Is a prime number dictionary attack a practical way to factor semiprimes?

whitenoise
technologies

Factoring semiprimes

This paper examines dictionary attacks for factoring pki semiprimes.

"The RSA Factoring Challenge was a challenge put forward by RSA Laboratories on March 18, 1991 to encourage research into computational number theory and the practical difficulty of factoring large integers and cracking RSA keys used in cryptography…" wiki

This is critical for cryptological uses which rely on the secrecy of the prime factors of a modulus. This builds on an alternative to sieve techniques that leverages Newton's bisection theory and an insight into testing the behavior of primes.

Given the significance of what the capability implies to the security of the Internet and all communications it merits continued investigation and continued experimentation and adaptation of our networks. We already know the WN factorial utility works.

What is a dictionary attack?

A simple dictionary attack against smaller semiprimes would not have been feasible just a couple of decades ago. Determining primality chewed up a lot of computer computational resources and the number of pre-calculated prime numbers was much smaller. That is not the case today. Computers are significantly faster. Many prime numbers have been determined so making a dictionary that covers all possibilities for prevalently used modulus strengths is easy to construct.

https://en.wikipedia.org/wiki/Dictionary_attack#Pre-computed_dictionary_attack.2FRainbow_table_attack

"In cryptanalysis and computer security, 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." wiki

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

The above example looks at known bad passwords (i.e. password, 123456 etc.). A prime number dictionary will look at available, **pre-computed prime numbers.**

Prime number dictionary attack

This paper started by anticipating possible questions related to the Whitenoise factoring technique to factor semiprimes that was shown publicly the first time at the Telecom Council of Silicon Valley Deep Dive on Security in the Snowden Era at Microsoft on October 28, 2015.

The Whitenoise Large Factor Utility to factor semiprimes can be further optimized in several ways:

- Create a 64-bit and greater utility instead of the 32-bit demonstration
- Parallel processing and serializing computers is a linear way of decreasing time to break semiprimes
- Multi-core processing will speed up the factoring

While considering optimization of this utility 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.

Additionally, there are a series of simple, logical questions that one can ask or test for to radically reduce the attack work area and the time needed to factor a semiprime.

This paper is defining attack work area as the range of values within which we must search to find the smaller of the two prime factors of a semiprime.

The entire security premise of public key frameworks, and the one-way function which is called into question, is that it is easy to multiply two prime numbers together to get a semiprime but it was considered infeasible or impossible to reverse the process and discover the two prime number factors from the semiprime (modulus) value. This value is central to their one-way function and it must not be reversible.

Attack work area

Let's take a look at attack work area:

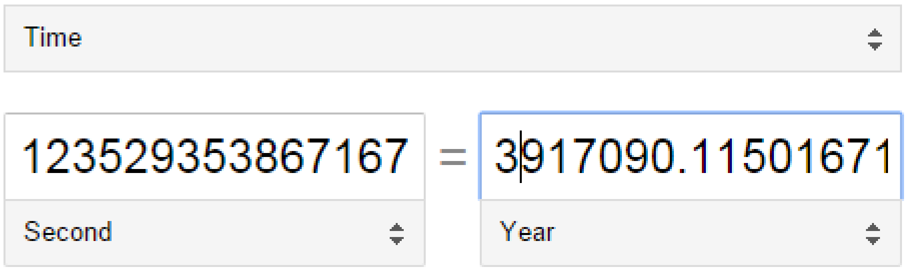
This is an example of a 128-bit semiprime created by multiplying the following two prime number values together:

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

By adding commas we see that the bi-prime value is 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 see which number divides evenly into our semiprime value. When we have discovered the small factor because it divides evenly into the semiprime value then we have also identified the other, larger 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.



123529353867167 = 3917090.11501671

The Telecom Council of Silicon Valley saw such a semiprime being broken in one second.

Radical reduction of work space

This is the most important step.

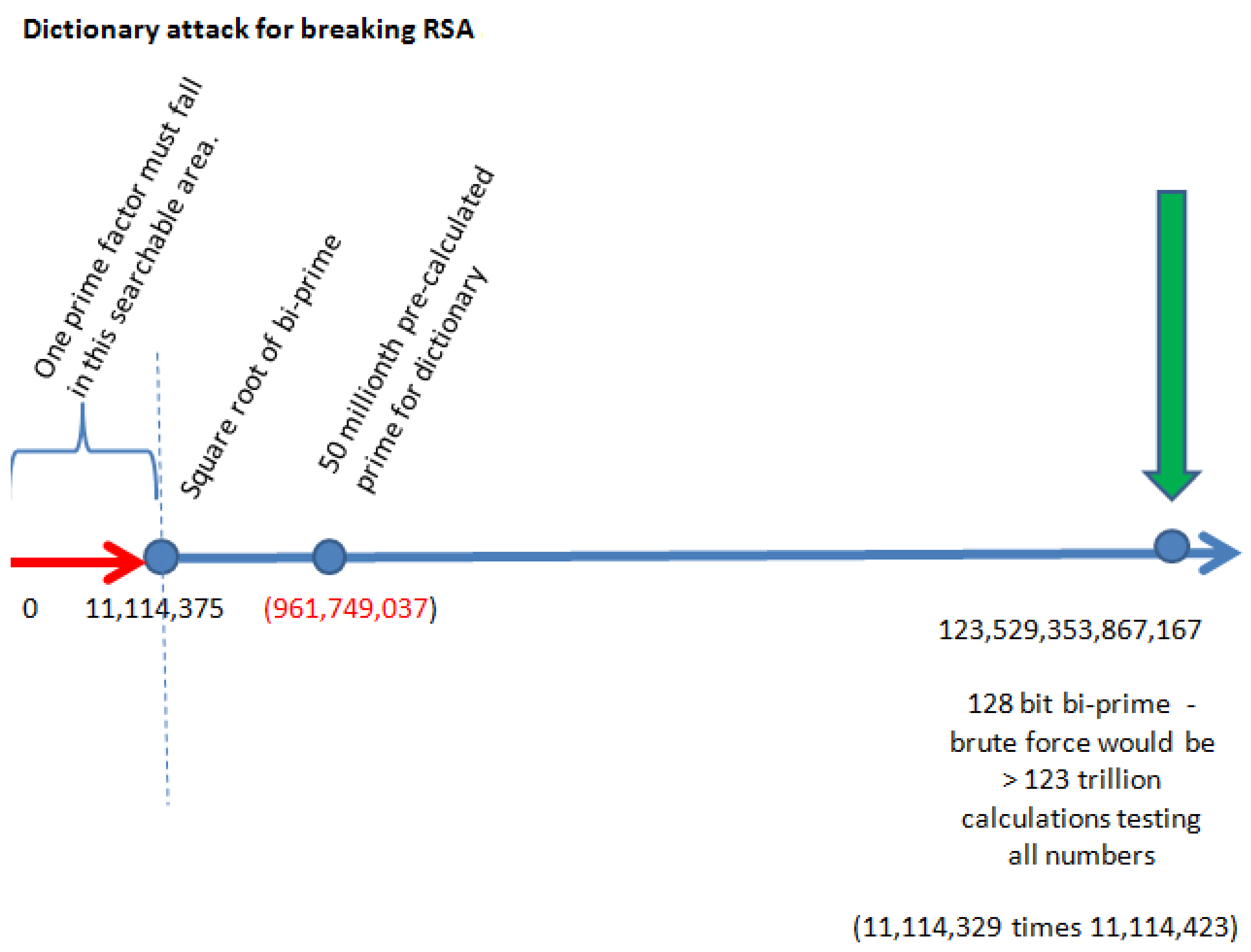
If we take the square root value of the semiprime then we **KNOW** that this will bisect the range of possible values (123,529,353,867,167) in a manner that insures that one prime factor will fall **BELOW** the square root value and one prime factor will fall **ABOVE** this value. This is illustrated below where we see the semiprime square root value falls between the two prime numbers we multiplied together.

In our example:

The square root of 123,529,353,867,167 equals 11,114,376 and we see that:

Factor 1 (11,114,329) < square root of the bi-prime key - 11,114,375 < Factor 2 (11,114,423)

The above example shows that the square root of a semiprime bisects the number line in the fashion desired.


Dictionary attack for breaking RSA
One prime factor must fall in this searchable area.
Square root of bi-prime
50 millionth pre-calculated prime for dictionary
0
11,114,375
(961,749,037)
123,529,353,867,167
128 bit bi-prime - brute force would be > 123 trillion calculations testing all numbers
(11,114,329 times 11,114,423)

This means that one of the prime number factor values we are looking for falls below the square root of the semiprime.

By solving for the smaller prime factor value we are reducing our "brute force testable universe" from 123,529,353,867,167 calculations to 11,114,376 calculations. This is a small fraction of the size of the range that would otherwise need to be tested.

That is a significant reduction in work effort to factor semiprimes.

Reducing work effort with prime number dictionary

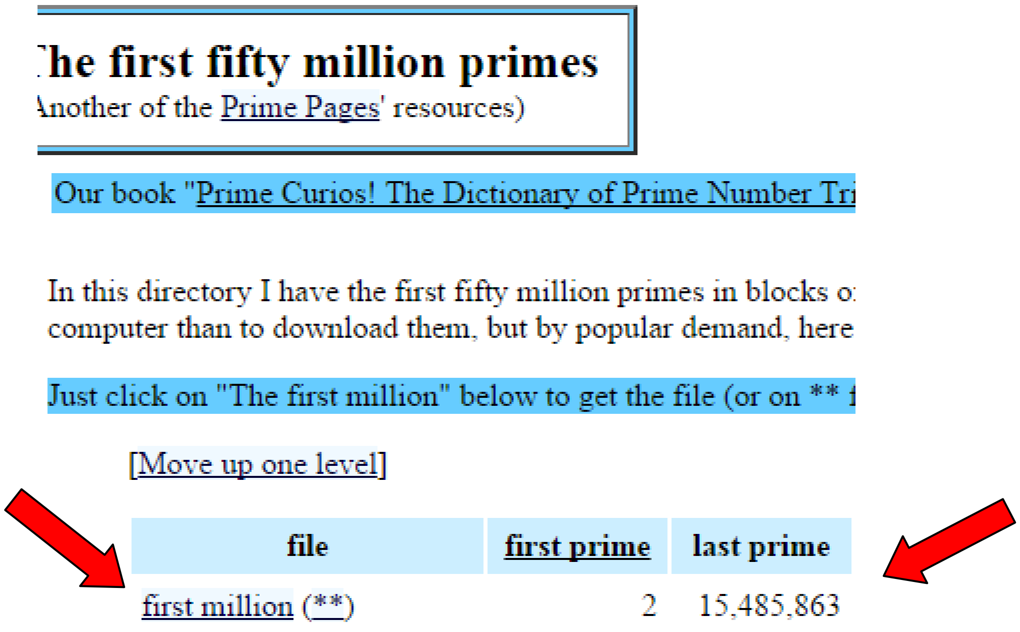
In our example we are looking for a prime number value that falls in the range of 0 to 11,114,376.

However, in a simple brute force attack we would still be testing non-prime numbers which cannot be one of the factors. This is unnecessary.

There is list after list of prime numbers values that have already been calculated and we can simply use only the calculated prime numbers in a dictionary attack to further reduce our searchable area.

In our example we want to search only the prime numbers between 0 and 11,114,375 which is the square root of our semi-prime.

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.



'he first fifty million primes

\nother of the Prime Pages' resources)

Our book "Prime Curios! The Dictionary of Prime Number Tri

In this directory I have the first fifty million primes in blocks o
computer than to download them, but by popular demand, here

Just click on "The first million" below to get the file (or on ** f

[Move up one level]

file	first prime	last prime
first million (**)	2	15,485,863

We are looking for a prime number that is smaller than the square root of our example 128-bit semiprime. The value in this example is 11,114,375. In the graphic above we see that the value falls within the first list of the first 1,000,000 primes.

Since these prime numbers are already calculated, it would be a simple task to incorporate them one time into a dictionary attack application. The routine would simply divide every prime number value into our semiprime until one of the numbers divides evenly.

Given the speed of even basic computers solving for one of 1,000,000 possibilities is not a daunting task.

Reducing the testable space further

128 bit semiprime is created by multiplying prime number pairs of the approximate corresponding byte lengths.

All pairs for a 128-bit semi-prime are listed below.

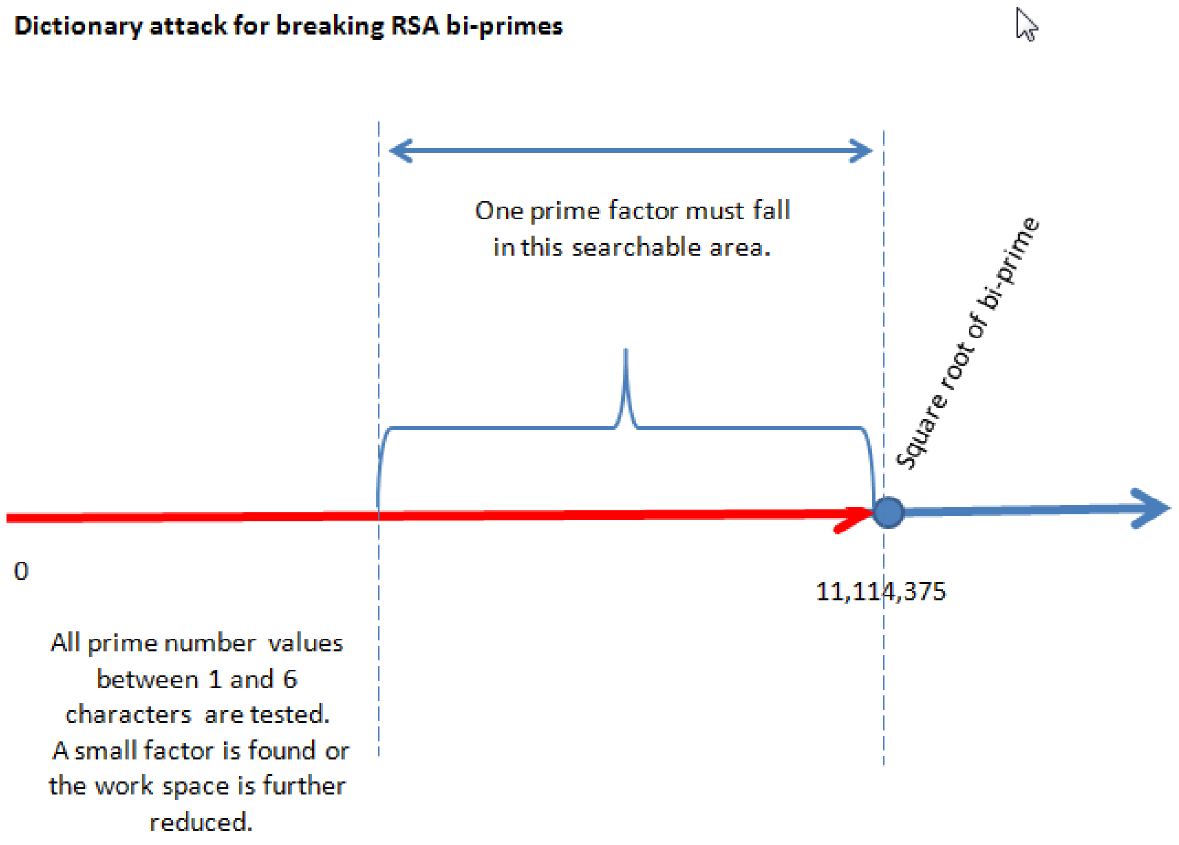
- 8 digit number multiplied by an 8 digit number
- 7 digit number multiplied by a 9 digit number
- 6 digit number multiplied by a 10 digit number
- 5 digit number multiplied by a 11 digit number –
- 4 digit number multiplied by a 12 digit number – etc.

- 3 digit number multiplied by a 13 digit number – there are (aprox) 45 three digit prime numbers
- 2 digit number multiplied by a 14 digit number – there are 21 two digit prime numbers
- 1 digit number multiplied by a 16 digit number – there are 4 one digit prime numbers

There are only about 35,000 prime numbers that are between 1 and 6 characters long.

When we exclude the area of one of the smaller digit primes we also exclude its large factor pair as a searchable area. For example – excluding a 1 digit prime factor means we don't have to search in the 16 digit prime space. (It is also well above the square root of the bi-prime and we are searching for the smaller prime number factor.)

Using the dictionary attack method above, we can rapidly test for one of the factors being between 1 and 6 digits. That is only 35,000 calculations. If a factor is not found within that range then we have further reduced our work space of where we need to search for a prime number factor.


Dictionary attack for breaking RSA bi-primes
One prime factor must fall in this searchable area.
Square root of bi-prime
0
11,114,375
All prime number values between 1 and 6 characters are tested. A small factor is found or the work space is further reduced.

Dictionary space and the bisection process

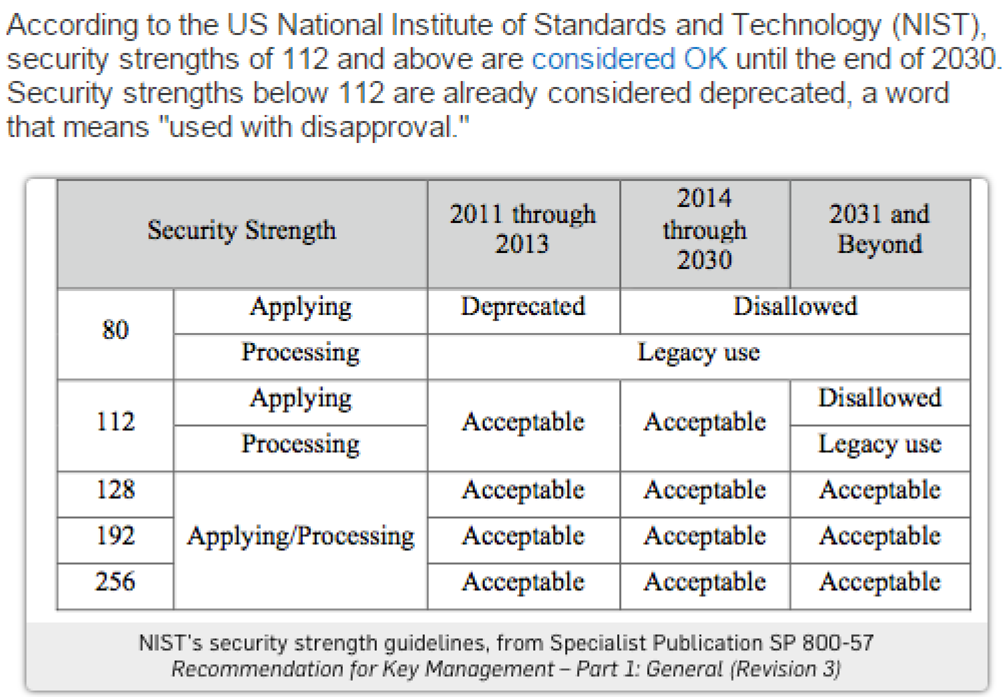
Prime numbers fall along the spectrum of the number line. As the Whitenoise Factorial Utility makes each bisection and test and jump, the range of possible prime numbers in our dictionary that contain the small factor is being bisected as well and the number of possible dictionary primes needed to be searched is dramatically reducing. This keeps reducing calculation times by searching only in the possible ranges where the factor resides.

Conclusion and NIST recommended security strength

The Whitenoise Large Factorial Utility is effective. It can be optimized.

Starting any widely available sieve method utility with the square root bisection method may dramatically speed up those applications by significantly reducing work space.

The example used in this paper shows breaking a 128-bit semiprime since that is a security strength that the NIST says will be sufficient security strengths until 2031 and beyond.



According to the US National Institute of Standards and Technology (NIST), security strengths of 112 and above are considered OK until the end of 2030. Security strengths below 112 are already considered deprecated, a word that means "used with disapproval."

Security Strength		2011 through 2013	2014 through 2030	2031 and Beyond
80	Applying	Deprecated	Disallowed	
	Processing	Legacy use		
112	Applying	Acceptable	Acceptable	Disallowed
	Processing			Legacy use
128	Applying/Processing	Acceptable	Acceptable	Acceptable
192		Acceptable	Acceptable	Acceptable
256		Acceptable	Acceptable	Acceptable

NIST's security strength guidelines, from Specialist Publication SP 800-57 *Recommendation for Key Management – Part 1: General (Revision 3)*

NIST looks to be dangerously wrong.

The concepts presented merit further testing. Scaling these concepts to radically reduce work space to break 256 bit – the top recommended key strength beyond 2031 - should not be difficult and wouldn't take any mathematical skills. 256-bit is the real practical ceiling of public security strengths currently being used in public key systems today because increasing key strengths turns overhead issues into a nightmare.

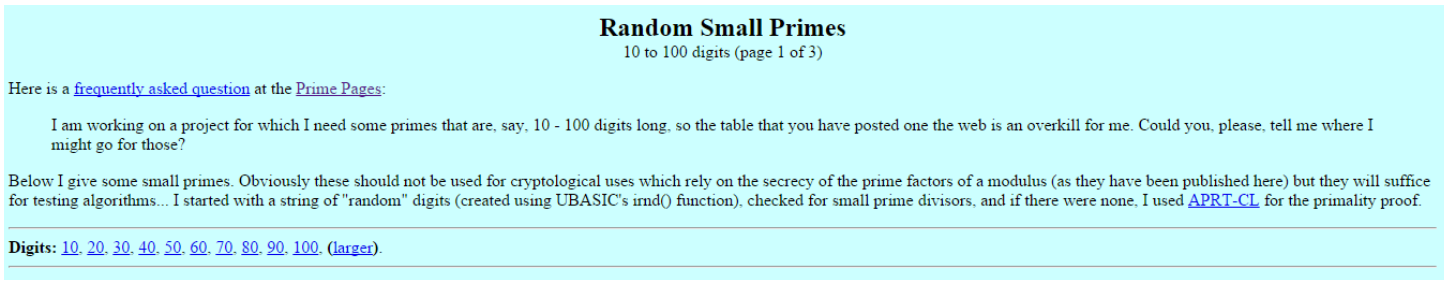
Adding dictionary techniques to the Whitenoise semiprime factoring insight (which is sufficient on its own) might compel mathematicians and computer scientists to investigate whether the security of RSA PKI is insufficient as a standalone security framework.

NIST says 128-bit security is sufficient through 2031. We saw in the example above those semiprime factors for this fall within the first one-million pre-calculated prime numbers.

The top public key recommended strength by NIST up to 2031 and beyond is 256-bits. That is a 32 digit key.

The square root of a 256-bit, 32-digit semiprime is 16 characters long and falls easily within the number of pre-computed prime numbers so making a dictionary for a similar attack on 256-bit semiprimes is already doable without any further calculations. The proportional attack time saving is similar in relative magnitude in our example and easily addressed with parallel computing.

This site has pre-calculated primes up to 100 digits.



Random Small Primes

10 to 100 digits (page 1 of 3)

Here is a frequently asked question at the Prime Pages:

> I am working on a project for which I need some primes that are, say, 10 - 100 digits long, so the table that you have posted one the web is an overkill for me. Could you, please, tell me where I might go for those?

Below I give some small primes. Obviously these should not be used for cryptological uses which rely on the secrecy of the prime factors of a modulus (as they have been published here) but they will suffice for testing algorithms... I started with a string of "random" digits (created using UBASIC's irnd() function), checked for small prime divisors, and if there were none, I used APRT-CL for the primality proof.

Digits: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, (larger).

https://primes.utm.edu/notes/faq/LongestList.html

“Perhaps the longest lists ever calculated (but not all stored) are those corresponding to the maximal prime gape (and twin prime constant) projects. See Nicely's lists. At the time I last updated this page, these projects had found all the primes up to 10^{18} and are available.”

Given that all the prime numbers up to 10^{18} have been calculated, and knowing that attack work area is first identified by taking the square root of the semiprime means that dictionary attacks can be constructed to factor semiprimes up to 10^{36} bytes in length.

“If you want an even longer list, run a sieve program on your machine. Folks quite regularly resieve to find all the primes up to 1,000,000,000,000; this should take well less than a minute.”

As public key strengths ramp up key strength to contend with this vulnerability the computational effort is exploding and more and more cheap or cheaper components or devices are being excluded from the universe that can use this stronger public key capability. It is a losing battle. PKI just does not scale well.

A 1024-bit key is 128 digits long. The largest prime calculated to date is over 17 million characters long.

Addendum

Whitenoise Factorial Utility

Any factorial utility can start with taking the square root of a semiprime to dramatically reduce our work area. Thereafter WNLs deploy Newton’s bisection theory and test to ask and answer: “On which side of the next bisection should we keep looking?”

There is a very simple mathematical sample test and comparison to answer that question.

http://www.wnlabs.com/pdf/How_to_Break_RSA_semiprimes.pdf

Should the utility jump to the wrong side, it could keep looking in the wrong range indefinitely unless one backtracks to the last good bisection and jump and reverses that next choice. This is simply an optimization test addition.

Use of the prime number dictionary approach within certain bisected ranges can test and provide confirmation of jumping in the right direction as opposed to, or in addition to, the simple mathematical sample test and comparison.

Comparing the volatility of a bisection test against a previous calculation will indicate relative volatility and whether the utility is proceeding in the correct direction.

Forward secrecy

This addendum looks at the difficulty of breaking a 512 bit key with this technique. A 512 bit Diffie-Hellman is the strength of key analyzed by INRIA Paris-Rocquencourt, INRIA Nancy-Grand Est, CNRS, and Université de Lorraine Microsoft Research, University of Pennsylvania § Johns Hopkins, and University of Michigan when they warned of vulnerabilities and discounted forward secrecy.

Assessment of forward security by INRIA Paris-Rocquencourt, INRIA Nancy-Grand Est, CNRS, and Université de Lorraine Microsoft Research, University of Pennsylvania, Johns Hopkins and the University of Michigan evaluated the strength of 512 DH in forward secrecy.

They are sounding a loud warning.

In layman's language: http://tonyarcieri.com/imperfect-forward-secrecy-the-coming-cryptocalypse

Scientific American article on standards and manipulation

In the article Ari Juels, chief scientist of computer storage provider EMC's RSA security division, tells us that the NIST Special Publication 800-90 cryptography standard includes four different algorithms—called "deterministic random bit generators," or DRBGs—for encoding data. Some may speculate that the algorithm included at the NSA's behest—Dual Elliptic Curve Deterministic Random Bit Generation (Dual_EC_DRBG) may have been the Certicom/Blackberry contribution to security in order to get certification to enter the game. Knowing that the value proposition of elliptical curve cryptography is offering a technique with lower overhead and a smaller footprint, and knowing that they led the smart phone revolution, can make one question whether rapid US approval was the quid pro quo for certification to enter the market quickly in a space (mobile devices) that RSA could not deploy well.

Security is always only as good as the weakest link in the security food chain. From there the bad guys piggy back into current asymmetric frameworks. Natural paths of egress to breach your networks include the billions of legacy appliances, many within our most critical infrastructures, which have not or can not adjust the key strengths that were considered safe and secure years ago.

--

Dictionary attack materials are freely available on the Internet and can be constructed by anyone for testing.

We are recommending that standards and regulatory groups mandate the use of much stronger keys where PKI is used or still in use as quickly as possible.

abrisson@wnlabs.com