Dynamic Identity Verification and Authentication

In Dynamic Distributed Key Infrastructures (DIVA and DDKI)

André Brisson
Whitenoise Laboratories Canada Inc.
#701 – 1736 W. 10th Ave.
Vancouver, British Columbia Canada V6J 2A6
abrisson@wnlabs.com

Abstract—**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.**

Keywords—RSA; factorization;

I. INTRODUCTION

"In August 2015, NSA announced that it planned to transition in the not distant future to a new cipher suite that is resistant to quantum attacks." Wikipedia

Ironically, while ECC may be easier to break with quantum attacks, RSA Integer Factorization cryptography is additionally challenged because of high processing requirements with minimal security strength for the effort. NIST states that the security provided by 2048 bit key is only 112 bits cryptographic strength[1].

The following two rapid factorization approaches we examine both rely on the exponential reduction in the size of searchable workspace.

It was initially noted that if one takes the sixth derivative of two points on a normal curve, the large point divided by the smaller point, that the closer you are to the solution the more volatile the remainder becomes. This then becomes the primary criteria in a decision tree in deciding which way to jump if one is deploying a bisection method to try to locate a prime number factor.

On a basic system it is possible to factor smaller prime number composites in under a second. Extending this technique will make factoring of large prime number composites an easy task that holds out great possibilities for the manipulation of primes and composite primes in general number theory.

Further research discovered that it may now possible to use a Prime Number Dictionary attack. This may create an existential threat to Trusted Computing.

II. RAPID FACTORIZATION ARCHITECTURE

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).

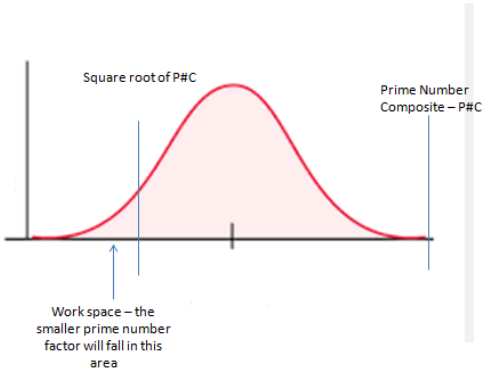
2. We can take the square root of the prime number composite and this becomes the mean.

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

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.



This process can be accelerated with a 64-bit application as well as parallel, serialized and multi-thread processing. A demonstration of this can be seen on YouTube [2]:

While considering optimization it was recognized that effective use of a prime number dictionary attack could augment or in some cases replace the existing factorial utility that exploits Newton's bisection theory and a mathematical test.

III. PRIME NUMBER DICTIONARY

"…a **dictionary attack** is a technique for defeating a cipher or authentication mechanism by trying to determine its decryption key or passphrase by trying hundreds or sometimes millions of likely possibilities, such as words in a dictionary." Wikipedia

"It is possible to achieve a time-space tradeoff by pre-computing a list of dictionary words (primes), and storing these in a database. This requires a considerable amount of preparation time, but allows the actual attack to be executed faster." Wikipedia

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. When we have discovered a factor because it divides evenly into the prime number composite then we have also identified the other factor. 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.

We can reduce the work search space further by using the first factorization utility introduced that uses bisection and a mathematical test for volatility.

Prime numbers fall along the spectrum of the number line. As the Whitenoise Factorial Utility makes each bisection and test and jump, the range of possible prime numbers in our dictionary that contain the small factor is being bisected as well and the number of possible dictionary primes needed to be searched keeps reducing dramatically by searching only the subsequent bisected area. We start with the bisection method and then conduct the dictionary attack on the remaining searchable workspace.

IV. HASHES AND SIGNATURES

Networks are complicated environments and integer factorization cryptography like RSA is used for other security functions like certificates, signatures and hashes. Different security functions use different key strengths out of necessity because of processing effort in different contexts.

The following code is used for RSA hashes. You can see that in creating a prime number composite they are choosing prime numbers between 1 and 255. Hackers are adept at compromising systems by attacking the weakest points.

From Wikipedia

"

```
*RSA has function reference implementation.
/**
*Generates an RSA hash
* https://en.wikipedia.org/wiki/RSA (cryptosystem) #A working example
* @returns {array} Result of RSA generation
*/
// generate values
var p = random prime (1,255), // 8 bit
q = random prime (1,255), // 8 bit
n = p * q,
t = (p-1) * (q – 1), //totient as φ(n) = (p − 1)(q − 1)
```

```
e = random prime (1,t)
d = modular multiplicitive inverse (e,t) ;
return {
n : n, // public key (part I)
e:e, // public key (part II)
d: d // private key
    };
};"
```

V. CONCLUSION

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.

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.

Google recently demonstrated breaking SHA. Why they finally expended the effort at this time is interesting when SHA has been known to be vulnerable to collision attacks for a decade.

Extending the two techniques presented herein may offer a contribution to the manipulation of primes in general number theory simplify as well as conducting attacks against Trusted Computing.

VI. ACKNOWLEDGEMENT

Stephen Lawrence Boren, Vancouver, British Columbia, Canada

VII. REFERENCES

[1] http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

[2] André Brisson Factorization of a prime number composite https://www.youtube.com/watch?v=GwkwgR_78dQ&feature=youtu.be

[3] André Brisson "Rapid Factorization of Semi Primes," http://www.wnlabs.com/pdf/Rapid_Factorization_of_semiprimes.pdf