The 14th IEEE Conference on
Advanced and Trusted Computing (ATC 2017)

August 4-8, 2017, San Francisco Bay Area, USA

IEEE CyberTrust workshop

Title: Rapid Factorization of Composite Primes - An alternative to the sieve method

Author: André Brisson

Summary:

Trusted computing is based on PKI and a 2048-bit RSA public and private key pair that is created randomly on the chip at time of manufacture and cannot be changed. SHA for which Google just demonstrated a collision break is a public key thumbprint (much smaller) that is a workhorse in secure computing. Trusted Computing relies on the secrecy of a prime number composite which is the product of two prime factors. Reversing this is factorization. The security underpinning is that prime number composites can only be attacked by brute force and that the work space is so large that this kind of attack is not feasible in any usable time frame.

NSA announced in August 2015, that it planned to move to a new cipher suite that is resistant to quantum attacks. Integer Factorization Cryptography and Elliptical Curve Cryptography are readily breakable by quantum computing attacks. NSA is completing quantum computing facilities.

The top public key recommended strength by NIST up to 2031 and beyond is 256-bits. When we are presented key sizes for integer factorization cryptography like RSA it is easy to get a false sense of confidence. It seems sufficient to have 2048 bit keys but a 2048-bit is 258 characters long and yet offers only 112 bits of security. That represents a lot of processing for minimal protection.

This paper examines other vulnerabilities of cryptography that relies on public key pairs and prime composites.

The 14th IEEE Conference on Advanced and Trusted Computing (ATC 2017)

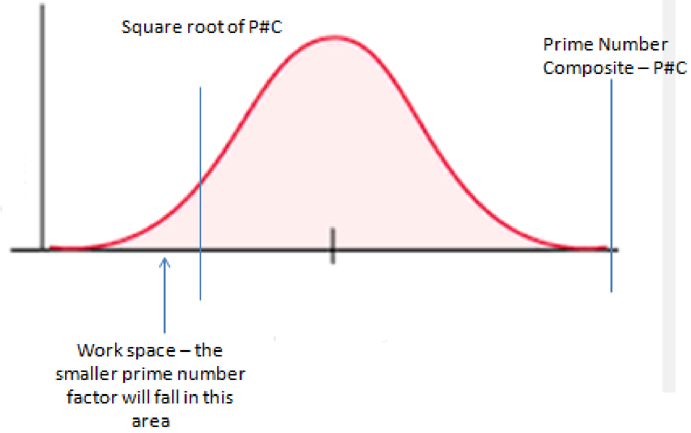
August 4-8, 2017, San Francisco Bay Area, USA

If one looks at a Cartesian plane and a rendition of a normal curve, the following elements are readily identifiable:

1. We can draw a normal curve with one endpoint on the X axis being equal to the Prime Number Composite (P#C).

2. We can take the square root of the prime number composite and this becomes the mean.

3. We know that one of each of the two factors we are looking for will reside somewhere on either side of the mean.


Square root of P#C
Prime Number Composite – P#C
Work space – the smaller prime number factor will fall in this area

4. Looking at the graph we will search for the smaller prime number factor of the prime number composite in the smaller workspace area. Our first step is to bisect the smaller area of the curve along the X axis. In this example it is labeled bisection.

5. We rapidly test to see whether the point of bisection is a whole number. If it is (unlikely) then we have found one of our factors. If this was the case then we simply have to divide this number into the prime number composite to get the other factor.

6. If this is not the case then the challenge now is to decide which side of this bisection we jump to in order to make the next bisection. We take two points on either side of the previous bisection and divide the larger point by the smaller point and take the sixth derivative of this value. The more volatile remainder will be closer to the solution so the next bisection will be on that side. Repeat steps until the smaller prime number factor is found.

A simple example of a 128-bit prime number composite is created by multiplying the following two prime number values together:
11,114,329 times 11,114,423 = 123,529,353,867,167

A simple brute force attack would test every single number from 1 to 123,529,353,867,167 including non-prime numbers to find a number that divides evenly into our prime number composite. However, testing over 123 trillion numbers is a BIG task. Doing one test per second of all possibilities it would take over 3,900 centuries.

Taking the square root value of the prime number composite will bisect the range of possible values (123,529,353,867,167) in a manner that insures that one prime factor will fall **BELOW the square root value and one prime factor will fall ABOVE this value. This exponentially reduces the workspace to search.**

Factor 1 (11,114,329) < square root of the example prime number composite - 11,114,375 < Factor 2 (11,114,423)

By solving for the smaller prime factor value we are reducing our “brute force workspace” in this example from 123,529,353,867,167 calculations to 11,114,375 calculations. The reduced attack space is one ten millionth the size of the prime number composite.

In a simple brute force attack we would still be testing non-prime numbers which cannot be one of the factors. There is list after list of prime numbers values that have already been calculated and using only prime numbers in a dictionary attack further reduces the number of searches.

Using https://primes.utm.edu/lists/small/millions we see that one of the prime number factors to break our example 128 bit value will be found in their list of the first million primes. This reduces the workspace search to only a million calculations. The routine would simply divide every prime number value into our prime number composite until one of the numbers divides evenly.

Given that prime numbers prime numbers up to 10^18 have been calculated this dictionary attack might be valid through the entire life expectancy and utility of RSA Integer Factorization as determined by the NIST.

DEMO

11,114,329 times 11,114,423 = 123529353867167

104729 times 81343 = 8518971047

100057 times 96293 = 9634788701

IFC is used in many place within networks and appear in different sizes i.e. hashes, signatures (Google broke SHA with collision attack)

Optimize: 64-bit, samples of comparisons after bisection, run on quantum computers etc.

It was necessary to write a fictional book so that the above techniques would not be lost:
In Denial Code Red.

http://www.wnlabs.com/products/InDenialCodeRedDirect.php

END