Deterministic pseudo-random number generation in the production of a one-time-pad

Creating a Whitenoise super key

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

Abstract—One-time-pad cryptography utilizes a symmetric encryption key, one that must match or exceed the length of some corresponding plaintext message that is to be encrypted. For scenarios in which plaintext data is sizable, a key of requisite length can prove expensive in terms of storage and management. As a technique for solving such issues, Whitenoise Laboratories describes a method of creating and organizing the pseudo-random data comprising such keys. By consecutively applying sub-keys having shorter, non-repeating random lengths, a sequence of pseudo-random values can be produced for use in the encryption and decryption of variable length and arbitrarily large plaintext messages.

The Whitenoise cryptosystem implements a stream cipher based upon the above mechanism, which cipher itself becomes a source of deterministic, quality pseudo-random data. This stream cipher is able to produce an arbitrarily large quantity of non-repeating data for use in, for instance, the encryption of large data set and the creation of cryptographically strong, variable length tokens.

The Whitenoise stream cipher also acts as a deterministic, highly pseudo random number generator. This creates a functionally unlimited amount of highly pseudo-random data to both encrypt large data streams and to create variable length tokens that can be polled ahead in a bit stream to provide continuous, dynamic one-time-token authentication.

Proper management of initialization vectors also allows a single key to be safely used for one-time-pad encryption, one-time-token authentication and other key based security controls. The most striking advantages of Whitenoise are its speed and one-time-pad security that requires the smallest amount of computational effort and resources.

Keywords—Whitenoise; one time pad; encryption; authentication

I. ALGORITHM DESCRIBED IN GLOBAL PATENTS

A deterministic (highly) pseudo random number generator allows endpoints in a communication to identically recreate any key stream with knowledge or the key schedule, initialization vectors and offsets.

"In symmetric methods of encryption, the sender and the recipient use the same code or key to encrypt and decrypt the message. The only completely secure cipher which cannot possibly be broken or deciphered is the One-Time Pad (OTP). An OTP takes a stream of bits that contains the plaintext message, and a secret random bit-stream of the same length as the plaintext (the key). To encrypt the plaintext with the key, each pair of bits from the key and plaintext is sequentially acted on by the exclusive-or function to obtain the ciphertext bit. The ciphertext cannot be deciphered if the key is truly random and the key is kept secret from an unauthorized party. The problem with this method is that the key should be at least the same length as the message. If a shorter key is used and repeated then the cipher can be broken. In some cases the data which needs to be encrypted is extremely large." [1]

Note: for clarity sections are quoted directly from a US patent. Patents have been granted globally in countries such as the US, China, India, EU, Canada etc.

"… therefore provides a method of generating an encryption key having length x, the method comprising the steps of: i) selecting a number n of sub-keys each having a unique non-repeating length m; ii) generating n random numbers, one for each sub-key, each having length m; iii) generating a n+1 st random number R; iv) for each bit whose position in said nth random number is calculated as Modm(R) applying a function to all n bits to generate a binary value; v) concatenating said binary value to the end of the encryption key; and vi) repeating step iv) until the key is x bits in length. Preferably the selected length m of each sub-key is a prime number.

"… each of the n random numbers is generated by: i) generating a first random number which is not a perfect square; ii) calculating the square root of the first random number; iii) generating a second random number; iv) commencing with a digit whose position in the first random number is calculated based on the second random number, taking finite strings of digits sequentially and converting each finite string into a hexadecimal byte; and vi) concatenating each hexadecimal byte sequentially to the random number until the selected length m of the random number has been reached.

"In particular an encryption key, a non-repeating key of indefinite length referred to herein as a Super Key, is formed by combining sub-keys. Any number n of sub keys K_n can be specified depending on the application. The greater the number of sub-keys, the greater the length of the non-repeating Super Key (sic). The length of each sub key is a prime number of bytes (preferably with prime numbers larger than 10).

"The first step in the process is to determine how large a Super Key, or cipher, to deploy. The number of sub-keys and the non-repeating length of each sub-key, in bytes, is selected.

The sub-keys each have a unique non-repeating length. No two sub-keys are of the same non-repeating length. Preferably the sub-key non-repeating lengths are prime numbers of bytes. The selection may be done by manually entering the number of sub-keys and their prime number non-repeating lengths. Alternatively, the number of keys and their prime number non-repeating lengths is programmed into an application, or a program randomly selects the number of sub-keys and their non-repeating length. For n sub-keys K_n, the non-repeating length of the Super Key will be Size (K_1) X Size (K_2) X Size (K_3) . . . X Size (K_n). For example, assume **10** sub-keys of the following prime number non-repeating lengths are used:

- Sub Key **1**=13 bytes=K_1
- Sub Key **2**=17 bytes=K_2
- Sub Key **3**=19 bytes=K_3
- Sub Key **4**=23 bytes=K_4
- Sub Key **5**=29 bytes=K_5
- Sub Key **6**=31 bytes=K_6
- Sub Key **7**=37 bytes=K_7
- Sub Key **8**=41 bytes=K_8
- Sub Key **9**=43 bytes=K_9
- Sub Key **10**=47 bytes=K_{10}

"The resulting non-repeating Super Key length is 13×17×19×23×29×31×37×41×43×47=266,186,053,068,611 bytes. Thus, using a small number of sub-keys, each of small prime number non-repeating length results in an extremely long non-repeating Super Key. The total definition for the size for the multi-key above is contained in 300 bytes and the header.

"While preferably the non-repeating length of each sub-key is a prime number of bytes, to improve the randomness of the resulting cipher, the method will also work if non-prime number lengths are used, as long as the resulting cipher is very large.

"Each sub-key of the multi-key process may be created as follows. First a random number which is not a perfect square is generated, preferably by a computer random number generator. This serves as a "first seed value" O. Random number generators that are included in the operating systems of most computers are pseudo-random and not very robust. These values, however, are sufficient as a starting point.

"It is verified that the selected value O is not a perfect square. If it is, then additional random values will be generated until one meets this criterion. A second random number P ("second seed value") is also generated by the computer's random number generator to serve as an offset to be utilized in this process. The square root Q of this first seed value O is calculated, resulting in an irrational number Q (one that extends infinitely after the decimal point since it is not evenly divisible). The resultant string of digits after the decimal point is potentially infinite in length and is highly random. The computer discards the digits in front of the decimal and computes the number Q up to P digits after the decimal. Then, starting at the Pth digit of Q after the decimal point, the computer sequentially selects **4** digits at a time, and calculates the Mod 256 value of the 4 digits. The single resultant random 8-bit byte may be represented in hexadecimal notation. This value is used as the first byte of the sub-key. This process is repeated **4** digits at a time, continuing with the next digits in sequence, until a string of random data equal to the prime number non-repeating length of the sub-key being created is completed. This process is repeated for all the sub keys until the non-repeating length for all the sub keys are created. Each sub-key then is formed by taking the non-repeating string of bytes thus created, and repeating it as often as necessary in combination with the other sub-keys to create the Super Key.

"Once all the sub-keys are created as above, the Super Key (cipher) is created to the length required. This means the Super Key will continue to be created to encrypt the associated data to be encrypted, and continues to be created only until all the data is encrypted. First a random number R ("third seed value", or the starting offset for the Super Key, as opposed to the starting offset P for the number Q) is generated. Starting with any one of the n sub-keys, having length m, the Modm of R is calculated and the Modm(R)th byte of each sub-key is consecutively exclusive-or'd (X/OR'd) with the corresponding Modm(R)th byte of every other sub-key. For example, if R=100, and the length of the first sub-key is 97 bytes, then the 3rd byte of sub-key **1** is selected and XOr'd with the corresponding bytes of the other remaining sub-keys based on R selected in the same way. The process is repeated until all the selected bytes from each sub-key have been X/OR'd. The resultant binary value is then added to the Super Key. The next, subsequent bytes of sub-key **1** is then X|OR'd with the next byte of Sub key **3** and so on. Again the process is repeated until all the selected bytes from each sub-key have been X/OR'd. The resulting binary value of each function is again added to the Super Key. While the X/OR function is preferred, it will be apparent that other functions can be applied. For example, mathematical functions of addition or subtraction can be used. As each byte of the Super Key is generated, the corresponding byte of the plaintext message is then encrypted with the corresponding byte of the Super Key by the exclusive-or function or some other mathematical function. Once all the bytes of the plaintext message have been encrypted the generation of the Super Key terminates. The encrypted message can then be decrypted applying the inverse of the encrypting function to it and the Super Key.

"While preferably the random non-repeating string which forms each sub-key is generated as described above, the method will also work if the non-repeating string of each sub-key is simply generated by a random number generator to form each sub-key, as long as the overall resultant length of the Super key is sufficiently large so that the resultant Super Key is at least double the size of the data to be encrypted." [1]

For additional clarity, let's look at how this process is described in a scope document:

"Whitenoise Substitution is obtained by plugging the implementation of the key setup phase into the implementation of the output generation phase. To fully

specify Whitenoise Substitution, it suffices to separately specify each of these two phases.

II. ALGORITHM DESCRIBED IN SCOPE DOCUMENT

"1. Treat seed1 as the decimal representation of an integer in the range of 500-700 digits.
2. Let X := seed1
3. Let Y := X is the irrational number generated by square rooting X
4. Let Z1, Z2, Z3, Z4, .. be the digits after the decimal point in the decimal representation of Y. Each Zi is in the range 0,..,9.
5. Call srand(seed2). // only the first time
6. Call rand() to get the irrational starting point, start.
7. Let start := rand() mod 100. start is in the range 0,1,..,99.
8. Throw away Z1 and Z2 all the way to Zstart.
9. Let tmp := 10*Z(start +1) + Z(start +2). Throw away those used values.
10. Let n := 11 + (tmp mod 20) and n is in the range 11,12,..,30.
11. For i := 1,2,..,10, do:
12. Let j = 4*(i-1)
13. Let tmp be the next byte from the Z stream.
14. Let tmp := 1000*Zj+1 + 100*Zj+2 + 10Zj+3 + Zj+4
15. Let t := 1862 - (tmp mod 1862) and t is in the range 1,2,.., 1862.
16. Let u be the tth prime among the sequence 2,3,5,.., 1862.
17. If u is equal to any of l1 , l2 , .., l{i-1}, set t to (t+1)mod 1862 goto 16
18. Set li = u.
19. Next I : goto 11 until all 10 subkey sizes are set.
20. Then get remainder of lengths
21. For i := 11,..,n, do:
22. Let j = 5*(i-1)
23. Let tmp be the next byte from the Z stream.
24. Let tmp := 10000*Zj+1 + 1000*Zj+2 + 100Zj+3 + 10*Zj+4 + Zj+5
25. Let t := 16000 - (tmp mod 16000). t is in the range 1,2,..,16000.
26. Let u be the t.
27. If u is divisible by any of l1 , l2 , .., l{i-1}, (or u is not co-prime) set t to (t+1)mod 16000 goto 26
28. Set li = u.
29. Next I : goto 21 until all subkey sizes are set.
30. For i := 1,2,..,n, do:
31. For j := 0,1,2,..,li , do:
32. Let k := 4*j
33. Let tmp be the next byte from the Z stream.
34. Let tmp := (1000*Zk + 100*Zk+1 + 10*Zk+2 + Zk+3) mod 256
35. Let := tmp i j s
36. Next j : Next subkey byte
37. Next I : Next subkey 6 of 8 Whitenoise Laboratories Inc.
38. For i := 0,1,2,..,65535, do:
39. Set S[i] := Int(i / 256) (integer value only, truncated)
40. Next i: initialize S-box to 256 of each value
41. For i := 0,1,2,..,65535, do:
42. Let j := 4*i
43. Let tmp := (10000*Zj + 1000*Zj+1 + 100*Zj+2 + 10*Zj+3 + Zi+4) mod 65536
44. Set tmp2 := S[i]
45. Set S[i] := S[tmp].
46. Set S[tmp] := tmp2.
47. Next i
48. Let offset := ZiZi+1…Zi+9
49. Return n, (l1 ,l2 ,..,ln), (s1 ,s2 ,..,sn), S[65536] and offset.
50. Save in keyfile and add seed1 and start value to DB
51. Increment seed1 and goto 2 //repeat until enough keys are created" [2]

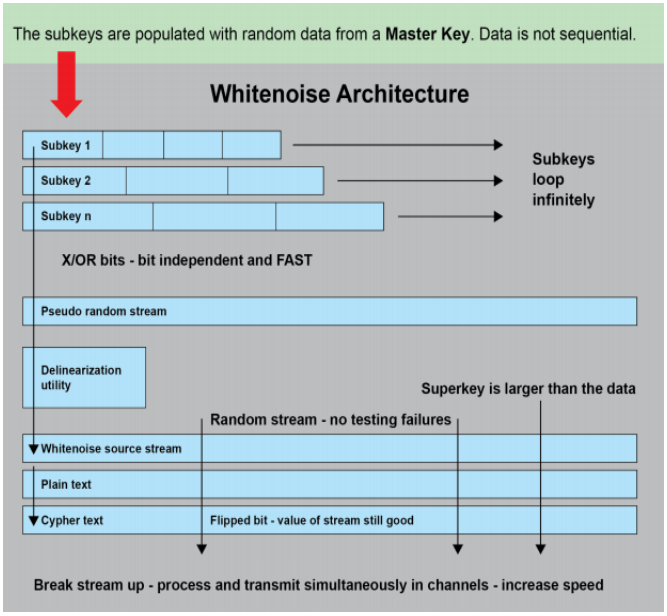
III. VISUAL MANIFESTATION OF THE ALGORITHM

It is also useful to be able to examine a visual representation of the Whitenoise key creation algorithm.

"A symmetric master key is normally used to derive one or more other keys". [3] Each endpoint is provisioned with a unique key structure which is generated from the server master key.

In one configuration the server retains copies of all the endpoint keys within its secure network. Each endpoint has a unique, distributed, symmetric key. After one-time key provision, there is generally never any exchange of key material. When session keys are used for direct point-to-point communications there is never any transfer of key material in an unencrypted state.

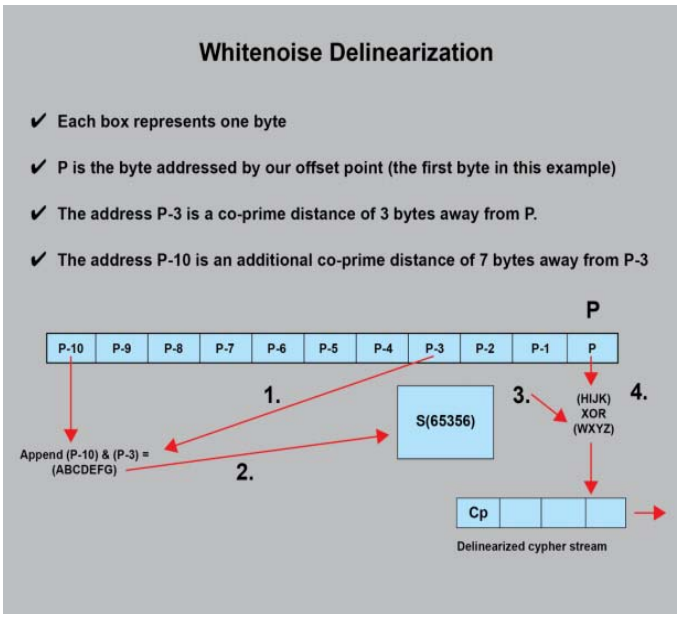
Let's look at Whitenoise key creation.



The top third of the above illustration shows subkeys that have been created. The subkeys loop left to right and together they form a data array that acts as a deterministic random number generator. This array can be used only until the point where all the seams of all the subkeys line up vertically. At this point the data source would be repeating itself and we would lose one characteristic of a one-time-pad because keys would repeat. A novel advantage of such a structure is that it has very little processing overhead and can be accurately recreated from a very small key schedule footprint.

The illustration shows:

- There are a variable number of prime number length subkeys. Each bit is XOr'd with the corresponding bits of the next subkey vertically.
- This results in a bit independent stream meaning that no bit in the key stream is related to any other bits.
- Two bytes (8 bits) are appended together and run through an S Box. Only one byte emerges. This becomes the first byte of a delinearized key stream.
- This creates a one-way function because you cannot go backwards and guess two bytes of information from one.



In the middle of the first illustration we see an S Box that delinearizes the key stream and creates several one way functions.

One delinearization solution is simply using an invertible delinearization utility such as is available from the federal Sandia Labs.

After an unproven break claim on Whitenoise with a 2-byte delinearization technique the three-byte delinearization technique illustrated was implemented at the suggestion of Wu and Wagner. http://eprint.iacr.org/2003/249/

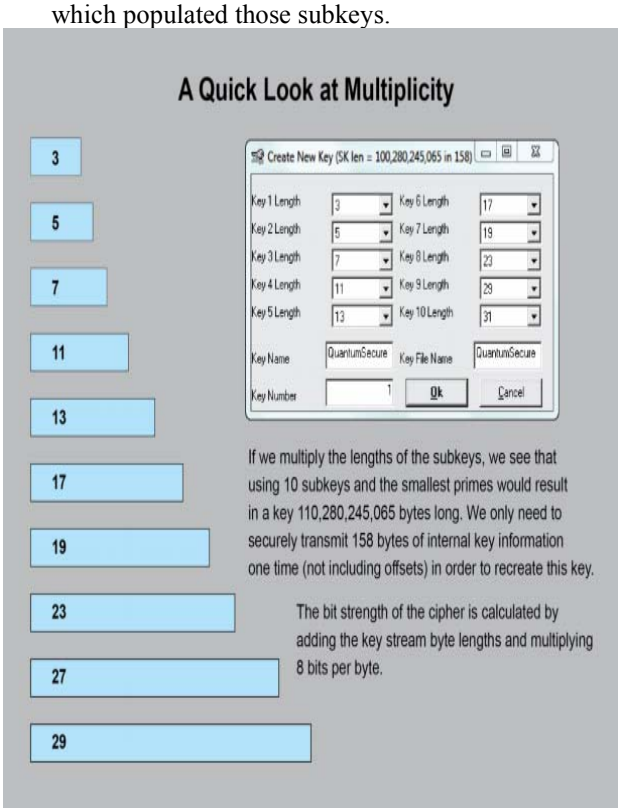
It deploys "a large substitution box with 65536 slots for the full 16 bit (2 bytes) addressing, but only outputs 1 byte or 8 bits for one slot. This makes for 256 outputs that would be the same value and 256 different values but would be arranged randomly (each value 0 to 255 would appear 256 times) to completely delinearize the output stream. To improve the data to noise ratio, the output is XORed with a third Whitenoise generated byte. This improved cipher was revised based on the attack given at http://eprint.iacr.org/2003/250/." [3] That attack has never been proven or demonstrated. Note that it was the very next paper submitted to ePrint. This would not indicate much research or testing relative to the filing of the unproven attack claim paper.

IV. One Way Functions

Many cryptographic functions rely on a single "one way function." A one way function is a process that is considered possible to do in one direction but is considered difficult or impossible to do in the opposite direction. An example of this would be RSA Integer Factorization Cryptography creating a prime number composite with the security proposition resting on the difficulty of factoring such composites.

Whitenoise has several one way functions:

- It operates as a one time pad which can be proven to be unbreakable.
- It is not possible to go backwards and guess or calculate two bytes of key information from one byte of captured information.
- The hacker has NO knowledge of the number of subkeys.
- The hacker has NO knowledge of the length of the subkeys.
- The hacker has NO knowledge of the master key which populated those subkeys.



The length of the resulting key stream is determined by multiplying the length of the subkeys in bytes. The strength of the key stream is determined by adding the lengths of the subkeys in bytes and multiplying by 8 bit per bytes. The above illustration shows the smallest and weakest out-of-scope Whitenoise key that can be made. This was the size evaluated by Wagner. We see that we can create a keystream of over 100 billion bytes long and we only need to store 158 bytes of key structure information to recreate this key.

A former scientist from General Dynamic UK noted:

- "Whitenoise generates deterministically random one time pad (OTP) from small base keys

- 138 byte key (smallest) generates 10^{11} byte OTP, to more than 10^{60} byte OTP." [4]

V. UNIQUE KEYS FROM GENERIC KEY SCHEDULE

The key schedule of subkey lengths allows creation of one-time-pad key streams. It requires little device computational resources and is very fast because of the use of the XOR function. It also creates key distribution and key management advantages.

With globalization many devices requiring security are manufactured abroad. This is a security risk in its own right. Looking at the illustrations it is easy to see that one can add as many subkeys as desired or required.

It is possible to provision devices with the same generic key schedule. The endpoint can then add secret passphrases (converted to subkey hexadecimal equivalents) to perturb the key schedule and create their own unique and secret key. The service provider would have no knowledge of this key and no responsibility in managing it. The user can access the cloud with confidence for storage and communications knowing that the service provider cannot breach their trust. Manufacturers would avoid permitting difficulties because all devices would have the same generic key.

VI. SECURITY AND PERFORMANCE

Cryptographic security conclusions come in two varieties. Following the canons of scientific method some are the product of research, testing, validation, repetition, demonstration and broad peer review. Some are unsupported statements or opinions driven by non-scientific priorities. Both have consequences.

Scientific analyses were performed by a cryptography icon, David Wagner, at the University of California, Berkeley (security analysis) and Dr. Issa Traore of the University of Victoria, British Columbia (performance analysis).

Wagner concluded: "I was unable to find any security weaknesses in the Whitenoise stream cipher. Whitenoise resists all of the attack methods I was able to think of…this provides evidence for the security of Whitenoise." … "it deserves to be further vetted by the security community at large." [5]

"Exhaustive keysearch is not a threat. With the recommended parameters, Whitenoise uses keys with at least 1600 bits of randomness. Exhaustive search of 1600-bit keys is completely and absolutely infeasible. Even if we hypothesized the existence of some magic computer that could test a trillion trillion key trials per second (very unlikely!), and even if we could place a trillion trillion such computers somewhere throughout the universe (even more unlikely!), and even if we were willing to wait a trillion trillion years (not a chance!), then the probability that we would discover the correct key would be negligible (about $1/2^{1340}$, which is unimaginably small). Hence, if keys are chosen appropriately and Whitenoise is implemented correctly, exhaustive keysearch is not a threat." [5]

"In a concerted attempt to break Whitenoise, I tried many creative attacks, including both standard ideas as well as off-the-wall non-standard methods. I was unable to come up with any approach that could make a dent in Whitenoise. I cannot think of any method that holds promise for attacking Whitenoise." [5]

Dr. Issa Traore tested the performance of Whitenoise against the appropriate NIST test suite. He was also unable to find any security flaws in testing against a super computer array. Dr. Traore increased the sensitivity of the tests by an order of magnitude allowing only one statistical error for every thousand tests. There was not a single randomness failure including anticipated statistical errors. He wrote:

"In our evaluation, using the significance level of a=0.001 and 1000 test sequences, the Whitenoise has no failure for any test cases. The Whitenoise algorithm exhibits excellent randomness level."[6] "The Whitenoise algorithm has a provable mathematical foundation, and the output of the Whitenoise algorithm exhibit good randomness based on the NIST statistical testing approaches."[6]

The high degree of randomness of Whitenoise keys and ciphertext was striking as it appears to be orders of magnitude more random than samples of radioactive decay. Additionally, it is surprising to note that as the number of Whitenoise subkeys increases so does the degree of entropy.

Increasing entropy is contrary to canon. Although determining why this is the case was not within the scope of the research study "it has been known for 30 years that there should be many ways to combine two imperfect sources to generate nearly perfect random numbers, but only now have researchers show explicitly how to create such "two-source extractors." [10] This has been validated by theoretical research scientist Avi Wigderson. Because randomness is so critical for quantum computing, cryptology, and general number theory this merits further investigation.

"A symmetric master key is normally used to derive one or more other keys." [3] A unique key is distributed to endpoints in the form of a subkey schedule. Each subkey length is populated with random data from the application or master key. It is possible that each subkey is acting as an additional random data source which would account for increasing entropy as the number of subkeys increases. Curiously the importance of randomness is often downplayed by cryptographers. This may be born out of the reality that available random number generators have limitations.

It has been reported that both Microsoft and PGP no longer use NIST prescribed random number generators because of the security risk of imperfect random number generation and the fear of possible back-doors such as those suspected in an NIST approved ECC random number generator.

There are also unsupported opinions and claims. "During the time the security analysis was being performed a prominent

cryptographer and on-line crypto blogger wrote a slanderous and unfounded editorial in “The Dog House”. [7] This was written contiguously with the security analysis and lacked any scientific foundation even though the author had access to testing results from his colleague Wagner.

Wu, as well, made an unproven claim of a Whitenoise break and was unable to demonstrate it in “The Whitenoise Security Challenge” and the “BlackHat/Defcon” security challenge. These were two, large, year-long, global security challenges with large money prizes for any successful demonstration of a break. There were no contest winners.

Wu wrote subsequently “The revised Whitenoise can resist my attack on the originnal (sic) Whitenoise.” [8]

Inexplicably, IACR ePrint refused to allow the publication of a mathematical proof written by Stephen Boren and Daniel Wevwrick, a crypto-mathematician from Communications Security Establishment (Canada).

“There is one critical flaw in the break suggested by Mr. Wu. That flaw is the fact that you can get an association of similar bytes of the inner Superkey stream but despite that, you can not get the actual byte represented. In formulating the linear system suggested by Mr. Wu to break the key, the resulting output of each linear equation is zero. As such, there are then many solutions to the system. One solution is where all variables are zero, another solution is where all subkey values are 1’s, and there are in fact many combinations of those variables that are solvable in this context but do not actually solve the system.” [9]

VII. CONCLUSION

The only encryption or key technologies that can be proven to be unbreakable are one-time-pads. There have never been any demonstrable attacks on Whitenoise. Proper management of initialization vectors and offsets allows Whitenoise super keys to be used as a one-time-pad.

Whitenoise imposes identity and data provenance with unique authenticated encryption.

Whitenoise can provide security benefits in cryptographic strength, speed, and flexibility. It can be used for rapidly scalable secure networks because a single key has to be distributed to an endpoint only one-time. The resulting key streams are so large as to realistically never be exhausted by an endpoint.

Whitenoise provides the strongest cryptographic strength keys with the minimum amount of computational effort and device resources.

VIII. REFERENCES

[1] Whitenoise Canada Patent Number 2 505 338 Method of Generating a Stream Cipher Using Multiple Keys (Note this technology is now broadly patented ie US, China, India, EU etc.) [Online] www.google.com/patents/WO2005076521A1?cl=ar

[2] A. Brisson, Stephen Boren “Software Specifications For Tinnitus Utilizing Whitenoise Substitution Stream Cipher (Revised) [online] http://eprint.iacr.org/2003/249.pdf

[3] National Institute of Standards and Technology NIST [Online] Available http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

[4] C. Coram, “Whitenoise – An Overview,” White Paper, June 2015. [Online] : http://www.wnlabs.com/papers/Whitenoise_Overview_Short.pdf

[5] D. Wagner, “A Security Evaluation of Whitenoise,” University of California, Berkeley, Oct. 2003. Page 4 [Online]. Available: https://eprint.iacr.org/2003/218.pdf

[6] I. Traore and M.Y. Liu, “Evaluation of Whitenoise Cryptosystem. Part 1: Encryption Algorithm,” Technical Report ECE03-3, University of Victoria, British Columbia, Canada, Feb. 2003. [Online]. Available: http://www.wnlabs.com/downloads/UVIC_Performance_Analysis.pdf

[7] A. Brisson, “Unclassified History of Whitenoise”, [Online] Available www.wnlabs.com/pdf/Unclassified_story_of_Whitenoise_2017.pdf

[8] A. Brisson “The 24 Hour Cycle of Truth”, http://www.wnlabs.com/Papers/History_of_Whitenoise_WN_Can't_be_broken.pdf

[9] S.L. Boren and DW (CSE), Mathematical rebuttal refuting false break technique,[Online].Available: http://www.wnlabs.com/pdf/Response.pdf page 2

[10] Don Monroe, January 2017 [Online] Available: https://cacm.acm.org/magazines/2017/1/211100-pure-randomness-extracted-from-two-poor-sources/fulltext