (not prime!)
- a is a *compositeness witness*
- Probability of detection compositeness after T tests is $> 1 - \frac{1}{4^T}$

Miller-Rabin Primality Test

- If we find an $a \in \mathbb{Z}_n^*$ such that:
  1. $a^d \not\equiv 1 \bmod n$
  2. $a^{2^r d} \not\equiv -1 \bmod n$ for r such that $0 \leq r \leq s-1$
- Then n is *composite* (not prime!)
- a is a *compositeness witness*
- Probability of detection compositeness after T tests is $> 1 - \frac{1}{4^T}$

Miller-Rabin Primality Test

Probabilistic [but with GRH it is deterministic!]

Algorithm 2 : **Miller-Rabin's Test**

Input: $n > 1$;
Output: **n prime** or *n composite*;
Begin:
$n - 1 = 2^s d$;
Repeat For All $a \in [2, \min(n-1, 2(\log n)^2]$
  If $(a^d \not\equiv 1 \bmod n)$ && $(a^{2^r d} \not\equiv -1 \bmod n)$ **For** $r \in [0, s-1]$
  Return *n composite*;
EndOfRepeat;
Return n prime;
End.

It is easy to factor a Charmichael number

How to factor a Charmichael number ?

- Let n be a Charmichael number, so if $GCD(a, n) = 1$ then $a^{n-1} \equiv 1 \bmod n$
- n is a strong pseudoprime for at most 1/4 of all numbers $a < n$
- So, we can find, probabilistically, a number a such that
  1. $a^k \not\equiv 1$ or $-1 \bmod n$
  2. $a^{2k} \equiv 1 \bmod n$
- Let $b = a^k \equiv 1 \bmod n$ then $b^2 \equiv 1 \bmod n$
- So n divides $(b+1)(b-1)$, since $b \neq 1$ *or* $-1 \bmod n$, n can not divide $b-1$ or $b+1$
- So $GCD(b-1, n) \neq 1$ and is a factor of n

Primality Tests and Factoring with the AKS polynomials
Factoring with funny but exotic algorithms
Robert Erra
LSE WEEK 2015
Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

It is easy to factor a Charmichael number

How to factor a Charmichael number ?

- Let n be a Charmichael number, so if $GCD(a, n) = 1$ then $a^{n-1} \equiv 1 \bmod n$
- n is a strong pseudoprime for at most 1/4 of all numbers $a < n$
- So, we can find, probabilistically, a number a such that
  1. $a^k \not\equiv 1$ or $-1 \bmod n$
  2. $a^{2k} \equiv 1 \bmod n$
- Let $b = a^k \equiv 1 \bmod n$ then $b^2 \equiv 1 \bmod n$
- So n divides $(b+1)(b-1)$, since $b \neq 1$ *or* $-1 \bmod n$, n can not divide $b-1$ or $b+1$
- So $GCD(b-1, n) \neq 1$ and is a factor of n

It is easy to factor a Charmichael number

How to factor a Charmichael number ?

- Let n be a Charmichael number, so if $GCD(a, n) = 1$ then $a^{n-1} \equiv 1 \bmod n$
- n is a strong pseudoprime for at most 1/4 of all numbers $a < n$
- So, we can find, probabilistically, a number a such that
  1. $a^k \not\equiv 1$ or $-1 \bmod n$
  2. $a^{2k} \equiv 1 \bmod n$
- Let $b = a^k \equiv 1 \bmod n$ then $b^2 \equiv 1 \bmod n$
- So n divides $(b+1)(b-1)$, since $b \neq 1$ *or* $-1 \bmod n$, n can not divide $b-1$ or $b+1$
- So $GCD(b-1, n) \neq 1$ and is a factor of n

The AKS primality test [1]

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Also known as Agrawal–Kayal–Saxena primality test and cyclotomic AKS test

. . . is a deterministic primality-proving algorithm created and published by Manindra Agrawal, Neeraj Kayal, and Nitin Saxena, computer scientists at the Indian Institute of Technology Kanpur, on August 6, 2002, in a paper titled "PRIMES is in P".
The algorithm determines whether a number is prime or composite within polynomial time. The authors received the 2006 Godel Prize and the 2006 Fulkerson Prize for this work.

The AKS primality test [1]

Primality AKS Test in a nutshell

1. An integer $n > 2$ is prime if and only if $(x + a)^n \equiv x^n + a \bmod n$ holds for all a coprime with n
2. $C_n^k \equiv 0 \bmod n$ for all $0 < k < n$ if and only if n is a prime (Expensive!)
3. So AKS uses $(x + a)^n \equiv \ \bmod (n, x^r - 1)$ with r a small integer
4. If n is prime $(x + a)^n \equiv x^n + a \bmod (n, x^r - 1)$
5. Proof of correctness for AKS: show that there exists a suitably small r and suitably small set of integers A such that, if the congruence holds for all such a in A, then n must be prime
6. Complexity: $O((\log(n)^{11.5})$ but lot of improvements from 2002 to now!

Introduction

Factoring

Primality Tests

Factoring with AKS polynomial?

Factoring via Modular Square Root

Conclusion

4 Factoring with AKS polynomial?

Factoring with AKS polynomial

Well, our algorithm is simple.

Algorithm 3 : **Factoring with AKS polynomial**

Input: $n > 2$ composite with $n = p\,q \cdots$ with $p < q$;

Output: **the smallest factor of n**;

Begin:

Compute $R = [\sqrt{n}]$

Comment: Square root rounded to the nearest integer

Compute $f(x) = (x + a)^n \bmod (n, x^R)$

Comment: Use Fast Polynomial Modular Exponentiation

Return $f(x) = 1 + qx^p + \cdots$

End.

Factoring with AKS polynomial

Example.

- $n = 253 = 11 \times 23$
- $R = 15$ (Square root rounded to the nearest integer)
- $f(x)$=PolynomialRemainder$[(x+1)^n, x^R, x] =$ $35011874485950604011000x^{14} + 20423593450137852333975x^{13} + 1101687613492913196 75x^{12} + 546291378591527 2050x^{11} + 247292393601102850x^{10} + 1013493416397962 5x^9 + 372303703982925x^8 + 12107437527900x^7 + 343125759900x^6 + 8301429675x^5 + 166695375x^4 + 2667126x^3 + 31878x^2 + 253x + 1$
- $f(x) \bmod n = 1 + 23x^{11}$

We can prove it but it is a folkloric method both for time and space complexity! So let us try heuristic variants.

Factoring with AKS polynomial

Introduction

Factoring

Primality Tests

Factoring with AKS polynomial?

Factoring via Modular Square Root

Conclusion

This algorithm is deterministic, but expensive! So . . .

- We can use the following trick: during the computation of $f(x) = (x+a)^n \bmod (n, x^R)$ we:
  1. Define $L_i = CoefficientList((x+a)^i \bmod (n, x^R), x)$ [Comment: limited by your RAM]
  2. Compute $GCD(L_i, n)$
- Or we
  1. Choose (randomly?) R a very low value
  2. We use again $f(x) = (x+a)^n \bmod (n, x^R)$
  3. Define $L_i = CoefficientList((x+a)^i \bmod (n, x^R), x)$ with i a low value
  4. Compute $GCD(L_i, n)$

Factoring with AKS polynomial

Why does this (folkloric, till we will have found an efficient variant) "algorithm" work?

- $n = 15 = 3 \times 5$
- $C_{15}^{3} = 5 \times 7 \times 13$
- So $C_{15}^{3} \equiv 5 \bmod 15$
- And $(x + a)^n = \sum_{i=0}^{n} C_n^i x^i a^{n-i}$
- *Funny property*: Let R be fixed, if the algorithm returns 0 then there is no factor of $n <$ R

Plan

5 Factoring via Modular Square Root

Primality Tests
and
Factoring with the
AKS polynomials

Factoring with funny but exotic algorithms
Robert Erra
LSE WEEK 2015

Introduction

Factoring

Primality Tests

Factoring with AKS polynomial?

Factoring via Modular Square Root

Conclusion

Computing a modular square root

Algorithm 4 : **Cipolla algorithm**

Input: p prime and a with LegendreSymbol$(a/p) = 1$;

Output: r **such that** $r^2 \equiv a$ **mod** p;

Begin:

Choose a t such that LegendreSymbol$(t^2 - 4a, p) = -1$

Comment: $t^2 - 4a$ is a quadratic nonresidue modulo p

Compute $r = x^{(p+1)/2} \bmod (p, x^2 - tx + a)$

Comment: Use Fast Polynomial Modular Exponentiation

Return y (it is a integer)

End.

Algorithm 5 : **Factoring via modular "pseudosquare" root**

Input: n composite;

Output: **1 or n or a factor of n**

Begin:

Choose a t such that LegendreSymbol$(t^2 - 4a, n) = -1$

Compute $r(x) = x^{(p+1)/2} \bmod (n, x^2 - tx + a)$

Commment: Use Fast Polynomial Modular Exponentiation

L(r)=CoefficientList(r(x),x)

Return $GCD(L, n)$

End.

Introduction
Factoring
Primality Tests
Factoring with AKS polynomial?
Factoring via Modular Square Root
Conclusion

Factoring via Modular Square Root

Example.

- $n = 649 = 11 \times 59$
- $a = 2, \cdots 16, 19, 20, 22, \cdots, 28, 31 \cdots$ it works
- For $a = 2$ we have $r(x) = 352 + 425x$
- $L(r) = \{352, 425\}$
- $GCD(L(r), n) = \{11, 1\}$. Factored!
- For $n = 649$ there are 383 value of a giving a correct factorization. Let's call them *good liars*
- What are the special numbers for which algorithm would be efficient ?
- Each iteration is fast (quite) but I don't know the exact time complexity (WIP) of the whole algorithm.
- We need to compute the exact number of *good liars*

Plan

6 Conclusion

Introduction

Factoring

Primality Tests

Factoring with AKS polynomial?

Factoring via Modular Square Root

Conclusion

Introduction

Factoring

Primality Tests

Factoring with AKS polynomial?

Factoring via Modular Square Root

Conclusion

So what?

- *Yes we can factor numbers with some primality tests*
- *We have to compute the exact complexity of the algorithms we have presented (WIP)*
- *We have to understand better when they works and so when they don't (WIP)*
- *Could Primality and Factoring be problems more "intricate" than expected?*
- *Open Conjecture:* **Is Factoring in P** *?*

Some lectures:

You can read for fun and profit:

1. https://en.wikipedia.org/wiki/AKS_primality_test
2. Bach and Shallit: *Algorithmic Number Theory, vol 1., Efficient Algorithms*, The MIT Press.
3. Crandall and Pomerance, *Prime Numbers, a Computational perspective*, Springer.
4. S. Muller, On probable Prime Testing and the computation of Square Roots mod n, ANTS-IV, 2000, Springer.