HEURISTIC ONE-TIME PAD ENCRYPTION

A Dissertation Submitted
to the Graduate School
University of Arkansas Little Rock

in partial fulfillment of requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

in the Department of Computer Science
in the Donaghey College of Engineering & Information Technology

December 2019

James Burford Joyce

B.S.E.S., University of South Florida, 1994

This dissertation, “Heuristic One-Time Pad Encryption,” by James Burford Joyce, is approved by:

Dissertation Advisor:

_____
Daniel Berleant
Professor of Information Science

Dissertation Co-Advisor:

_____
Kenji Yoshigoe
Professor of Information Networking for Innovation and Design, Toyo University

Dissertation Committee

_____
Daniel Berleant
Professor of Information Science

_____
Chia-Chu Chiang
Professor of Computer Science

_____
Mariofanna Milanova
Professor of Computer Science

_____
Kenji Yoshigoe
Professor of Information Networking for Innovation and Design, Toyo University

Program Coordinator:

_____
Daniel Berleant
Professor of Information Science

Graduate Dean:

_____
Brian Berry
Professor of Chemistry

Fair Use

Duplication

HEURISTIC ONE-TIME PAD ENCRYPTION by James Burford Joyce, December 2019

ABSTRACT

Heuristic One-Time Pad (HOP) represents a methodology that is compliant with the requirements as established by Claude Shannon for a globally scalable and permanent encryption solution. The technique is not limited to encryption; rather, it also encompasses dynamic hashing and authentication. The two key Shannon-identified problems of continuous random number generation and secure distribution of encryption keys have been solved and are integral to the method. At each step of the algorithms, entropy is '1' and unicity is 'infinite'. All encryption key generation is independent of the value of the data being encrypted, and is dependent upon a combination of quasi-random data values and their respective matrix index values. The creation of functionally random numbers results from combining a plurality of quasi-random sources and breaking up any potential linear, sequential, or harmonic anomalies. This, in conjunction with a novel data shuffling and salting technique, ensures that the encryption cannot be attacked via frequency analysis or even brute force. Contrary to all previous versions of the One-Time Pad, HOP does not require large encryption keys to be themselves distributed across any medium; rather, the keys are generated exactly when and where they are needed. The footprint of HOP is small enough to support operation on systems as small as Internet of Things (IoT) and wireless sensors, while also being robust enough to handle the encryption needs of the largest supercomputers. Regarding speed, in direct head-to-head testing HOP is approximately 3.5 times faster than Advanced Encryption Standard (AES). With regards to key security, even if the encryption keys are stolen, they cannot be used by an unauthorized person or system to

break the encryption. HOP represents a mathematically-proven uncrackable encryption system which is faster than the competition, runs on all platforms, is globally scalable, and is, as is intuitive from Shannon's work, quantum as well as future proof. As such, it ends the historically perpetual cycle, or race-condition, of encryption algorithm creation and encryption algorithm breaking.

Acknowledgements

With respect to getting the encryption code to its current point, I must acknowledge and thank Anna Lanham and Edwin Carp for their countless hours of devotion and tremendous talents in back office systems architecture and coding. Regarding the last ten years, I would like to thank my Committee members. Each of you have taught me deep knowledge in areas where I thought I knew something going in, but now realize I had yet to scratch the surface. This work would not have been possible without your abilities to confer your knowledge, skills, and experience. Thank you.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

So, what is encryption anyway? Encryption is the art/science of encoding information with the hope that only authorized people, devices, or processes will be able to access it. The history of encryption goes back to numerous ancient cultures. Examples include the Greek Skytale, the Caesar Cipher of Rome, and even the Kama Sutra has a chapter dedicated to "secret writing". Also, historically, the world of encryption has been in a race-condition; meaning, since the proverbial dawn of time, someone invents a method to encrypt data, and someone thereafter invents a way to break it. The most significant event in encryption cracking did not occur recently. In fact, al Kindi (~800 A.D.) discovered "frequency analysis", which has been the backbone of encryption cracking ever since. Contemporary cryptographers have moved encryption algorithms towards more and more complex mathematics in an effort to minimize the risk posed by the ever more evolving world of encryption cracking. An excellent example of this is Vigenere's Autokey Cipher (once "believed" to be unbreakable), and the cracking work of Charles Babbage. Every time a new encryption method has been released, it has eventually been successfully attacked by new cracking techniques, thereby perpetuating the encryption race-condition.

Encryption, in general, is not limited to the fields of information theory, mathematics, and computer science; rather, it also finds itself rife with shades of politics, law enforcement, sociology, industrial espionage. Its influence has been definitive in issues ranging from the privacy of a love note, to the lives and deaths of people and empires. Such has been the import of encryption throughout recorded history, and I posit that its impact on the world will increase

going forward, as its use is rapidly becoming an imperative as opposed to an option. That being said, this work is not a treatise on the history of encryption: I recommend Singh, Simon, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* (New York: Random House, 1999). For One-Time Pad history specifically, append Singh's work with, Bellovin, Steven M., "Frank Miller: Inventor of the One-Time Pad". *Cryptologia*. 35,(2011), pg. 203–222. As well, this is not an argument for or against the need for stronger encryption, as the National Institute of Standards and Technology (NIST) has already made that point in the National Institute of Standards and Technology Interagency/Internal Report (NISTIR) 8105 – Report on Post-Quantum Cryptography, 28 April 2016. It is not a list or exhibition of grievances regarding the plurality of industrial espionage efforts directed towards this new one-time pad technique over the past nine years. Furthermore, it is not a political statement or judgement related to privacy rights versus the desires of law enforcement agencies to have backdoors or trapdoors built into all commercial encryption, though I would be glad to answer any questions in any of these areas in another forum. This work is strictly limited to the technology at hand.

Akin to patent work, it is assumed that the reader of this has "ordinary skill in the art" (encryption), and has a general knowledge of the one-time pad. The argument of the need for strong encryption to mitigate the threat of Quantum Computing (QC) has been made by the National Institute of Standards and Technology (NIST) in their Interagency/Internal Report (NISTIR) 8105 – Report on Post-Quantum Cryptography, 28 April 2016. I agree in general with their assessment of the threat, and am proceeding into this work with that in mind: QC does present a real threat to contemporary encryption, especially (though not exclusively) as it pertains to asymmetric encryption and encryption handshake techniques. Many "expert"

recommendations are that we expand the sizes of our current keys, using existing algorithms, so that "larger" quantum computers will be required to crack the encryption. The two obvious downsides to this are that 1) it perpetuates the race condition and, 2) contemporary algorithms are already too large to serve the needs of smaller devices (Internet of Things (IoT), wireless sensor devices, etc.); ergo, increasing the system requirements to run an algorithm leaves smaller devices unsecured. Furthermore, with respect to attempting to hold off the QC risk (and perpetuating the race-condition), the future of QC, Non-deterministic Universal Turing Machines (NUTM), should make it clear that contemporary complex-algorithmic encryption techniques will eventually lose the race. To clarify, whereas a quantum computer will run an algorithm with all possible data simultaneously, NUTMs are theorized to run a large plurality (possibly all) algorithms with all possible data simultaneously. As such, I believe that, ultimately, contemporary "difficult to break" encryption will lose the race. A different type of encryption model will be necessary to ensure data privacy and security going forward.

Specifically, regarding the second point above (encryption for small devices), a paper by Kenji Yoshigoe and Murat Al ("Adaptive Confidentiality Mechanism for Hierarchical Wireless Sensor Networks," *2008 IEEE Globecom Workshops*, (2008)) is initially what gave me the spark for this dissertation. The paper expressed the fact that wireless sensor devices do not have the computing and/or power resources to handle contemporary encryption algorithms, and, as such, there was a need for strong light-weight encryption techniques to serve that market. I thought "I can do that". Then I thought "if this will work for small systems, it should work for all systems", and here we are. To note: NIST is looking for both quantum resistant and light-weight

encryption solutions in two different programs. This work consolidates that, resolves the encryption needs for all devices, and ends the encryption race-condition forever.

ONE-TIME PAD DISCUSSION

During World War I the United States experimented with a then new type of encryption, the One-time Pad (OTP), which was believed to be impervious to cracking attempts. In 1947, Claude Shannon (the father of Information Theory) proved that the OTP was mathematically impossible to break. It is the only form of encryption ever conceived to hold this distinction; however, while Shannon did prove uncrackability, he also identified two difficulties that would need to be overcome for the technique to be used on a large scale: first, OTP relies on a perpetual supply of random numbers, which is known to be exceedingly difficult to create; and second, there must be a secure way to distribute encryption keys to all parties involved in the communications. Perhaps in no small part to the elusive nature of the solutions to these two issues, the OTP has been dubbed the Holy Grail of encryption.

Specific to random number generation, numerous techniques have been tried, from trivial (trying to type randomly on a keyboard: this technique has been proven to not work) to complex (measuring radioactive particle decay, photon observations, or measuring thermal sensor changes: none of these techniques are fast enough to support wire-speed data rates). The fastest contemporary random number generation systems (Quantis from ID Quantique, and Entropy Engine from Whitewood Encryption Systems) can only generate keys at rates ranging from 128 - 350 megabits per second respectively. These keys, which still need to be distributed to relevant systems, must then be used by external encryption algorithms. This is nowhere near fast enough for current needs, and is a non-sequitur to a practical scalable encryption solution. For example,

a couple of years ago I was consulted regarding the encryption for India's new national tax system. During the course of our discussions, it was related to me that they had tried both Quantis and Entropy Engine, but found that, at best, it would take 18 hours each day just to distribute encryption keys to the computers in the nationwide network; whereas, they needed the keys to be available in real time to service 24x7 operations, this "solution" would have limited their production day to six hours.

The way that contemporary cryptographers have looked at OTP is, in my mind, the biggest reason that no one else has yet figured out this problem. Prevailing belief is that, for example, to encrypt 10 petabytes of data, one needs to pre-generate and distribute a 10-petabyte encryption key to needed locations. As such, the prevailing belief amongst cryptographers is that trying to implement a scalable OTP would be a key management nightmare. An example of this is found in a Bruce Schneier essay on the OTP: "Cryptography after the Aliens Land", *IEEE Security & Privacy,* September/October 2018. In fact, Schneier goes so far as to say: "Today, only crackpots try to build general-use systems based on one-time pads—and cryptographers laugh at them, because they replace algorithm design problems (easy) with key management and physical security problems (much, much harder)." Another example of cryptographers believing that OTP keys must be generated and distributed is a theme I have seen in cryptography blogs: to paraphrase - 'The increasing availability of bandwidth and the decreasing costs of bandwidth are making OTP key distribution more feasible.'

It should be intuitively obvious that the above bandwidth reasoning is a non sequitur and perpetuates the race-condition. It should also be intuitive that if you try to tackle the OTP

problem from a position of traditional thinking, any attempts to scale the OTP will be fruitless; ergo, Schneier's key management point is valid… from a traditional perspective. Regarding Schneier's conjecture that there is some increase in physical security problems, I would offer that the point would be valid IF you actually did have to continuously transmit encryption keys to everyone all the time. Heuristic One-Time Pad resolves these issues and provides a fast, lightweight, and globally scalable encryption solution that ends the encryption race-condition once and for all.

No one has yet to consider the possibility of generating synchronous, functionally random, encryption keys on a plurality of devices at the exact time they are needed (e.g. while the data to be encrypted or decrypted is being read into memory, and on the system that is doing the encrypting/decrypting). This work solves the problems first identified by Shannon, complies with Shannon's fundamental requirements that keys are destroyed upon use and may not be reused, and represents the world's first and only uncrackable and globally scalable encryption. Not only is this the strongest available encryption, but it is also approximately five times faster than Advanced Encryption Standard (AES), our current de facto encryption standard, making this the only proven unbreakable algorithm and the fastest encryption on (or off) the planet. It is impervious to the threats of today and tomorrow, including ever more evolving cracking techniques, QC, NUTM, or anything that comes after (Schneier: "aliens"). To go out on a limb, I would (tongue in cheek) tend to agree with Bruce in that it is certainly alien-proof. Regarding Schneier's specific issues of key management and physical security, due to new and unique key distribution and key-negotiation mechanisms, even if someone steals your keys during initial

installation, they still cannot crack your encryption. As such, the Heuristic One-Time Pad does end the encryption race-condition.

PROBLEM ACKNOWLEDGEMENT

Prior to being "well" into this research, initial versions of this document were significantly longer, and, in my opinion, the argument that I was making, for the need for post-quantum encryption, was less strong due to reliance upon speculation/extrapolation on technology evolution, trends, etc. The specific events that made me decide to modify my dissertation are as follows. First: In 2016, popular websites Yahoo and Dropbox made public announcements that their userbases had been compromised as a result of the cracking of their encryption. These represent the first two major public announcements by large organizations that they were hacked as a result of encryption cracking; Second: while I know of encryption compromises in the real world, I cannot be more specific due to legal/contractual obligations. To be clear, in my 30 plus year cyber-security career, I have cracked encryption while performing penetration testing for clients, and as a cyber security curriculum author and instructor for the United States defense/intelligence community, I have been privy to extensive information on encryption cracking; Third: The NISTIR 8105 Report on Post-Quantum Cryptography was released. Prior to the advent of the information from Yahoo, Dropbox, and the NIST Report, my paper essentially boiled down to my personal opinion regarding the need for stronger encryption. Having NIST validation, and public examples of current technology-based encryption cracking, significantly strengthens the argument that stronger encryption is necessary.

HOP METHOD NOTES

The actual technique for this encryption is extensively covered in the dissertation defense presentation. As this is a pattern-based heuristic, it seemed that the best way to express this technique was to graphically demonstrate it. As the details are covered in the presentation, I will touch on the key transition points here. At the core of this method is the ability to generate random numbers when and where needed. I have toyed with the one-time pad since I was a child and have had the concept of a one-way function in my brain for about as long. It occurred to me that multiple quasi-random generators could be used to generate streams of "functionally" random numbers. The idea of "matrix hopscotch" was also something that I have thought about for decades. When I first started writing them out ("them" being matrices with random distributions of the numbers 0 through 255), I tried to manipulate them with controlled sequences of modifiers (by "manipulate", I am referring to the "Go" concept shown in the presentation). For example, I tried to take a matrix and do a "Go 0", then a "Go 1", and repeat to modify the matrices. I found that this would always lead to the matrix converging back to its original state rather quickly. As such, this would never qualify for perpetual random number generation. After much empiricism, I found that modifying a quasi-random distribution based upon another quasi-random distribution would result in a sequence of patterns that did not repeat throughout its cycle.

I did, however, find possible classes of sequences that would break the encryption requirement of non-repeating keysets. An example of these sequences is, albeit randomly occurring, when the matrix falls into a straightforward (or backward) sequence of numbers from 0 to 255 in order (no matter where it starts). The "Flip and Swap" technique breaks this

condition. As well, as is intuitive from the graphics, the combination of "numerical order" and "matrix index" processes breaks any possible repetition short of absolute numerical exhaustion, which in this case is 16 matrices of 256 elements that can be arranged in any order [16 x 256! x 16!]. That's 16 times 8.578177753 E+506 times 2.092278988 E+13. Online resources refer to 256! times 2 as being functionally infinite. While we know that it's not actually infinite, we do know that we, and perhaps this planet, will be long gone before those permutations have been exhausted.

Specifically, regarding the Flip & Swap technique, I initially used a single offset, but Dr. Yoshigoe suggested that it would be preferable to use a double offset. I also tried a triple offset, but it did not give the algorithm any greater advantage, and would have just used more processor cycles. The key rotation method, in conjunction with Flip & Swap and the Epoch Roll processes described in the presentation, appears to satisfy a true one-way function. As such, it also appears to yield the solution to the "P vs NP" Millennial Math problem (spoiler: P != NP), though that formal indirect proof work, while implicitly alluded to, is not included in this dissertation. That being said, the proof should show that the matrix size (currently 256) approaches infinity, and should extend the principles for vector-space.

All details of methods for random number generation, data shuffling, salting, encryption, decryption, dynamic system identification, universal one-way hash functionality, and all primitives are fully covered in the dissertation defense presentation.

PERFORMANCE

As for performance, head-to-head testing was conducted against AES-128 to determine a relative speed for encryption. All code was written in C, identical test data was used, and the tests were run on the same processor core (Intel i7 6700K, 4GHz). AES key size – 128 bits, HOP key size – 2048 bits. AES speed – 624Mbps, HOP speed – 2.162Gbps. It is also important to note a significant difference between HOP and AES. AES, in and of itself, is not authenticated encryption, and relies upon an additional hash algorithm to provide authenticated encryption functionality. HOP is fully authenticated already, as a HOP-based hash function is built into the encryption and decryption routines. As such, HOP's speed advantage over AES is even more significant in that it is generating the encryption keys, performing the encryption, and generating the hash at the above stated speed, while AES is only performing the encryption.

Random key generation speed was tested against the manufacturers published key generation rates for Quantis and Entropy Engine. Quantis generates keys at 128Mbps, Entropy Engine at 350Mbps, and HOP at 2.162Gbps.

HOP OVERVIEW

- Symmetric One-Time Pad Encryption – Shannon compliant
- No external RNG or floating-point instructions – stay in kernel space
- Synchronized random keys generated on the fly on each platform
    - No need to transfer large keys between relevant systems
    - No need to store large keys
- 256-byte block authenticated encryption (not chained)
- Authenticated via 2048-bit Universal One-Way Hash Function (chained)
- Dynamic system/engine ID
- Secure distribution, authentication, & non-repudiation via auto-negotiate

ENCRYPTION AND DECRYPTION OVERVIEW

Encryption:

- Shuffle the un-encrypted data
- Salt the shuffled data – XOR Salt Key with shuffled data
- Shuffle the salted data
- Encrypt the shuffled salted data – XOR Pepper Key with shuffled salted data

Decryption:

- Decrypt the encrypted shuffled salted data – XOR with Pepper Key
- Un-shuffle the shuffled salted data
- Un-salt the salted data – XOR with Salt Key
- Un-shuffle the un-salted data

KEY GENERATION OVERVIEW – MATRIX HOPSCOTCH

- Each 16x16 matrix is a randomly distributed permutation of the values 0 – 255*
- The values in each matrix are rotated based upon a “value” vs “index” heuristic
- The heuristic is akin to the childhood game of Hopscotch
- … but the Hopscotch field is changing and is mapped to a Möbius strip
- Each key stack is comprised of four 16x16 matrices*
- The matrices are called Epoch, Cycle, Go, and Key
- Go controls Key, Cycle controls Go, & Epoch controls the rotation of Cycle values
- Two stacks are required to create functionally random keys

HOP HOST REQUIREMENTS

Each host will maintain:

- HOTPad executable code < 50kBytes
- Salt & Pepper stacks for Send & Receive for all relevant systems
- SID stacks for all relevant systems
- Dynamic Hash table
- Key Management Database
- < 7kBytes/connection

AUTO-NEGOTIATE PROCESS

The auto-negotiate process is a mechanism whereby two or more entangled encryption engines modify their own keyset in a way such that only they can possibly know what the updated keysets are, even if someone else had managed to obtain a copy of the original keysets. The importance of this goes beyond protecting keys from malicious actors out in the world; rather, this process also protects users against a rogue administrator that has access to the systems that generated the initial keysets for users' initial installations of the product. As such, a manufacturer of this technology will only be able to decrypt traffic that it directed towards its own systems, but they will not be able to track the encryption keysets of other user-to-user or system- to-system communications. As such, without actually taking possession of, and having the ability to authenticate on, a device, it will not be possible for any outside system to compromise the encryption.

APPLICATIONS

It should be noted that the following list represents product rollout over a ten year period.

- *HOP™ On-Cloud* – Encryption proxy, TLS-*HOP™*, and storage services
- *HOP™ On-Premise – HOP™ On-Cloud* within your firewall
- *HOP™ App* - Secure voice/video/data mobile and desktop apps for peer-to-peer and group communications
- Cyber-security consulting services
- Secure Email – *HOP™* plugins for email
- Secure Web Browsing – *HOP™* plugins for major browsers
- Authentication
- Biometrics
- Blockchain
- Financial transactions
- IoT & wireless sensors
- Chipsets
- Inter Process Communications
- Virtual reality
- User interface – I/O
- Nanotechnology

HOP PROXY

HOP Proxy Services represent the world's first mathematically-proven uncrackable, and globally scalable authenticated encryption solution. This is the first anticipated product that will use this technology.

- Entangled random HOP keys
- Scale to data size on the fly
- Auto-negotiation
    - Establishes encrypted link
    - Authenticates
    - Non-repudiates
    - Modifies & synchronizes keys
- Simple enrolment and delivery
- On-Cloud or On-Premise
- Small footprint
- No proprietary equipment
- No asymmetric component
- No random number generator

HOP TLS SERVICES

About two years ago, I reached out to Jack Lloyd, the creator of Botan SSL, and contracted him to build a port for HOP into TLS 1.2. As TLS has been upgraded to version 1.3, the plan is to get in touch with Jack and have him update the port.

Transport Layer Security (TLS) & its predecessor, Secure Sockets Layer (SSL), both frequently referred to as "SSL", are the cryptographic protocols that provide communications security.

- HOP™ – 1st option in the TLS algorithm stack
- Fall-through support for other TLS protocols
- Fastest & strongest TLS implementation available
- Platform ubiquity
- No proprietary equipment
- No asymmetric component
- No external random number generator required

HOP VOIP AND MOBILE

HOP™ Desktop – Uncrackable voice, video, and data communications for laptops, desktops, or even clusters – peer-to-peer and conference

HOP™ Mobile – Uncrackable voice, video, and data communications for smart phones and tablets – peer-to-peer and conference

PRODUCT DELIVERY

It is not sufficient to simply develop an algorithm to provide a full solution. The product delivery architecture has been crafted to ensure that the user experience is intuitive and the installation mechanics are transparent. A full and robust environment has been coded up, is ready for deployment, and covers end-user needs as well as all requisite manufacturer back office functionality. This architecture is shown on the first of the two flowcharts directly below, and the data flow is shown on the second flowchart.

Figure 1

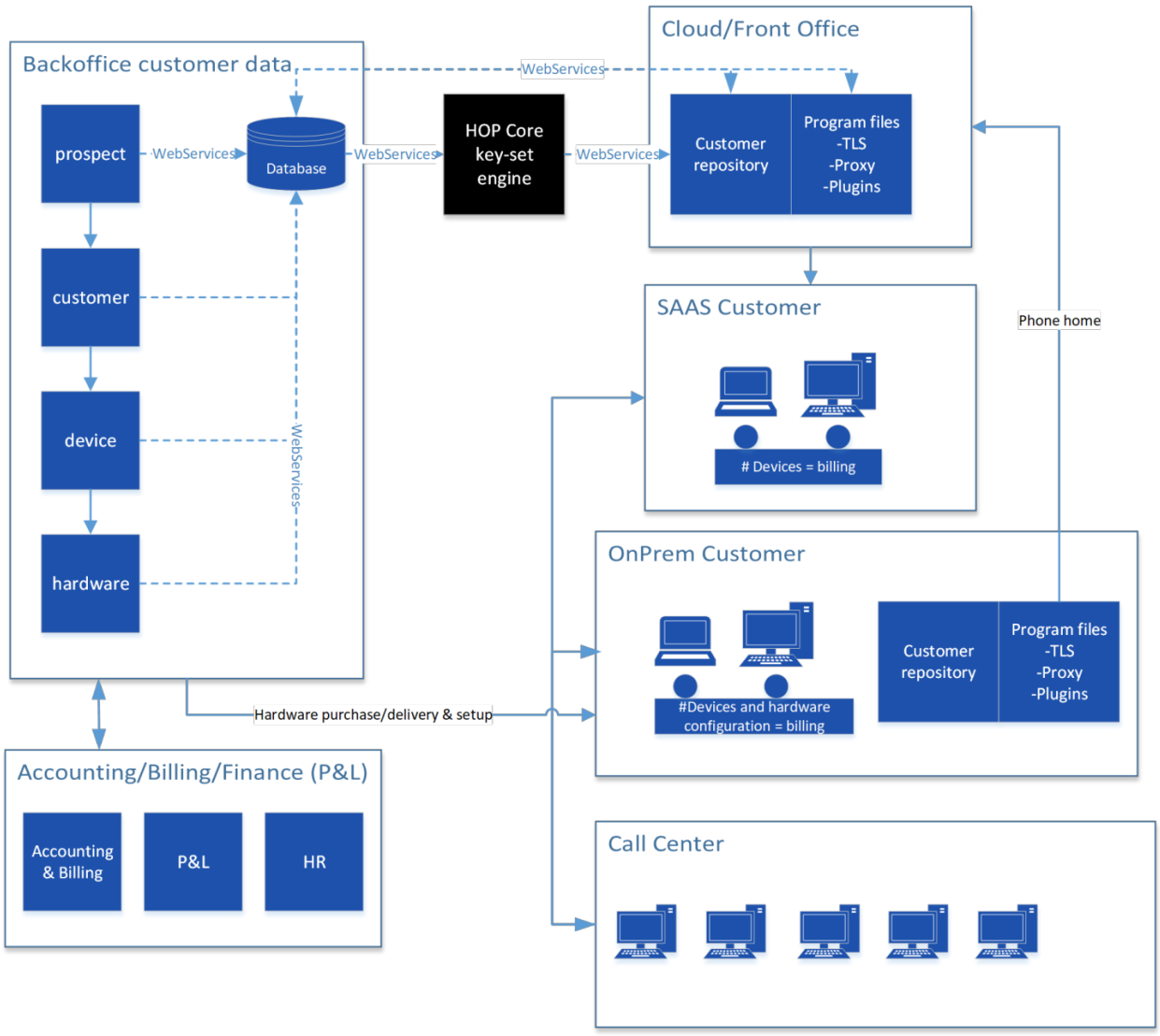
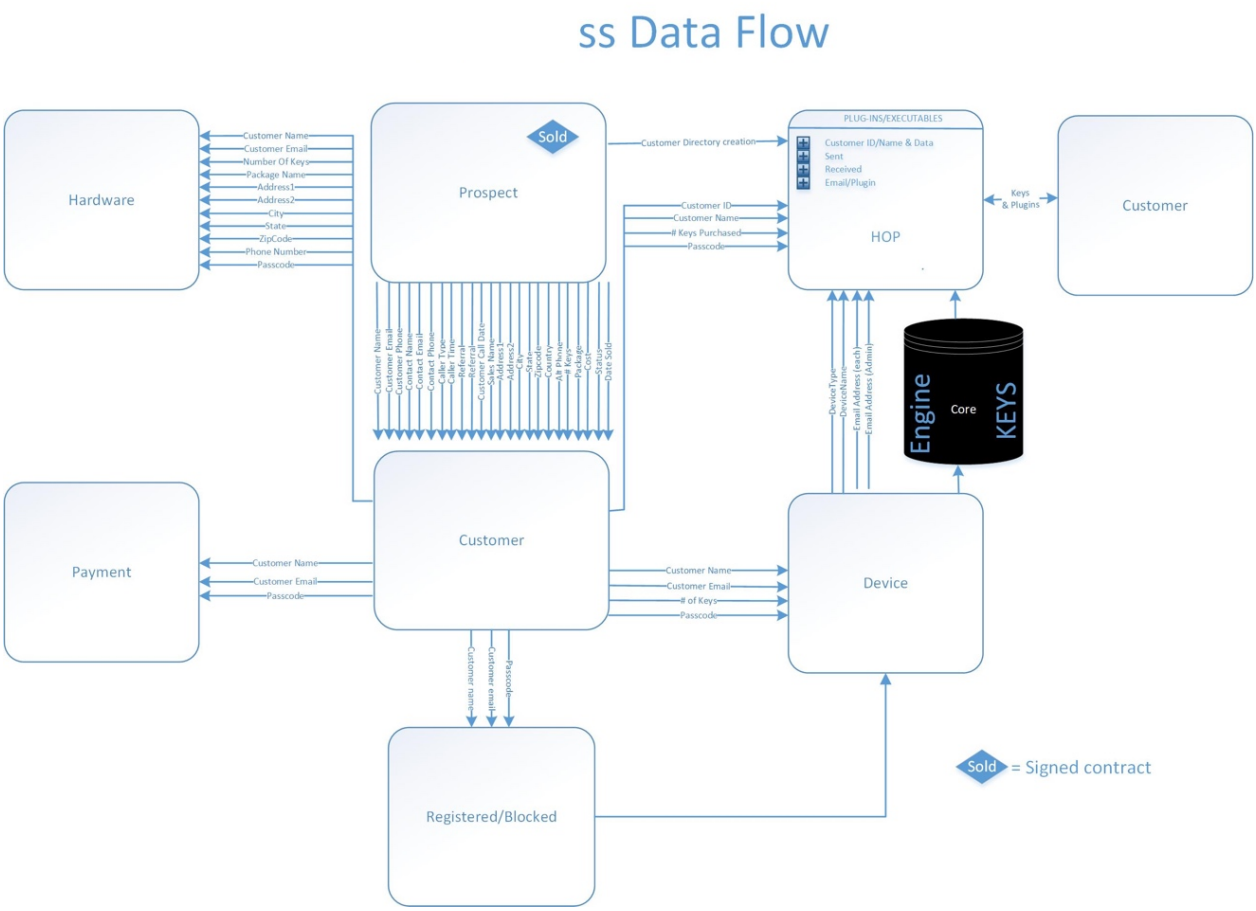

Backoffice customer data
prospect
WebServices
Database
customer
device
hardware
WebServices
WebServices
HOP Core key-set engine
WebServices
Cloud/Front Office
Customer repository
Program files -TLS -Proxy -Plugins
WebServices
SAAS Customer
Devices = billing
Phone home
OnPrem Customer
#Devices and hardware configuration = billing
Customer repository
Program files -TLS -Proxy -Plugins
Hardware purchase/delivery & setup
Accounting/Billing/Finance (P&L)
Accounting & Billing
P&L
HR
Call Center

Figure 2


ss Data Flow
Hardware
Customer Name
Customer Email
Number Of Keys
Package Name
Address1
Address2
City
State
ZipCode
Phone Number
Passcode
Sold
Prospect
Customer Directory creation
PLUG-INS/EXECUTABLES
Customer ID/Name & Data
Sent
Received
Email/Plugin
HOP
Keys & Plugins
Customer
Customer ID
Customer Name
Keys Purchased
Passcode
Engine
Core
KEYS
Payment
Customer Name
Customer Email
Passcode
Customer
Customer Name
Customer Email
# of Keys
Passcode
Device
Registered/Blocked
Sold = Signed contract

HOP SUMMARY

- Authenticated encryption
- The HOP Random Number Generator has passed the NIST STS suite
- Generate keys, encrypt, & hash at 2.16Gbps on a 4.0 GHz i7 (single core)
- Coded in C – far from optimized
- Suitable for all platforms from IoT to Clusters
- TLS friendly
- Mathematically proven uncrackable
- Quantum proof
- NUTM proof
- 👽 proof

RANDOM NUMBER GENERATION

Regarding the ability to create functionally random sequences, 6 megabytes of encryption keys were generated and sent to Dr. Kenji Yoshigoe. He ran the keys through the appropriate NIST testing (National Institute of Standards and Technology (NIST), Special Publication 800-22: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, April 2010). The keysets passed in all testing as shown below. As such, this passes the random number generation requirement per Shannon. Regarding the ability to securely distribute keys to respective locations, it should be intuitive to one with ordinary skill that the technique described in the presentation delineate a reasonable mechanism for secure key distribution, also as per Shannon. As such, this meets the requirements for a one-time pad. In addition, this encryption also accounts for known-text attack attempts against the keysets by shuffling and salting the data at two different points in the encryption process. As the functionally random keysets are, at the core, created deterministically, this series of shuffling and salting prevent an attacker from gaining any purchase upon any part of the encryption keys themselves. Below are the results on the NIST randomness testing. The gist is that this technique does generate functionally random numbers from deterministic quasi-random matrices.

NIST STATISTICAL TEST SUITE

There are several batteries of tests available for testing random or pseudorandom number generators; however, the NIST Statistical Test Suite (NIST STS) is the most widely accepted test suite. It is often used in preparation of formal certifications or approvals and was used in the process for establishing Advanced Encryption Standard (AES) as the current encryption gold standard. The NIST STS package is a set of statistical testing procedures for assessing randomness of binary sequences of interest and incorporates all recommended NIST tests. This has used NIST STS to assess the randomness quality of binary sequences being generated by its core technology. Below is a summarized description of the 15 tests of the NIST STS [1]. Interested readers are encouraged to read [2].

1. **Frequency (Monobits) Test**

   The focus of the test is the proportion of zeroes and ones for the entire sequence. The purpose of this test is to determine whether that number of ones and zeros in a sequence are approximately the same as would be expected for a truly random sequence. The test assesses the closeness of the fraction of ones to ½, that is, the number of ones and zeroes in a sequence should be about the same.

2. **Test For Frequency Within A Block**

   The focus of the test is the proportion of zeroes and ones within M-bit blocks. The purpose of this test is to determine whether the frequency of ones in an M-bit block is approximately M/2. The default value of M = 128, recommended by NIST, was used for this test.

3. **Runs Test**

The focus of this test is the total number of zero and one runs in the entire sequence, where a run is an uninterrupted sequence of identical bits. A run of length k means that a run consists of exactly k identical bits and is bounded before and after with a bit of the opposite value. The purpose of the runs test is to determine whether the number of runs of ones and zeros of various lengths is as expected for a random sequence. In particular, this test determines whether the oscillation between such substrings is too fast or too slow.

4. **Test For The Longest Run Of Ones In A Block**

The focus of the test is the longest run of ones within M-bit blocks. The purpose of this test is to determine whether the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones that would be expected in a random sequence. Note that an irregularity in the expected length of the longest run of ones implies that there is also an irregularity in the expected length of the longest run of zeroes. Long runs of zeroes were not evaluated separately due to a concern about statistical independence among the tests.

5. **Random Binary Matrix Rank Test**

The focus of the test is the rank of disjoint sub-matrices of the entire sequence. The purpose of this test is to check for linear dependence among fixed length substrings of the original sequence.

6. Discrete Fourier Transform (Spectral) Test

The focus of this test is the peak heights in the discrete Fast Fourier Transform. The purpose of this test is to detect periodic features (i.e., repetitive patterns that are near each other) in the tested sequence that would indicate a deviation from the assumption of randomness.

7. Non-Overlapping (Aperiodic) Template Matching Test

The focus of this test is the number of occurrences of pre-defined target substrings. The purpose of this test is to reject sequences that exhibit too many occurrences of a given non-periodic (aperiodic) pattern. For this test and for the Overlapping Template Matching test, an m-bit window is used to search for a specific m-bit pattern. If the pattern is not found, the window slides one bit position. For this test, when the pattern is found, the window is reset to the bit after the found pattern, and the search resumes. The default value of m = 9, recommended by NIST, was used for this test.

8. Overlapping (Periodic) Template Matching Test

The focus of this test is the number of pre-defined target substrings. The purpose of this test is to reject sequences that show deviations from the expected number of runs

of ones of a given length. Note that when there is a deviation from the expected number of ones of a given length, there is also a deviation in the runs of zeroes. Runs of zeroes were not evaluated separately due to a concern about statistical independence among the tests. For this test and for the Non-overlapping Template Matching test, an m-bit window is used to search for a specific m-bit pattern. If the pattern is not found, the window slides one bit position. For this test, when the pattern is found, the window again slides one bit, and the search is resumed. The default value of m = 9, recommended by NIST, was used for this test.

9. Maurer's Universal Statistical Test

The focus of this test is the number of bits between matching patterns. The purpose of the test is to detect whether or not the sequence can be significantly compressed without loss of information. An overly compressible sequence is considered to be non-random.

10. Linear Complexity Test

The focus of this test is the length of a generating feedback register. The purpose of this test is to determine whether or not the sequence is complex enough to be considered random. Random sequences are characterized by a longer feedback register. A short feedback register implies non-randomness. The default value of 500, recommended by NIST, was used for this test.

11. Serial Test

The focus of this test is the frequency of each and every overlapping m-bit pattern across the entire sequence. The purpose of this test is to determine whether the number of occurrences of the 2^m m-bit overlapping patterns is approximately the same as would be expected for a random sequence. The pattern can overlap. The default value of m = 16, recommended by NIST, was used for this test.

12. Approximate Entropy Test

The focus of this test is the frequency of each and every overlapping m-bit pattern. The purpose of the test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and m+1) against the expected result for a random sequence. The default value of m = 10, recommended by NIST, was used for this test.

13. Cumulative Sum (Cusum) Test

The focus of this test is the maximal excursion (from zero) of the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the sequence. The purpose of the test is to determine whether the cumulative sum of the partial sequences occurring in the tested sequence is too large or too small relative to the expected behavior of that cumulative sum for random sequences. This cumulative sum may be considered as a random walk. For a random sequence, the random walk should be near zero. For non-random sequences, the excursions of this random walk away from zero will be too large.

14. Random Excursions Test

The focus of this test is the number of cycles having exactly K visits in a cumulative sum random walk. The cumulative sum random walk is found if partial sums of the (0, 1) sequence are adjusted to (-1, +1). A random excursion of a random walk consists of a sequence of n steps of unit length taken at random that begin at and return to the origin. The purpose of this test is to determine if the number of visits to a state within a random walk exceeds what one would expect for a random sequence.

15. Random Excursions Variant Test

The focus of this test is the number of times that a particular state occurs in a cumulative sum random walk. The purpose of this test is to detect deviations from the expected number of occurrences of various states in the random walk.

TESTING PARAMETERS AND ENVIRONMENT

Subject: Heuristic One-Time Pad Encryption Engine – The algorithm being used as the core of our encryption technology to deterministically generate binary sequence.

Purpose: To assess the maturity of the subject to deterministically generate random number sequence.

Sequences Being Tested:

1. Core-generated binary sequence - Binary sequence generated by the subject.
2. Core XOR Low Density Plaintexts* - Binary sequence was generated as a result of bitwise XOR operation between the core-generated binary sequence and the artificially

generated non-random sequence of highly frequent appearances of zeros as described below.

3. Core XOR High Density Plaintexts** - Binary sequence generated as a result of bitwise XOR operation between the core-generated binary sequence and the artificially generated non-random sequence of highly frequent appearances of ones as described below.

* Low Density Plaintexts consisted of 8,257 blocks as described in [3]. These blocks were formed from one all zero plaintext block, 128 plaintext blocks of a single one and 127 zeroes (the one appearing in each of the possible 128 bit positions), and 8,128 plaintext blocks of two ones and 126 zeroes (the two ones appearing in each combination of two bit positions within the 128-bit positions).

**High Density Plaintexts consisted of 8,257 blocks as described in [3]. These blocks were formed from one all ones plaintext block, 128 plaintext blocks of a single zero and 127 ones (the zero appearing in each of the possible 128 bit positions), and 8,128 plaintext blocks of two zeroes and 126 ones (the two zeroes appearing in each combination of two bit positions within the 128-bit positions).

Testing Strategy

Randomness testing was performed using the following strategy:

a) Input parameters such as the *sequence length*, *sample size*, and *significance level* were fixed for each sample. These parameters were 1,000,000 bits, 1000 binary sequences, and 0.01; respectively, as recommended by NIST [1]. For each binary sequence and each

statistical test, a P-value was reported.

b) For each P-value, a success/failure assessment was made based on whether it exceeded or fell below the pre-selected significance level of 0.01.

c) For each statistical test and each sample, two evaluations were made. First, the proportion of binary sequences in a sample that passed the statistical test was calculated. The P-value for this proportion is equal to the probability of observing a value equal to or greater than the calculated proportion. Second, an additional P-value was calculated, based on a chi-square test (with nine degrees of freedom) applied to the P-values in the entire sample to ensure uniformity.

d) For both measures described in step (c) above, an assessment was made. A sample was considered to have passed a statistical test if it satisfied both the proportion and uniformity assessments.

RESULTS

Core-Generated Binary Sequences (1,000,000 bits x 1,000 Sequences)

The minimum pass rate for each statistical test (except for the random excursion (variant) test) is approximately = 980 for the sample size of 1000 binary sequences that was being tested. The subject has surpassed this minimum pass rate for all the associated tests. The minimum pass rate for the random excursion (variant) test is approximately = 617 for the sample size of 631 binary sequences that was being tested. The subject has surpassed this minimum pass rate. ***In***

summary, the subject has passed all 15 statistical tests defined by the NIST Statistical Test Suite.

Core XOR Low Density Plaintext (1,000,000 bits x 1,000 Sequences) [+]

The minimum pass rate for each statistical test (except for the random excursion (variant) test) is approximately = 908 for the sample size of 1000 binary sequences that was being tested. The subject has surpassed this minimum pass rate for all the associated tests. The minimum pass rate for the random excursion (variant) test is approximately = 599 for the sample size of 613 binary sequences that was being tested. The subject has surpassed this minimum pass rate. ***In summary, the subject has passed all 15 statistical tests defined by the NIST Statistical Test Suite.***

Core XOR High Density Plaintext (1,000,000 bits x 1,000 Sequences) [+]

The minimum pass rate for each statistical test (except for the random excursion (variant) test) is approximately = 980 for the sample size of 1000 binary sequences that was being tested. The subject has surpassed this minimum pass rate for all the associated tests. The minimum pass rate for the random excursion (variant) test is approximately = 599 for the sample size of 613 binary sequences that was being tested. The subject has surpassed this minimum pass rate. ***In summary, the subject has passed all 15 statistical tests defined by the NIST Statistical Test Suite.***

[+] For those tests using high-density plaintexts and low-density plaintexts, the original sequence of plaintexts were directly XOR-ed to the sequence of the core generated binary sequences. That is, no obfuscation techniques such as shuffling have been applied to the plaintext prior to these

tests to assess the strength of the core algorithm. Test results suggest application of such technique is unnecessary even if plaintext exhibits obvious patterns. It is noted here that AES and other stream cipher algorithms being used today require obfuscation of plaintext as a part of their algorithms.

SUMMARY

The core-generated keys have passed all 15 NIST STS tests demonstrating that the core algorithm generates statistically random binary sequences. More importantly, the binary sequences produced by directly XOR-ing the binary sequences generated by the core algorithm and the low-density plaintexts passed all 15 NIST STS tests. Also, the binary sequences produced by directly XOR-ing the binary sequences generated by the core algorithm and the high-density plaintexts passed all 15 NIST STS tests. This clearly demonstrates that our core algorithm can generate statistically random binary sequences out of plaintexts with obvious patterns. Additionally, if these patterns were kept unchanged our core algorithm continues to generate statistically random binary sequences.

NIST STS REFERENCES

1. Computer Resource Center, Random Bit Generation: Guide to the Statistical Tests, May 2016. https://csrc.nist.gov/Projects/Random-Bit-Generation/Documentation-and-Software/Guide-to-the-Statistical-Tests

2. Rukhin A., et al., A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications, Version STS-2.1, NIST Special Publication 800-22rev1a, April, 2010. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

3. Soto, Juan, "Randomness Testing of the Advanced Encryption Standard Candidate Algorithms", U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, 1999. http://csrc.nist.gov/publications/nistir/ir6390.pdf

NIST STS APPENDIX A – RESULT OF THE CORE GENERATED SEQUENCE (1000 SEQUENCES)

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

----

generator is <data2/core.txt>

- -

The minimum pass rate for each statistical test with the exception of the

random excursion (variant) test is approximately = 980 for a

sample size = 1000 binary sequences.

The minimum pass rate for the random excursion (variant) test

is approximately = 617 for a sample size = 631 binary sequences.

For further guidelines construct a probability table using the MAPLE program

provided in the addendum section of the documentation.

- -

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
95	95	99	107	114	107	112	80	88	103	0.317565	988/1000	Frequency
115	98	99	93	90	106	102	107	101	89	0.749884	992/1000	BlockFrequency
86	108	106	102	103	100	93	107	95	100	0.889118	989/1000	CumulativeSums
91	97	105	115	87	128	89	101	102	85	0.058243	991/1000	CumulativeSums
107	95	96	104	107	99	95	102	98	97	0.99178	987/1000	Runs
89	121	94	103	103	119	94	91	78	108	0.058612	993/1000	LongestRun
108	76	109	115	110	98	96	94	91	103	0.217857	988/1000	Rank
107	98	96	105	98	92	112	107	103	82	0.649612	986/1000	FFT
97	91	104	105	114	91	106	98	98	96	0.861264	989/1000	NonOverlappingTemplate
109	108	87	100	95	114	90	97	89	111	0.450297	987/1000	NonOverlappingTemplate
96	98	107	83	108	108	100	85	112	103	0.471146	987/1000	NonOverlappingTemplate
111	106	109	100	100	92	91	105	93	93	0.829047	994/1000	NonOverlappingTemplate
99	97	92	100	102	104	96	107	94	109	0.973055	987/1000	NonOverlappingTemplate
102	107	101	104	101	86	101	114	86	98	0.674543	994/1000	NonOverlappingTemplate
108	114	100	91	99	100	100	99	104	85	0.755819	989/1000	NonOverlappingTemplate
98	107	96	107	106	106	100	99	92	89	0.926487	989/1000	NonOverlappingTemplate
117	95	104	105	87	90	102	101	87	112	0.399442	989/1000	NonOverlappingTemplate
106	78	90	74	118	111	94	102	99	128	0.002322	987/1000	NonOverlappingTemplate
106	94	100	91	103	105	101	90	102	108	0.937919	986/1000	NonOverlappingTemplate
94	98	109	99	100	87	104	100	102	107	0.935716	990/1000	NonOverlappingTemplate
106	107	112	98	107	102	80	95	95	98	0.595549	987/1000	NonOverlappingTemplate
115	96	110	99	99	97	92	82	102	108	0.526105	984/1000	NonOverlappingTemplate
108	107	113	89	117	75	116	82	101	92	0.020408	989/1000	NonOverlappingTemplate
100	120	89	89	100	102	91	111	100	98	0.482707	994/1000	NonOverlappingTemplate
91	97	116	116	84	109	103	94	102	88	0.254411	992/1000	NonOverlappingTemplate
109	99	100	98	84	110	97	104	101	98	0.858002	985/1000	NonOverlappingTemplate

101	105	108	106	89	95	86	97	111	102	0.737915	993/1000	NonOverlappingTemplate
97	101	91	110	111	87	116	89	98	100	0.473064	985/1000	NonOverlappingTemplate
111	82	93	108	91	103	104	107	113	88	0.32985	991/1000	NonOverlappingTemplate
104	97	82	98	103	95	118	96	110	97	0.498313	989/1000	NonOverlappingTemplate
99	96	105	95	106	106	94	98	91	110	0.935716	992/1000	NonOverlappingTemplate
99	109	96	92	91	102	111	99	102	99	0.927677	993/1000	NonOverlappingTemplate
89	95	101	103	105	110	96	93	102	106	0.920383	996/1000	NonOverlappingTemplate
116	103	92	95	101	85	107	99	97	105	0.674543	987/1000	NonOverlappingTemplate
110	107	103	82	99	87	100	114	92	106	0.394195	988/1000	NonOverlappingTemplate
98	113	82	97	96	106	95	103	95	115	0.492436	992/1000	NonOverlappingTemplate
91	79	93	112	108	95	105	111	116	90	0.15991	992/1000	NonOverlappingTemplate
73	117	92	98	114	110	99	85	102	110	0.04687	994/1000	NonOverlappingTemplate
112	88	109	90	118	90	95	109	110	79	0.080519	989/1000	NonOverlappingTemplate
103	84	85	115	100	99	109	95	104	106	0.461612	993/1000	NonOverlappingTemplate
102	100	94	86	113	108	91	98	87	121	0.234373	989/1000	NonOverlappingTemplate
107	89	96	91	97	93	96	118	126	87	0.088226	988/1000	NonOverlappingTemplate
98	118	99	89	91	95	115	89	100	106	0.402962	990/1000	NonOverlappingTemplate
109	89	96	102	104	111	109	109	82	89	0.378705	990/1000	NonOverlappingTemplate
110	98	100	107	98	87	88	114	99	99	0.670396	992/1000	NonOverlappingTemplate
103	96	108	124	71	98	100	108	97	95	0.06523	990/1000	NonOverlappingTemplate
98	102	107	105	90	94	89	92	102	121	0.486588	992/1000	NonOverlappingTemplate
96	92	91	97	106	103	105	109	114	87	0.651693	991/1000	NonOverlappingTemplate
102	106	96	92	93	96	108	92	117	98	0.733899	988/1000	NonOverlappingTemplate
106	86	98	104	88	101	105	101	108	103	0.837781	987/1000	NonOverlappingTemplate
109	93	120	111	86	102	102	95	98	84	0.249284	987/1000	NonOverlappingTemplate
108	119	99	97	110	90	107	79	104	87	0.158133	985/1000	NonOverlappingTemplate
98	94	98	94	115	105	100	94	106	96	0.899171	992/1000	NonOverlappingTemplate
92	91	114	93	105	94	100	89	118	104	0.426272	992/1000	NonOverlappingTemplate

103	89	108	104	85	115	109	99	104	84	0.339271	990/1000	NonOverlappingTemplate
93	114	98	101	98	103	107	97	102	87	0.839507	989/1000	NonOverlappingTemplate
86	98	113	89	100	99	101	107	100	107	0.749884	995/1000	NonOverlappingTemplate
89	117	95	107	102	95	99	100	100	96	0.807412	989/1000	NonOverlappingTemplate
95	91	105	97	95	109	98	98	102	110	0.936823	993/1000	NonOverlappingTemplate
107	87	90	98	92	101	108	103	109	105	0.773405	988/1000	NonOverlappingTemplate
113	106	100	101	99	99	107	94	90	91	0.856359	987/1000	NonOverlappingTemplate
101	110	82	95	106	108	97	98	106	97	0.731886	991/1000	NonOverlappingTemplate
100	104	96	96	93	102	94	106	99	110	0.973718	993/1000	NonOverlappingTemplate
93	90	100	108	102	103	77	114	104	109	0.313041	994/1000	NonOverlappingTemplate
131	106	101	101	81	102	93	92	94	99	0.087162	988/1000	NonOverlappingTemplate
104	107	98	88	106	96	110	91	99	101	0.877083	992/1000	NonOverlappingTemplate
106	99	90	96	105	87	98	110	109	100	0.805569	992/1000	NonOverlappingTemplate
108	102	90	75	115	104	108	90	109	99	0.171867	997/1000	NonOverlappingTemplate
97	94	108	113	98	99	108	92	105	86	0.707513	993/1000	NonOverlappingTemplate
96	90	95	107	117	106	106	90	91	102	0.599693	990/1000	NonOverlappingTemplate
97	83	114	94	106	113	99	115	78	101	0.106877	990/1000	NonOverlappingTemplate
100	111	108	106	91	101	90	106	102	85	0.670396	992/1000	NonOverlappingTemplate
102	105	121	101	88	82	106	91	101	103	0.299736	988/1000	NonOverlappingTemplate
101	107	96	105	85	108	98	91	106	103	0.825505	990/1000	NonOverlappingTemplate
99	99	107	94	98	103	114	100	99	87	0.862883	995/1000	NonOverlappingTemplate
97	88	111	108	111	112	89	91	88	105	0.371941	986/1000	NonOverlappingTemplate
115	86	87	93	110	102	96	98	108	105	0.482707	984/1000	NonOverlappingTemplate
104	106	103	94	85	97	107	104	90	110	0.743915	995/1000	NonOverlappingTemplate
86	121	110	97	103	105	92	104	97	85	0.279844	991/1000	NonOverlappingTemplate
103	97	78	88	105	108	120	98	103	100	0.244236	992/1000	NonOverlappingTemplate
85	89	106	109	94	98	95	100	109	115	0.500279	988/1000	NonOverlappingTemplate
97	96	96	115	97	101	94	103	110	91	0.832561	988/1000	NonOverlappingTemplate

97	91	104	106	113	90	108	97	98	96	0.830808	989/1000	NonOverlappingTemplate
79	98	97	102	96	113	106	105	114	90	0.350485	995/1000	NonOverlappingTemplate
105	97	103	123	94	90	91	89	113	95	0.272977	986/1000	NonOverlappingTemplate
84	103	110	103	103	100	101	82	96	118	0.313041	991/1000	NonOverlappingTemplate
106	93	90	107	110	107	89	95	102	101	0.80372	991/1000	NonOverlappingTemplate
110	107	98	94	92	105	103	98	97	96	0.957612	992/1000	NonOverlappingTemplate
98	94	97	93	112	102	113	96	94	101	0.861264	988/1000	NonOverlappingTemplate
104	97	97	89	94	103	121	93	93	109	0.514124	989/1000	NonOverlappingTemplate
118	81	98	98	96	89	115	109	98	98	0.24675	987/1000	NonOverlappingTemplate
90	102	100	94	105	94	109	107	113	86	0.641284	990/1000	NonOverlappingTemplate
95	106	103	90	106	95	120	87	90	108	0.380407	988/1000	NonOverlappingTemplate
99	96	94	99	105	99	97	111	100	100	0.989786	993/1000	NonOverlappingTemplate
103	86	111	99	120	88	92	104	94	103	0.353733	986/1000	NonOverlappingTemplate
86	101	87	99	93	99	94	118	117	106	0.274341	997/1000	NonOverlappingTemplate
114	93	84	97	116	104	103	86	93	110	0.251837	986/1000	NonOverlappingTemplate
96	112	97	109	95	113	105	91	84	98	0.524101	993/1000	NonOverlappingTemplate
110	85	91	103	89	105	104	116	107	90	0.365253	989/1000	NonOverlappingTemplate
104	99	94	90	109	105	98	101	91	109	0.893482	994/1000	NonOverlappingTemplate
90	97	112	82	111	112	107	91	96	102	0.357	991/1000	NonOverlappingTemplate
110	107	89	110	84	84	101	115	93	107	0.209948	992/1000	NonOverlappingTemplate
99	111	94	105	84	105	118	87	106	91	0.293952	991/1000	NonOverlappingTemplate
98	106	92	116	88	112	107	87	90	104	0.365253	987/1000	NonOverlappingTemplate
96	91	97	107	99	93	114	84	101	118	0.365253	994/1000	NonOverlappingTemplate
108	122	90	103	94	100	87	97	107	92	0.363593	988/1000	NonOverlappingTemplate
100	92	111	98	91	103	90	113	89	113	0.496351	994/1000	NonOverlappingTemplate
105	112	99	110	109	95	86	80	99	105	0.352107	991/1000	NonOverlappingTemplate
100	84	121	100	98	108	108	88	92	101	0.320607	990/1000	NonOverlappingTemplate
86	110	83	115	112	102	116	98	87	91	0.100109	991/1000	NonOverlappingTemplate

100	115	97	101	96	97	108	103	102	81	0.639202	997/1000	NonOverlappingTemplate
100	87	112	95	116	107	113	96	91	83	0.214439	987/1000	NonOverlappingTemplate
105	108	95	95	108	98	96	104	93	98	0.967382	987/1000	NonOverlappingTemplate
91	120	105	93	99	98	104	102	96	92	0.678686	993/1000	NonOverlappingTemplate
98	98	103	111	109	96	104	85	97	99	0.846338	989/1000	NonOverlappingTemplate
99	72	97	95	98	108	105	104	118	104	0.177628	993/1000	NonOverlappingTemplate
105	104	101	93	119	93	88	90	106	101	0.552383	990/1000	NonOverlappingTemplate
118	89	100	87	104	80	96	116	108	102	0.133404	986/1000	NonOverlappingTemplate
104	96	91	118	85	116	98	99	99	94	0.383827	990/1000	NonOverlappingTemplate
102	105	97	106	115	84	108	103	94	86	0.474986	985/1000	NonOverlappingTemplate
119	102	106	80	106	103	100	100	96	88	0.34565	994/1000	NonOverlappingTemplate
90	84	102	119	103	96	107	100	92	107	0.429923	987/1000	NonOverlappingTemplate
92	105	105	118	93	106	99	94	90	98	0.674543	991/1000	NonOverlappingTemplate
103	96	95	97	92	112	86	108	106	105	0.751866	993/1000	NonOverlappingTemplate
93	110	91	113	119	98	93	86	91	106	0.258307	993/1000	NonOverlappingTemplate
93	98	102	100	97	103	100	83	116	108	0.653773	991/1000	NonOverlappingTemplate
98	104	86	107	101	106	118	86	104	90	0.402962	991/1000	NonOverlappingTemplate
102	102	116	87	101	93	113	104	90	92	0.502247	988/1000	NonOverlappingTemplate
108	100	92	102	108	119	108	81	86	96	0.216713	992/1000	NonOverlappingTemplate
117	93	92	103	99	101	88	105	100	102	0.753844	989/1000	NonOverlappingTemplate
95	94	102	96	118	76	104	118	99	98	0.15119	993/1000	NonOverlappingTemplate
104	93	87	106	89	115	110	106	84	106	0.316052	987/1000	NonOverlappingTemplate
95	95	100	107	91	94	109	115	104	90	0.701366	993/1000	NonOverlappingTemplate
100	100	86	98	94	96	108	100	109	109	0.853049	990/1000	NonOverlappingTemplate
84	117	96	83	117	114	93	91	90	115	0.036352	988/1000	NonOverlappingTemplate
99	101	99	117	88	96	101	118	84	97	0.317565	995/1000	NonOverlappingTemplate
111	99	100	115	87	88	92	96	110	102	0.490483	991/1000	NonOverlappingTemplate
108	95	123	97	93	104	110	98	83	89	0.209948	991/1000	NonOverlappingTemplate

103	105	102	86	96	110	106	87	104	101	0.767582	989/1000	NonOverlappingTemplate
96	103	106	106	94	101	88	113	98	95	0.854708	992/1000	NonOverlappingTemplate
96	101	99	112	113	95	86	109	94	95	0.643366	989/1000	NonOverlappingTemplate
95	100	102	115	82	104	95	105	94	108	0.591409	991/1000	NonOverlappingTemplate
98	96	108	106	94	102	113	95	94	94	0.893482	991/1000	NonOverlappingTemplate
96	105	94	110	82	106	107	102	97	101	0.739918	989/1000	NonOverlappingTemplate
93	103	106	101	89	105	107	104	104	88	0.862883	989/1000	NonOverlappingTemplate
92	118	117	83	108	90	105	90	102	95	0.169981	993/1000	NonOverlappingTemplate
102	115	88	86	104	100	111	95	94	105	0.542228	991/1000	NonOverlappingTemplate
98	99	88	106	103	93	103	112	105	93	0.859637	985/1000	NonOverlappingTemplate
107	101	103	90	104	96	109	100	87	103	0.875539	990/1000	NonOverlappingTemplate
116	82	114	88	100	106	121	105	90	78	0.017546	987/1000	NonOverlappingTemplate
100	86	102	104	106	111	117	101	77	96	0.208837	989/1000	NonOverlappingTemplate
83	103	91	98	106	104	97	99	99	120	0.488534	992/1000	NonOverlappingTemplate
105	85	110	101	99	93	102	106	100	99	0.881662	992/1000	NonOverlappingTemplate
101	95	100	96	86	103	96	110	93	120	0.5221	994/1000	NonOverlappingTemplate
87	99	89	101	80	107	104	113	121	99	0.134172	992/1000	NonOverlappingTemplate
97	96	95	116	97	100	95	103	110	91	0.807412	988/1000	NonOverlappingTemplate
123	75	85	98	105	97	123	96	104	94	0.017068	994/1000	OverlappingTemplate
111	102	91	89	93	102	89	109	97	117	0.455937	990/1000	Universal
107	99	92	82	112	105	110	95	93	105	0.528111	986/1000	ApproximateEntropy
59	51	60	59	64	75	68	72	64	59	0.617296	623/631	RandomExcursions
69	66	76	62	63	49	61	66	61	58	0.617296	626/631	RandomExcursions
63	71	63	53	69	79	76	52	49	56	0.072735	625/631	RandomExcursions
67	58	65	61	69	70	61	71	51	58	0.750985	622/631	RandomExcursions
55	54	60	72	74	64	67	65	61	59	0.709396	628/631	RandomExcursions
72	62	57	62	62	78	51	57	61	69	0.44021	628/631	RandomExcursions
62	54	72	57	63	63	58	61	78	63	0.617296	625/631	RandomExcursions

65	68	66	60	54	73	55	68	68	54	0.666838	624/631	RandomExcursions
62	75	61	58	57	68	67	56	66	61	0.837067	623/631	RandomExcursionsVariant
60	70	66	58	50	77	80	53	66	51	0.069925	624/631	RandomExcursionsVariant
63	65	62	71	67	57	68	60	69	49	0.72554	626/631	RandomExcursionsVariant
69	62	66	64	59	68	64	62	65	52	0.945667	625/631	RandomExcursionsVariant
67	64	64	63	66	66	58	55	61	67	0.985752	627/631	RandomExcursionsVariant
67	59	62	57	63	66	60	76	54	67	0.778883	625/631	RandomExcursionsVariant
73	52	56	64	65	65	71	57	64	64	0.735143	626/631	RandomExcursionsVariant
66	49	79	65	62	60	61	75	53	61	0.24052	624/631	RandomExcursionsVariant
55	59	68	74	60	53	61	76	53	72	0.280086	621/631	RandomExcursionsVariant
46	64	77	49	73	71	67	62	62	60	0.127498	626/631	RandomExcursionsVariant
51	65	75	57	60	62	69	72	66	54	0.470113	623/631	RandomExcursionsVariant
66	43	65	62	67	58	81	62	60	67	0.172922	623/631	RandomExcursionsVariant
66	42	64	61	75	65	56	63	77	62	0.142216	625/631	RandomExcursionsVariant
66	59	58	65	67	72	62	58	63	61	0.969117	624/631	RandomExcursionsVariant
64	59	76	55	66	72	66	61	52	60	0.568055	623/631	RandomExcursionsVariant
65	54	75	60	64	62	56	60	63	72	0.738329	618/631	RandomExcursionsVariant
61	62	66	72	58	62	42	67	72	69	0.273539	624/631	RandomExcursionsVariant
56	65	70	72	68	48	58	51	67	76	0.186968	627/631	RandomExcursionsVariant
92	97	104	110	82	107	90	112	103	103	0.510153	990/1000	Serial
104	85	100	83	107	114	103	99	98	107	0.496351	994/1000	Serial
94	106	101	114	98	89	100	101	97	100	0.90876	994/1000	LinearComplexity

NIST STS APPENDIX B – RESULT OF THE CORE XOR LOW DENSITY PLAINTEXTS SEQUENCE (1000 SEQUENCES)

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

---

generator is <data2/low_density_xor.txt>

- -

The minimum pass rate for each statistical test with the exception of the

random excursion (variant) test is approximately = 980 for a

sample size = 1000 binary sequences.

The minimum pass rate for the random excursion (variant) test

is approximately = 599 for a sample size = 613 binary sequences.

For further guidelines construct a probability table using the MAPLE program

provided in the addendum section of the documentation.

- -

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
101	100	94	91	106	108	99	97	103	101	0.983938	988/1000	Frequency
103	107	109	97	94	106	88	104	100	92	0.880145	990/1000	BlockFrequency
93	122	96	82	104	113	93	109	93	95	0.180568	991/1000	CumulativeSums
91	98	95	95	95	117	99	109	93	108	0.6952	993/1000	CumulativeSums
116	100	113	98	82	113	101	84	105	88	0.142062	984/1000	Runs
112	88	109	77	105	91	102	100	103	113	0.221317	994/1000	LongestRun
98	89	97	97	98	104	95	118	102	102	0.816537	992/1000	Rank
111	90	110	101	96	92	103	109	100	88	0.703417	987/1000	FFT
97	94	113	90	92	80	93	124	104	113	0.06943	995/1000	NonOverlappingTemplate
104	94	116	99	106	109	92	103	91	86	0.558502	988/1000	NonOverlappingTemplate
125	90	104	96	96	89	91	113	102	94	0.22248	983/1000	NonOverlappingTemplate
110	107	100	96	91	93	101	108	106	88	0.798139	989/1000	NonOverlappingTemplate
87	118	83	97	92	92	109	112	110	100	0.189625	987/1000	NonOverlappingTemplate
103	102	106	97	96	93	99	101	103	100	0.998169	984/1000	NonOverlappingTemplate
103	101	109	102	97	111	101	74	106	96	0.38899	985/1000	NonOverlappingTemplate
99	113	85	88	98	107	107	116	89	98	0.33297	990/1000	NonOverlappingTemplate
95	90	107	113	102	105	97	92	104	95	0.846338	987/1000	NonOverlappingTemplate
114	98	87	99	98	90	89	100	128	97	0.126658	985/1000	NonOverlappingTemplate
107	103	119	84	94	86	106	87	104	110	0.197981	986/1000	NonOverlappingTemplate
99	99	85	91	109	102	95	106	105	109	0.779188	988/1000	NonOverlappingTemplate
100	87	115	125	106	101	94	93	96	83	0.106877	992/1000	NonOverlappingTemplate
112	108	101	96	100	105	96	92	92	98	0.912724	992/1000	NonOverlappingTemplate
102	93	93	104	104	93	99	110	94	108	0.921624	992/1000	NonOverlappingTemplate
83	113	101	106	109	91	107	92	97	101	0.55442	997/1000	NonOverlappingTemplate
89	99	90	95	107	97	124	105	90	104	0.33297	993/1000	NonOverlappingTemplate
115	100	79	85	95	119	92	115	99	101	0.073872	992/1000	NonOverlappingTemplate

100	103	78	111	109	99	112	96	93	99	0.431754	987/1000	NonOverlappingTemplate
100	85	94	108	102	96	97	103	100	115	0.751866	992/1000	NonOverlappingTemplate
115	98	90	94	93	107	94	110	102	97	0.727851	990/1000	NonOverlappingTemplate
93	105	101	93	100	91	103	93	104	117	0.771469	987/1000	NonOverlappingTemplate
106	99	87	109	106	90	111	102	86	104	0.574903	990/1000	NonOverlappingTemplate
110	91	98	97	118	100	104	95	97	90	0.670396	993/1000	NonOverlappingTemplate
86	108	94	95	109	87	92	114	102	113	0.347257	996/1000	NonOverlappingTemplate
93	93	107	106	97	85	102	117	99	101	0.624627	990/1000	NonOverlappingTemplate
96	105	115	89	100	103	97	101	101	93	0.870856	995/1000	NonOverlappingTemplate
96	98	82	91	110	112	98	100	102	111	0.536163	991/1000	NonOverlappingTemplate
103	95	96	100	97	103	102	105	95	104	0.997943	993/1000	NonOverlappingTemplate
84	82	111	103	116	92	101	101	110	100	0.254411	992/1000	NonOverlappingTemplate
97	103	110	106	106	86	86	118	101	87	0.292519	991/1000	NonOverlappingTemplate
104	107	85	114	110	100	93	87	95	105	0.480771	989/1000	NonOverlappingTemplate
100	100	105	90	96	101	100	105	110	93	0.957612	994/1000	NonOverlappingTemplate
94	85	107	104	92	119	111	93	100	95	0.39594	992/1000	NonOverlappingTemplate
83	109	123	81	112	111	111	75	90	105	0.004365	990/1000	NonOverlappingTemplate
102	99	97	107	86	104	115	104	97	89	0.693142	991/1000	NonOverlappingTemplate
105	109	85	94	71	99	122	93	119	103	0.012128	990/1000	NonOverlappingTemplate
115	109	80	117	91	100	100	92	93	103	0.214439	988/1000	NonOverlappingTemplate
110	82	97	105	119	96	90	120	88	93	0.083526	991/1000	NonOverlappingTemplate
95	97	100	115	91	96	96	107	111	92	0.733899	991/1000	NonOverlappingTemplate
116	84	97	110	100	88	91	110	110	94	0.288249	982/1000	NonOverlappingTemplate
103	82	107	97	98	96	110	102	95	110	0.699313	991/1000	NonOverlappingTemplate
115	106	95	93	111	84	98	101	109	88	0.399442	990/1000	NonOverlappingTemplate
112	100	104	109	93	90	93	100	105	94	0.834308	989/1000	NonOverlappingTemplate
93	94	83	108	102	100	105	97	119	99	0.496351	991/1000	NonOverlappingTemplate
92	111	90	105	99	104	91	112	89	107	0.614226	995/1000	NonOverlappingTemplate

98	91	89	97	102	116	98	100	104	105	0.816537	992/1000	NonOverlappingTemplate
93	103	111	101	92	104	102	98	116	80	0.415422	994/1000	NonOverlappingTemplate
85	88	99	97	103	109	104	109	108	98	0.705466	987/1000	NonOverlappingTemplate
114	97	106	104	103	78	98	111	102	87	0.313041	991/1000	NonOverlappingTemplate
103	98	101	102	102	96	93	107	85	113	0.807412	992/1000	NonOverlappingTemplate
107	107	92	88	99	87	103	92	111	114	0.469232	988/1000	NonOverlappingTemplate
120	103	105	83	113	97	101	97	95	86	0.254411	986/1000	NonOverlappingTemplate
105	95	101	85	91	101	109	108	105	100	0.809249	985/1000	NonOverlappingTemplate
95	107	103	105	110	89	91	89	103	108	0.735908	994/1000	NonOverlappingTemplate
87	106	109	87	93	106	104	103	113	92	0.536163	989/1000	NonOverlappingTemplate
108	115	102	96	113	62	107	94	103	100	0.01695	990/1000	NonOverlappingTemplate
127	91	101	107	110	88	96	82	89	109	0.057875	986/1000	NonOverlappingTemplate
97	90	106	88	94	105	108	116	111	85	0.337688	991/1000	NonOverlappingTemplate
103	96	99	101	88	99	113	96	102	103	0.930026	992/1000	NonOverlappingTemplate
107	104	100	97	82	102	87	105	106	110	0.603841	992/1000	NonOverlappingTemplate
103	101	88	94	105	99	110	102	99	99	0.95493	990/1000	NonOverlappingTemplate
98	98	113	111	94	83	97	104	111	91	0.484646	989/1000	NonOverlappingTemplate
101	99	116	95	95	92	117	102	103	80	0.293952	991/1000	NonOverlappingTemplate
93	102	96	85	107	106	105	101	98	107	0.869278	991/1000	NonOverlappingTemplate
95	104	103	100	99	90	110	89	117	93	0.626709	988/1000	NonOverlappingTemplate
114	106	98	95	101	86	98	113	102	87	0.53012	988/1000	NonOverlappingTemplate
96	104	100	98	99	91	100	109	105	98	0.986227	990/1000	NonOverlappingTemplate
93	109	91	107	102	97	101	105	97	98	0.959347	987/1000	NonOverlappingTemplate
89	118	108	91	96	88	101	107	99	103	0.524101	985/1000	NonOverlappingTemplate
116	102	86	93	122	94	107	100	87	93	0.16626	991/1000	NonOverlappingTemplate
98	82	94	106	104	114	104	90	92	116	0.313041	991/1000	NonOverlappingTemplate
93	85	93	104	92	122	103	99	110	99	0.352107	985/1000	NonOverlappingTemplate
86	103	131	96	106	103	102	101	86	86	0.062036	997/1000	NonOverlappingTemplate

97	94	113	90	92	80	93	124	103	114	0.06523	995/1000	NonOverlappingTemplate
76	98	112	109	100	78	105	111	104	107	0.090936	997/1000	NonOverlappingTemplate
113	93	103	125	90	107	102	77	86	104	0.041981	980/1000	NonOverlappingTemplate
95	103	117	86	89	84	101	102	109	114	0.225998	988/1000	NonOverlappingTemplate
93	110	109	90	116	100	100	88	92	102	0.536163	992/1000	NonOverlappingTemplate
104	106	111	102	102	97	106	101	90	81	0.649612	992/1000	NonOverlappingTemplate
107	90	109	86	96	117	114	88	90	103	0.224821	989/1000	NonOverlappingTemplate
94	100	94	81	88	110	130	102	95	106	0.058612	991/1000	NonOverlappingTemplate
104	99	111	96	90	89	79	100	104	128	0.063615	991/1000	NonOverlappingTemplate
82	96	100	102	101	110	102	109	95	103	0.775337	988/1000	NonOverlappingTemplate
100	91	103	101	116	103	92	97	90	107	0.761719	992/1000	NonOverlappingTemplate
99	100	84	95	108	102	100	123	91	98	0.380407	985/1000	NonOverlappingTemplate
125	108	93	103	97	91	91	102	96	94	0.371941	982/1000	NonOverlappingTemplate
82	107	92	90	96	104	104	106	117	102	0.424453	993/1000	NonOverlappingTemplate
92	92	115	96	106	97	92	91	120	99	0.383827	988/1000	NonOverlappingTemplate
105	101	102	83	107	111	88	110	97	96	0.576961	990/1000	NonOverlappingTemplate
93	115	93	97	101	108	102	99	95	97	0.886162	995/1000	NonOverlappingTemplate
100	82	107	93	97	102	121	89	117	92	0.141256	989/1000	NonOverlappingTemplate
96	95	105	95	101	113	98	101	102	94	0.961869	984/1000	NonOverlappingTemplate
89	100	111	86	106	112	101	96	90	109	0.518106	992/1000	NonOverlappingTemplate
116	80	112	111	102	101	100	86	103	89	0.185555	985/1000	NonOverlappingTemplate
104	87	117	108	87	94	73	102	117	111	0.02641	990/1000	NonOverlappingTemplate
94	122	97	90	95	106	104	92	102	98	0.55646	991/1000	NonOverlappingTemplate
100	107	102	100	112	89	87	111	88	104	0.566688	997/1000	NonOverlappingTemplate
102	110	83	101	89	111	95	99	110	100	0.572847	992/1000	NonOverlappingTemplate
101	111	82	102	94	122	109	100	91	88	0.17377	994/1000	NonOverlappingTemplate
97	115	103	104	99	84	94	118	90	96	0.357	992/1000	NonOverlappingTemplate
102	93	102	95	98	111	101	99	95	104	0.980883	989/1000	NonOverlappingTemplate

86	110	91	111	99	103	76	106	115	103	0.139655	993/1000	NonOverlappingTemplate
82	97	101	104	91	99	98	107	119	102	0.484646	994/1000	NonOverlappingTemplate
93	93	89	105	112	104	94	105	104	101	0.849708	993/1000	NonOverlappingTemplate
93	105	103	95	96	116	91	108	102	91	0.72987	993/1000	NonOverlappingTemplate
109	102	82	88	115	119	105	94	91	95	0.15991	988/1000	NonOverlappingTemplate
101	94	99	99	102	96	107	101	95	106	0.995373	986/1000	NonOverlappingTemplate
102	113	99	97	101	92	107	101	88	100	0.881662	992/1000	NonOverlappingTemplate
96	96	109	94	107	100	104	97	99	98	0.986227	987/1000	NonOverlappingTemplate
101	90	89	92	98	100	118	106	94	112	0.504219	991/1000	NonOverlappingTemplate
112	91	102	101	117	87	91	104	93	102	0.496351	988/1000	NonOverlappingTemplate
110	103	102	78	102	102	103	97	102	101	0.711601	989/1000	NonOverlappingTemplate
94	101	85	110	101	101	98	101	115	94	0.709558	992/1000	NonOverlappingTemplate
104	96	97	82	115	118	110	91	75	112	0.028434	988/1000	NonOverlappingTemplate
95	112	97	102	101	90	93	100	108	102	0.911413	992/1000	NonOverlappingTemplate
88	90	95	99	117	87	104	117	109	94	0.242986	987/1000	NonOverlappingTemplate
114	107	93	77	102	96	110	98	97	106	0.357	986/1000	NonOverlappingTemplate
91	120	90	104	93	103	112	108	81	98	0.197981	989/1000	NonOverlappingTemplate
94	97	104	114	111	91	96	93	96	104	0.783019	990/1000	NonOverlappingTemplate
89	114	95	107	98	100	104	107	105	81	0.488534	989/1000	NonOverlappingTemplate
82	112	95	111	115	102	102	107	80	94	0.148653	991/1000	NonOverlappingTemplate
99	105	110	108	111	87	104	87	97	92	0.59762	991/1000	NonOverlappingTemplate
90	111	102	82	95	101	108	109	104	98	0.595549	990/1000	NonOverlappingTemplate
103	96	95	106	92	112	113	101	89	93	0.705466	985/1000	NonOverlappingTemplate
96	96	112	106	97	100	92	117	91	93	0.632955	992/1000	NonOverlappingTemplate
96	98	95	90	104	108	106	110	88	105	0.807412	993/1000	NonOverlappingTemplate
88	115	113	104	104	108	83	88	99	98	0.295391	993/1000	NonOverlappingTemplate
114	97	89	97	80	112	101	109	91	110	0.248014	985/1000	NonOverlappingTemplate
106	107	105	104	100	96	87	96	107	92	0.883171	984/1000	NonOverlappingTemplate

97	94	108	99	102	114	90	94	95	107	0.816537	991/1000	NonOverlappingTemplate
101	117	100	109	105	90	99	97	88	94	0.651693	992/1000	NonOverlappingTemplate
107	96	104	102	99	99	105	110	83	95	0.81108	989/1000	NonOverlappingTemplate
114	109	94	113	96	96	106	88	98	86	0.442831	986/1000	NonOverlappingTemplate
100	111	111	110	106	84	104	95	80	99	0.292519	989/1000	NonOverlappingTemplate
103	115	100	83	90	78	98	98	105	130	0.015598	992/1000	NonOverlappingTemplate
106	92	92	103	95	120	126	79	89	98	0.030806	990/1000	NonOverlappingTemplate
97	91	104	87	98	107	111	116	94	95	0.568739	987/1000	NonOverlappingTemplate
105	97	100	113	109	100	99	104	88	85	0.668321	993/1000	NonOverlappingTemplate
99	113	84	97	109	102	96	99	102	99	0.796268	992/1000	NonOverlappingTemplate
88	113	102	98	88	113	97	106	92	103	0.583145	994/1000	NonOverlappingTemplate
99	106	99	118	94	88	94	104	102	96	0.725829	992/1000	NonOverlappingTemplate
111	107	107	99	120	84	96	96	93	87	0.258307	992/1000	NonOverlappingTemplate
91	101	104	99	95	99	93	107	103	108	0.96586	992/1000	NonOverlappingTemplate
111	98	92	122	90	90	100	108	80	109	0.116065	989/1000	NonOverlappingTemplate
109	112	99	80	105	90	96	107	103	99	0.508172	986/1000	NonOverlappingTemplate
101	91	84	118	99	94	94	135	85	99	0.009333	989/1000	NonOverlappingTemplate
86	104	131	95	106	103	102	100	87	86	0.064418	997/1000	NonOverlappingTemplate
112	91	92	86	105	109	100	92	101	112	0.534146	988/1000	OverlappingTemplate
121	94	99	92	110	93	95	102	103	91	0.524101	985/1000	Universal
92	109	91	109	87	108	106	101	88	109	0.532132	988/1000	ApproximateEntropy
51	55	66	70	61	69	72	47	55	67	0.249701	607/613	RandomExcursions
68	48	56	77	59	66	75	57	46	61	0.070744	604/613	RandomExcursions
61	68	63	63	65	56	59	53	59	66	0.955982	609/613	RandomExcursions
72	71	57	53	70	48	60	60	52	70	0.22948	608/613	RandomExcursions
63	65	56	54	55	61	56	74	67	62	0.748093	601/613	RandomExcursions
83	64	51	58	58	54	64	58	58	65	0.241456	603/613	RandomExcursions
60	72	48	51	71	59	78	55	64	55	0.116054	608/613	RandomExcursions

53	56	59	64	58	67	57	72	62	65	0.839987	607/613	RandomExcursions
68	57	69	59	54	68	61	55	62	60	0.88243	607/613	RandomExcursionsVariant
64	67	72	58	55	67	57	60	47	66	0.53911	607/613	RandomExcursionsVariant
67	69	64	63	71	54	60	60	52	53	0.667811	605/613	RandomExcursionsVariant
68	65	71	67	59	56	63	64	53	47	0.509568	604/613	RandomExcursionsVariant
63	74	61	58	59	76	62	54	53	53	0.379067	603/613	RandomExcursionsVariant
62	68	72	57	55	71	51	60	56	61	0.599625	603/613	RandomExcursionsVariant
66	60	62	83	59	61	52	54	59	57	0.282511	605/613	RandomExcursionsVariant
71	65	57	67	54	64	54	61	60	60	0.872348	605/613	RandomExcursionsVariant
62	63	50	53	70	68	64	75	55	53	0.330947	607/613	RandomExcursionsVariant
63	47	46	61	79	57	60	66	61	73	0.082315	606/613	RandomExcursionsVariant
57	50	54	67	65	64	67	47	77	65	0.196314	608/613	RandomExcursionsVariant
59	52	59	45	72	71	67	66	65	57	0.289435	602/613	RandomExcursionsVariant
61	52	52	62	71	56	66	75	62	56	0.458724	602/613	RandomExcursionsVariant
62	55	62	55	43	56	70	75	57	78	0.057593	606/613	RandomExcursionsVariant
64	61	54	62	48	51	54	79	57	83	0.019935	606/613	RandomExcursionsVariant
66	52	59	55	52	80	68	47	67	67	0.095539	605/613	RandomExcursionsVariant
67	50	57	69	60	56	65	58	60	71	0.701669	607/613	RandomExcursionsVariant
61	57	71	59	53	61	55	61	70	65	0.816833	608/613	RandomExcursionsVariant
102	93	103	107	102	88	98	93	100	114	0.827279	994/1000	Serial
110	94	93	101	114	104	80	84	106	114	0.169044	990/1000	Serial
82	109	101	110	124	97	102	90	99	86	0.1252	991/1000	LinearComplexity

NIST STS APPENDIX C – RESULT OF THE CORE XOR HIGH DENSITY PLAINTEXTS SEQUENCE (1000 SEQUENCES)

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

---

generator is <data2/high_density_xor.txt>

- -

The minimum pass rate for each statistical test with the exception of the

random excursion (variant) test is approximately = 980 for a

sample size = 1000 binary sequences.

The minimum pass rate for the random excursion (variant) test

is approximately = 599 for a sample size = 613 binary sequences.

For further guidelines construct a probability table using the MAPLE program

provided in the addendum section of the documentation.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
101	100	94	91	106	108	99	97	103	101	0.983938	988/1000	Frequency
103	107	109	97	94	106	88	104	100	92	0.880145	990/1000	BlockFrequency
93	122	96	82	104	113	93	109	93	95	0.180568	991/1000	CumulativeSums
91	98	95	95	95	117	99	109	93	108	0.6952	993/1000	CumulativeSums
116	100	113	98	82	113	101	84	105	88	0.142062	984/1000	Runs
101	95	97	90	94	111	104	118	94	96	0.653773	990/1000	LongestRun
100	79	110	96	96	103	103	123	110	80	0.062821	994/1000	Rank
111	90	110	101	96	92	103	109	100	88	0.703417	987/1000	FFT
86	104	131	95	106	103	102	100	87	86	0.064418	997/1000	NonOverlappingTemplate
101	91	84	118	99	94	94	135	85	99	0.009333	989/1000	NonOverlappingTemplate
109	112	99	80	105	90	96	107	103	99	0.508172	986/1000	NonOverlappingTemplate
111	98	92	122	90	90	100	108	80	109	0.116065	989/1000	NonOverlappingTemplate
91	101	104	99	95	99	93	107	103	108	0.96586	992/1000	NonOverlappingTemplate
111	107	107	99	120	84	96	96	93	87	0.258307	992/1000	NonOverlappingTemplate
99	106	99	118	94	88	94	104	102	96	0.725829	992/1000	NonOverlappingTemplate
88	113	102	98	88	113	97	106	92	103	0.583145	994/1000	NonOverlappingTemplate
99	113	84	97	109	102	96	99	102	99	0.796268	992/1000	NonOverlappingTemplate
105	97	100	113	109	100	99	104	88	85	0.668321	993/1000	NonOverlappingTemplate
97	91	104	87	98	107	111	116	94	95	0.568739	987/1000	NonOverlappingTemplate
106	92	92	103	95	120	126	79	89	98	0.030806	990/1000	NonOverlappingTemplate
103	115	100	83	90	78	98	98	105	130	0.015598	992/1000	NonOverlappingTemplate
100	111	111	110	106	84	104	95	80	99	0.292519	989/1000	NonOverlappingTemplate
114	109	94	113	96	96	106	88	98	86	0.442831	986/1000	NonOverlappingTemplate
107	96	104	102	99	99	105	110	83	95	0.81108	989/1000	NonOverlappingTemplate
101	117	100	109	105	90	99	97	88	94	0.651693	992/1000	NonOverlappingTemplate
97	94	108	99	102	114	90	94	95	107	0.816537	991/1000	NonOverlappingTemplate

106	107	105	104	100	96	87	96	107	92	0.883171	984/1000	NonOverlappingTemplate
114	97	89	97	80	112	101	109	91	110	0.248014	985/1000	NonOverlappingTemplate
88	115	113	104	104	108	83	88	99	98	0.295391	993/1000	NonOverlappingTemplate
96	98	95	90	104	108	106	110	88	105	0.807412	993/1000	NonOverlappingTemplate
96	96	112	106	97	100	92	117	91	93	0.632955	992/1000	NonOverlappingTemplate
103	96	95	106	92	112	113	101	89	93	0.705466	985/1000	NonOverlappingTemplate
90	111	102	82	95	101	108	109	104	98	0.595549	990/1000	NonOverlappingTemplate
99	105	110	108	111	87	104	87	97	92	0.59762	991/1000	NonOverlappingTemplate
82	112	95	111	115	102	102	107	80	94	0.148653	991/1000	NonOverlappingTemplate
89	114	95	107	98	100	104	107	105	81	0.488534	989/1000	NonOverlappingTemplate
94	97	104	114	111	91	96	93	96	104	0.783019	990/1000	NonOverlappingTemplate
91	120	90	104	93	103	112	108	81	98	0.197981	989/1000	NonOverlappingTemplate
114	107	93	77	102	96	110	98	97	106	0.357	986/1000	NonOverlappingTemplate
88	90	95	99	117	87	104	117	109	94	0.242986	987/1000	NonOverlappingTemplate
95	112	97	102	101	90	93	100	108	102	0.911413	992/1000	NonOverlappingTemplate
104	96	97	82	115	118	110	91	75	112	0.028434	988/1000	NonOverlappingTemplate
94	101	85	110	101	101	98	101	115	94	0.709558	992/1000	NonOverlappingTemplate
110	103	102	78	102	102	103	97	102	101	0.711601	989/1000	NonOverlappingTemplate
112	91	102	101	117	87	91	104	93	102	0.496351	988/1000	NonOverlappingTemplate
101	90	89	92	98	100	118	106	94	112	0.504219	991/1000	NonOverlappingTemplate
96	96	109	94	107	100	104	97	99	98	0.986227	987/1000	NonOverlappingTemplate
102	113	99	97	101	92	107	101	88	100	0.881662	992/1000	NonOverlappingTemplate
101	94	99	99	102	96	107	101	95	106	0.995373	986/1000	NonOverlappingTemplate
109	102	82	88	115	119	105	94	91	95	0.15991	988/1000	NonOverlappingTemplate
93	105	103	95	96	116	91	108	102	91	0.72987	993/1000	NonOverlappingTemplate
93	93	89	105	112	104	94	105	104	101	0.849708	993/1000	NonOverlappingTemplate
82	97	101	104	91	99	98	107	119	102	0.484646	994/1000	NonOverlappingTemplate
86	110	91	111	99	103	76	106	115	103	0.139655	993/1000	NonOverlappingTemplate

102	93	102	95	98	111	101	99	95	104	0.980883	989/1000	NonOverlappingTemplate
97	115	103	104	99	84	94	118	90	96	0.357	992/1000	NonOverlappingTemplate
101	111	82	102	94	122	109	100	91	88	0.17377	994/1000	NonOverlappingTemplate
102	110	83	101	89	111	95	99	110	100	0.572847	992/1000	NonOverlappingTemplate
100	107	102	100	112	89	87	111	88	104	0.566688	997/1000	NonOverlappingTemplate
94	122	97	90	95	106	104	92	102	98	0.55646	991/1000	NonOverlappingTemplate
104	87	117	108	87	94	73	102	117	111	0.02641	990/1000	NonOverlappingTemplate
116	80	112	111	102	101	100	86	103	89	0.185555	985/1000	NonOverlappingTemplate
89	100	111	86	106	112	101	96	90	109	0.518106	992/1000	NonOverlappingTemplate
96	95	105	95	101	113	98	101	102	94	0.961869	984/1000	NonOverlappingTemplate
100	82	107	93	97	102	121	89	117	92	0.141256	989/1000	NonOverlappingTemplate
93	115	93	97	101	108	102	99	95	97	0.886162	995/1000	NonOverlappingTemplate
105	101	102	83	107	111	88	110	97	96	0.576961	990/1000	NonOverlappingTemplate
92	92	115	96	106	97	92	91	120	99	0.383827	988/1000	NonOverlappingTemplate
82	107	92	90	96	104	104	106	117	102	0.424453	993/1000	NonOverlappingTemplate
125	108	93	103	97	91	91	102	96	94	0.371941	982/1000	NonOverlappingTemplate
99	100	84	95	108	102	100	123	91	98	0.380407	985/1000	NonOverlappingTemplate
100	91	103	101	116	103	92	97	90	107	0.761719	992/1000	NonOverlappingTemplate
82	96	100	102	101	110	102	109	95	103	0.775337	988/1000	NonOverlappingTemplate
104	99	111	96	90	89	79	100	104	128	0.063615	991/1000	NonOverlappingTemplate
94	100	94	81	88	110	130	102	95	106	0.058612	991/1000	NonOverlappingTemplate
107	90	109	86	96	117	114	88	90	103	0.224821	989/1000	NonOverlappingTemplate
104	106	111	102	102	97	106	101	90	81	0.649612	992/1000	NonOverlappingTemplate
93	110	109	90	116	100	100	88	92	102	0.536163	992/1000	NonOverlappingTemplate
95	103	117	86	89	84	101	102	109	114	0.225998	988/1000	NonOverlappingTemplate
113	93	103	125	90	107	102	77	86	104	0.041981	980/1000	NonOverlappingTemplate
76	98	112	109	100	78	105	111	104	107	0.090936	997/1000	NonOverlappingTemplate
97	94	113	90	92	80	93	124	103	114	0.06523	995/1000	NonOverlappingTemplate

86	103	131	96	106	103	102	101	86	86	0.062036	997/1000	NonOverlappingTemplate
93	85	93	104	92	122	103	99	110	99	0.352107	985/1000	NonOverlappingTemplate
98	82	94	106	104	114	104	90	92	116	0.313041	991/1000	NonOverlappingTemplate
116	102	86	93	122	94	107	100	87	93	0.16626	991/1000	NonOverlappingTemplate
89	118	108	91	96	88	101	107	99	103	0.524101	985/1000	NonOverlappingTemplate
93	109	91	107	102	97	101	105	97	98	0.959347	987/1000	NonOverlappingTemplate
96	104	100	98	99	91	100	109	105	98	0.986227	990/1000	NonOverlappingTemplate
114	106	98	95	101	86	98	113	102	87	0.53012	988/1000	NonOverlappingTemplate
95	104	103	100	99	90	110	89	117	93	0.626709	988/1000	NonOverlappingTemplate
93	102	96	85	107	106	105	101	98	107	0.869278	991/1000	NonOverlappingTemplate
101	99	116	95	95	92	117	102	103	80	0.293952	991/1000	NonOverlappingTemplate
98	98	113	111	94	83	97	104	111	91	0.484646	989/1000	NonOverlappingTemplate
103	101	88	94	105	99	110	102	99	99	0.95493	990/1000	NonOverlappingTemplate
107	104	100	97	82	102	87	105	106	110	0.603841	992/1000	NonOverlappingTemplate
103	96	99	101	88	99	113	96	102	103	0.930026	992/1000	NonOverlappingTemplate
97	90	106	88	94	105	108	116	111	85	0.337688	991/1000	NonOverlappingTemplate
127	91	101	107	110	88	96	82	89	109	0.057875	986/1000	NonOverlappingTemplate
108	115	102	96	113	62	107	94	103	100	0.01695	990/1000	NonOverlappingTemplate
87	106	109	87	93	106	104	103	113	92	0.536163	989/1000	NonOverlappingTemplate
95	107	103	105	110	89	91	89	103	108	0.735908	994/1000	NonOverlappingTemplate
105	95	101	85	91	101	109	108	105	100	0.809249	985/1000	NonOverlappingTemplate
120	103	105	83	113	97	101	97	95	86	0.254411	986/1000	NonOverlappingTemplate
107	107	92	88	99	87	103	92	111	114	0.469232	988/1000	NonOverlappingTemplate
103	98	101	102	102	96	93	107	85	113	0.807412	992/1000	NonOverlappingTemplate
114	97	106	104	103	78	98	111	102	87	0.313041	991/1000	NonOverlappingTemplate
85	88	99	97	103	109	104	109	108	98	0.705466	987/1000	NonOverlappingTemplate
93	103	111	101	92	104	102	98	116	80	0.415422	994/1000	NonOverlappingTemplate
98	91	89	97	102	116	98	100	104	105	0.816537	992/1000	NonOverlappingTemplate

92	111	90	105	99	104	91	112	89	107	0.614226	995/1000	NonOverlappingTemplate
93	94	83	108	102	100	105	97	119	99	0.496351	991/1000	NonOverlappingTemplate
112	100	104	109	93	90	93	100	105	94	0.834308	989/1000	NonOverlappingTemplate
115	106	95	93	111	84	98	101	109	88	0.399442	990/1000	NonOverlappingTemplate
103	82	107	97	98	96	110	102	95	110	0.699313	991/1000	NonOverlappingTemplate
116	84	97	110	100	88	91	110	110	94	0.288249	982/1000	NonOverlappingTemplate
95	97	100	115	91	96	96	107	111	92	0.733899	991/1000	NonOverlappingTemplate
110	82	97	105	119	96	90	120	88	93	0.083526	991/1000	NonOverlappingTemplate
115	109	80	117	91	100	100	92	93	103	0.214439	988/1000	NonOverlappingTemplate
105	109	85	94	71	99	122	93	119	103	0.012128	990/1000	NonOverlappingTemplate
102	99	97	107	86	104	115	104	97	89	0.693142	991/1000	NonOverlappingTemplate
83	109	123	81	112	111	111	75	90	105	0.004365	990/1000	NonOverlappingTemplate
94	85	107	104	92	119	111	93	100	95	0.39594	992/1000	NonOverlappingTemplate
100	100	105	90	96	101	100	105	110	93	0.957612	994/1000	NonOverlappingTemplate
104	107	85	114	110	100	93	87	95	105	0.480771	989/1000	NonOverlappingTemplate
97	103	110	106	106	86	86	118	101	87	0.292519	991/1000	NonOverlappingTemplate
84	82	111	103	116	92	101	101	110	100	0.254411	992/1000	NonOverlappingTemplate
103	95	96	100	97	103	102	105	95	104	0.997943	993/1000	NonOverlappingTemplate
96	98	82	91	110	112	98	100	102	111	0.536163	991/1000	NonOverlappingTemplate
96	105	115	89	100	103	97	101	101	93	0.870856	995/1000	NonOverlappingTemplate
93	93	107	106	97	85	102	117	99	101	0.624627	990/1000	NonOverlappingTemplate
86	108	94	95	109	87	92	114	102	113	0.347257	996/1000	NonOverlappingTemplate
110	91	98	97	118	100	104	95	97	90	0.670396	993/1000	NonOverlappingTemplate
106	99	87	109	106	90	111	102	86	104	0.574903	990/1000	NonOverlappingTemplate
93	105	101	93	100	91	103	93	104	117	0.771469	987/1000	NonOverlappingTemplate
115	98	90	94	93	107	94	110	102	97	0.727851	990/1000	NonOverlappingTemplate
100	85	94	108	102	96	97	103	100	115	0.751866	992/1000	NonOverlappingTemplate
100	103	78	111	109	99	112	96	93	99	0.431754	987/1000	NonOverlappingTemplate

115	100	79	85	95	119	92	115	99	101	0.073872	992/1000	NonOverlappingTemplate
89	99	90	95	107	97	124	105	90	104	0.33297	993/1000	NonOverlappingTemplate
83	113	101	106	109	91	107	92	97	101	0.55442	997/1000	NonOverlappingTemplate
102	93	93	104	104	93	99	110	94	108	0.921624	992/1000	NonOverlappingTemplate
112	108	101	96	100	105	96	92	92	98	0.912724	992/1000	NonOverlappingTemplate
100	87	115	125	106	101	94	93	96	83	0.106877	992/1000	NonOverlappingTemplate
99	99	85	91	109	102	95	106	105	109	0.779188	988/1000	NonOverlappingTemplate
107	103	119	84	94	86	106	87	104	110	0.197981	986/1000	NonOverlappingTemplate
114	98	87	99	98	90	89	100	128	97	0.126658	985/1000	NonOverlappingTemplate
95	90	107	113	102	105	97	92	104	95	0.846338	987/1000	NonOverlappingTemplate
99	113	85	88	98	107	107	116	89	98	0.33297	990/1000	NonOverlappingTemplate
103	101	109	102	97	111	101	74	106	96	0.38899	985/1000	NonOverlappingTemplate
103	102	106	97	96	93	99	101	103	100	0.998169	984/1000	NonOverlappingTemplate
87	118	83	97	92	92	109	112	110	100	0.189625	987/1000	NonOverlappingTemplate
110	107	100	96	91	93	101	108	106	88	0.798139	989/1000	NonOverlappingTemplate
125	90	104	96	96	89	91	113	102	94	0.22248	983/1000	NonOverlappingTemplate
104	94	116	99	106	109	92	103	91	86	0.558502	988/1000	NonOverlappingTemplate
97	94	113	90	92	80	93	124	104	113	0.06943	995/1000	NonOverlappingTemplate
117	108	96	120	117	79	74	90	108	91	0.003996	985/1000	OverlappingTemplate
121	94	99	92	110	93	95	102	103	91	0.524101	985/1000	Universal
92	109	91	109	87	108	106	101	88	109	0.532132	988/1000	ApproximateEntropy
53	56	59	64	58	67	57	72	62	65	0.839987	607/613	RandomExcursions
60	72	48	51	71	59	78	55	64	55	0.116054	608/613	RandomExcursions
83	64	51	58	58	54	64	58	58	65	0.241456	603/613	RandomExcursions
63	65	56	54	55	61	56	74	67	62	0.748093	601/613	RandomExcursions
72	71	57	53	70	48	60	60	52	70	0.22948	608/613	RandomExcursions
61	68	63	63	65	56	59	53	59	66	0.955982	609/613	RandomExcursions
68	48	56	77	59	66	75	57	46	61	0.070744	604/613	RandomExcursions

51	55	66	70	61	69	72	47	55	67	0.249701	607/613	RandomExcursions
61	57	71	59	53	61	55	61	70	65	0.816833	608/613	RandomExcursionsVariant
67	50	57	69	60	56	65	58	60	71	0.701669	607/613	RandomExcursionsVariant
66	52	59	55	52	80	68	47	67	67	0.095539	605/613	RandomExcursionsVariant
64	61	54	62	48	51	54	79	57	83	0.019935	606/613	RandomExcursionsVariant
62	55	62	55	43	56	70	75	57	78	0.057593	606/613	RandomExcursionsVariant
61	52	52	62	71	56	66	75	62	56	0.458724	602/613	RandomExcursionsVariant
59	52	59	45	72	71	67	66	65	57	0.289435	602/613	RandomExcursionsVariant
57	50	54	67	65	64	67	47	77	65	0.196314	608/613	RandomExcursionsVariant
63	47	46	61	79	57	60	66	61	73	0.082315	606/613	RandomExcursionsVariant
62	63	50	53	70	68	64	75	55	53	0.330947	607/613	RandomExcursionsVariant
71	65	57	67	54	64	54	61	60	60	0.872348	605/613	RandomExcursionsVariant
66	60	62	83	59	61	52	54	59	57	0.282511	605/613	RandomExcursionsVariant
62	68	72	57	55	71	51	60	56	61	0.599625	603/613	RandomExcursionsVariant
63	74	61	58	59	76	62	54	53	53	0.379067	603/613	RandomExcursionsVariant
68	65	71	67	59	56	63	64	53	47	0.509568	604/613	RandomExcursionsVariant
67	69	64	63	71	54	60	60	52	53	0.667811	605/613	RandomExcursionsVariant
64	67	72	58	55	67	57	60	47	66	0.53911	607/613	RandomExcursionsVariant
68	57	69	59	54	68	61	55	62	60	0.88243	607/613	RandomExcursionsVariant
102	93	103	107	102	88	98	93	100	114	0.827279	994/1000	Serial
110	94	93	101	114	104	80	84	106	114	0.169044	990/1000	Serial
85	92	111	104	103	95	104	97	107	102	0.800005	994/1000	LinearComplexity

- -

ANECDOTAL INFORMATION

Bruce Schneier is widely regarded as one of the greatest minds in cryptography and certainly deserves his own paragraph. I have crossed paths with him in the past – another story. I find it humorous and a bit coincidental that while working to commercialize this, in a part of my sales pitch I would jokingly say that "even if the aliens come down, they can't crack our encryption". My financial advisor, who apparently failed to see the humor, strongly advised me to stop saying that, even though everyone who heard it laughed and took it tongue-in-cheek. Shortly after I stopped using that jovial verbiage, Bruce Schneier released a blog essay ("Cryptography after the Aliens Land", inline below) that stated anyone who would attempt an OTP was a crackpot. I am including this as an example of the mindset prevalent amongst cryptographers with respect to OTP. This mindset has been the result of the general thinking that large encryption keys must be continuously distributed to all relevant parties in order to perform this type of encryption.

"CRYPTOGRAPHY AFTER THE ALIENS LAND"

- **Bruce Schneier, *IEEE Security & Privacy,* September/October 2018.**

  Quantum computing is a new way of computing—one that could allow humankind to perform computations that are simply impossible using today's computing technologies. It allows for very fast searching, something that would break some of the encryption algorithms we use today. And it allows us to easily factor large numbers, something that would break the RSA cryptosystem for any key length.

  This is why cryptographers are hard at work designing and analyzing "quantum-resistant" public-key algorithms. Currently, quantum computing is too nascent for

cryptographers to be sure of what is secure and what isn't. But even assuming aliens have developed the technology to its full potential, quantum computing doesn't spell the end of the world for cryptography. Symmetric cryptography is easy to make quantum-resistant, and we're working on quantum-resistant public-key algorithms. If public-key cryptography ends up being a temporary anomaly based on our mathematical knowledge and computational ability, we'll still survive. And if some inconceivable alien technology can break all of cryptography, we still can have secrecy based on information theory—albeit with significant loss of capability.

At its core, cryptography relies on the mathematical quirk that some things are easier to do than to undo. Just as it's easier to smash a plate than to glue all the pieces back together, it's much easier to multiply two prime numbers together to obtain one large number than it is to factor that large number back into two prime numbers. Asymmetries of this kind—one-way functions and trap-door one-way functions—underlie all of cryptography.

To encrypt a message, we combine it with a key to form ciphertext. Without the key, reversing the process is more difficult. Not just a little more difficult, but astronomically more difficult. Modern encryption algorithms are so fast that they can secure your entire hard drive without any noticeable slowdown, but that encryption can't be broken before the heat death of the universe.

With symmetric cryptography—the kind used to encrypt messages, files, and drives— that imbalance is exponential, and is amplified as the keys get larger. Adding one bit of key increases the complexity of encryption by less than a percent (I'm hand-

waving here) but doubles the cost to break. So a 256-bit key might seem only twice as complex as a 128-bit key, but (with our current knowledge of mathematics) it's 340,282,366,920,938,463, 463,374,607,431,768,211,456 times harder to break.

Public-key encryption (used primarily for key exchange) and digital signatures are more complicated. Because they rely on hard mathematical problems like factoring, there are more potential tricks to reverse them. So you'll see key lengths of 2,048 bits for RSA, and 384 bits for algorithms based on elliptic curves. Here again, though, the costs to reverse the algorithms with these key lengths are beyond the current reach of humankind.

This one-wayness is based on our mathematical knowledge. When you hear about a cryptographer "breaking" an algorithm, what happened is that they've found a new trick that makes reversing easier. Cryptographers discover new tricks all the time, which is why we tend to use key lengths that are longer than strictly necessary. This is true for both symmetric and public-key algorithms; we're trying to future-proof them.

Quantum computers promise to upend a lot of this. Because of the way they work, they excel at the sorts of computations necessary to reverse these one-way functions. For symmetric cryptography, this isn't too bad. Grover's algorithm shows that a quantum computer speeds up these attacks to effectively halve the key length. This would mean that a 256-bit key is as strong against a quantum computer as a 128-bit key is against a conventional computer; both are secure for the foreseeable future.

For public-key cryptography, the results are more dire. Shor's algorithm can easily break all of the commonly used public-key algorithms based on both factoring and

the discrete logarithm problem. Doubling the key length increases the difficulty to break by a factor of eight. That's not enough of a sustainable edge.

There are a lot of caveats to those two paragraphs, the biggest of which is that quantum computers capable of doing anything like this don't currently exist, and no one knows when—or even if— we'll be able to build one. We also don't know what sorts of practical difficulties will arise when we try to implement Grover's or Shor's algorithms for anything but toy key sizes. (Error correction on a quantum computer could easily be an unsurmountable problem.) On the other hand, we don't know what other techniques will be discovered once people start working with actual quantum computers. My bet is that we will overcome the engineering challenges, and that there will be many advances and new techniques—but they're going to take time to discover and invent. Just as it took decades for us to get supercomputers in our pockets, it will take decades to work through all the engineering problems necessary to build large-enough quantum computers.

In the short term, cryptographers are putting considerable effort into designing and analyzing quantum-resistant algorithms, and those are likely to remain secure for decades. This is a necessarily slow process, as both good cryptanalysis transitioning standards take time. Luckily, we have time. Practical quantum computing seems to always remain "ten years in the future," which means no one has any idea.

After that, though, there is always the possibility that those algorithms will fall to aliens with better quantum techniques. I am less worried about symmetric cryptography, where Grover's algorithm is basically an upper limit on quantum improvements, than I am about public-key algorithms based on number theory, which feel more fragile. It's

possible that quantum computers will someday break all of them, even those that today are quantum resistant.

If that happens, we will face a world without strong public-key cryptography. That would be a huge blow to security and would break a lot of stuff we currently do, but we could adapt. In the 1980s, Kerberos was an all-symmetric authentication and encryption system. More recently, the GSM cellular standard does both authentication and key distribution—at scale—with only symmetric cryptography. Yes, those systems have centralized points of trust and failure, but it's possible to design other systems that use both secret splitting and secret sharing to minimize that risk. (Imagine that a pair of communicants get a piece of their session key from each of five different key servers.) The ubiquity of communications also makes things easier today. We can use out-of-band protocols where, for example, your phone helps you create a key for your computer. We can use in-person registration for added security, maybe at the store where you buy your smartphone or initialize your Internet service. Advances in hardware may also help to secure keys in this world. I'm not trying to design anything here, only to point out that there are many design possibilities. We know that cryptography is all about trust, and we have a lot more techniques to manage trust than we did in the early years of the Internet. Some important properties like forward secrecy will be blunted and far more complex, but as long as symmetric cryptography still works, we'll still have security.

It's a weird future. Maybe the whole idea of number theory—based encryption, which is what our modern public-key systems are, is a temporary detour based on our incomplete model of computing. Now that our model has expanded to include quantum computing, we might end up back to where we were in the late 1970s and early 1980s:

symmetric cryptography, code-based cryptography, Merkle hash signatures. That would be both amusing and ironic.

Yes, I know that quantum key distribution is a potential replacement for public-key cryptography. But come on—does anyone expect a system that requires specialized communications hardware and cables to be useful for anything but niche applications? The future is mobile, always-on, embedded computing devices. Any security for those will necessarily be software only.

There's one more future scenario to consider, one that doesn't require a quantum computer. While there are several mathematical theories that underpin the one-wayness we use in cryptography, proving the validity of those theories is in fact one of the great open problems in computer science. Just as it is possible for a smart cryptographer to find a new trick that makes it easier to break a particular algorithm, we might imagine aliens with sufficient mathematical theory to break all encryption algorithms. To us, today, this is ridiculous. Public- key cryptography is all number theory, and potentially vulnerable to more mathematically inclined aliens. Symmetric cryptography is so much nonlinear muddle, so easy to make more complex, and so easy to increase key length, that this future is unimaginable. Consider an AES variant with a 512-bit block and key size, and 128 rounds. Unless mathematics is fundamentally different than our current understanding, that'll be secure until computers are made of something other than matter and occupy something other than space.

But if the unimaginable happens, that would leave us with cryptography based solely on information theory: one-time pads and their variants. This would be a huge

blow to security. One-time pads might be theoretically secure, but in practical terms they are unusable for anything other than specialized niche applications. Today, only crackpots try to build general-use systems based on one-time pads—and cryptographers laugh at them, because they replace algorithm design problems (easy) with key management and physical security problems (much, much harder). In our alien-ridden science-fiction future, we might have nothing else.

Against these godlike aliens, cryptography will be the only technology we can be sure of. Our nukes might refuse to detonate and our fighter jets might fall out of the sky, but we will still be able to communicate securely using one-time pads. There's an optimism in that.

I have a feeling that I am Bruce's crackpot, and he does perfectly parody the tongue-in-cheek humor from my marketing presentations (which was already a parody, specifically of the movie "Independence Day", 1996), but that's okay. Bruce is brilliant, and I'm not offended. To detail my very limited experience with Bruce, in the late 1990's when I was a network architect and technology security director for Deutsche Bank, Bruce was a respected cryptographer at banking seminars. His pitch at the time was essentially "you can have absolute security if you use my encryption". A few years later when I was running a security startup based on the heuristic firewall and Bruce was running a new secure datacenter company (Counterpoint), his pitch was essentially "you can't have absolute security, but if you use my datacenters you'll be as secure as you can be". We spoke at the time about the possibility of blending technologies, but he was not interested. That being said, he wrote patents a couple of years later in which he forward cited my patent (U.S. Patent 6,519,703 – "Methods and Apparatus for Heuristic Firewall"), so there was perhaps some interest in the tech after all. I anticipate that this

encryption technology will be initially received by Schneier in similar fashion with hopes that perhaps one day he will lift the “crackpot” moniker from deep within my psyche;-)

CONCLUSION

As the first globally scalable OTP solution, HOP represents the strongest and fastest encryption available. It is compact enough to run on wireless sensors and Internet of Things (IoT) devices, robust enough to handle the needs of phones, laptops, and the largest cluster computers, and the fastest and strongest option for Internet Exchange Points. It brings an end to the encryption race-condition and provides a permanent solution to the encryption needs of any and all devices for the foreseeable future. While this dissertation is specific to the encryption world, it should be intuitive that HOP's random number generation primitives are not limited to encryption; rather, this technique most definitely applies to any situation in which random numbers are useful. Additionally intuitive, as HOP is fully authenticated, are the implications of using this technique with processes and inter process communications, and the positive effects this will yield in platform, operating system, application, and data security overall. The dynamic identification primitives have the potential to change the way we look at, and in fact do, authentication. As a former member of the NIST Biometrics Consortium, I clearly see that the integration of these techniques with biometric authentication is a no-brainer. Penultimately, though this is by no means an exhaustive list of applicability, by applying HOP at the core of a distributed ledger, it will be possibly to not only significantly speed up a blockchain (due to greatly reduced consensus operations requirements), but also to certify that the blockchain is secure enough to perform currently elusive tasks such as Title Transfer in real estate, and the creation of unassailable smart contract environments. Finally, it is not a stretch to say that HOP can make positive change in all aspects of our handling and processing of data. To quote the Beatles (1970): "Let It Be".

REFERENCES AND SUPPLEMENTAL RESOURCES

- Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 4055.
- Advanced Encryption Standard (AES), FIPS Publication 197, National Institute of Standards and Technology (NIST).
- Al-Kadi, Ibraham A., The Origins of Cryptology: The Arab Contributions, *Cryptologia*, vol. 16, no. 2 (April 1992).
- Anderson, Ross, *Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd Edition* (Somerset, NJ: Wiley, 2008).
- Ash, Avner, Gross, Robert, *Fearless Symmetry: Exposing the Hidden Pattern of Numbers* (Princeton: Princeton University Press, 2006).
- Bamford, James, *Body of Secrets: Anatomy of the Ultra-Secret National Security Agency* (New York: Random House, 2001).
- Bellovin, Steven M., Frank Miller: Inventor of the One-Time Pad, *Cryptologia*, vol. 35 (2011), pg. 203–222.
- Bernstein, Dennis S., *Matrix Mathematics: Theory, Facts, and Formulas - Second Edition* (Princeton: Princeton University Press, 2009).
- Bernstein, Dennis S., Scalar, Vector, and Matrix Mathematics: Theory, Facts, and Formulas - Revised and Expanded Edition (Princeton: Princeton University Press, 2018).
- Beutelspacher, Albrecht, *Cryptology* (Washington, D.C.: Mathematical Association of America, 1994).
- Blunden, Bill, *The Rootkit Arsenal* (Plano, TX: Wordware Publishing, 2009).

- Broder, Christian G., *Practische Grammatik der Lateinischen Sprache*, (Leipzig: F.C.W. Bogel, 1815).
- The Camellia Cipher in OpenPGP, RFC 5581.
- Chen, Lily, NIST Crypto Standard Approaches – Past, Present, and Future, NIST Presentation, https://csrc.nist.gov/CSRC/media/Presentations/The-NIST-Standardization-Approach-on-Cryptography/images-media/chen-lily-threshold-crypto-workshop-March-2019.pdf.
- Chow, Jerry and Osbourne, Michael, "ENCRYPTION INCEPTION: The solution to quantum computers cracking cryptography is already here", *Quartz*, 2 May 2019.
- Coron, Jean-Sebastien; Dodis, Yevgeniy; Malinaud, Cecile; & Puniya, Prashant, Merkle- Damgård revisited : how to construct a hash function", September 4, 2007, https://cs.nyu.edu/~dodis/ps/merkle.pdf.
- Cracker Tools, https://blackarch.org/cracker.html.
- CrackStation, https://crackstation.net.
- *Crypto StackExchange*, community crypto blog, https://crypto.stackexchange.com.
- Cryptographic Algorithm Implementation Requirements and Usage Guidance for Encapsulating Security Payload (ESP) and Authentication Header (AH), RFC 8221.
- Cryptographic Message Syntax (CMS), RFC 3369.
- Cryptographic Message Syntax (CMS) Algorithms, RFC 3370.
- Cryptographic Protection of TCP Streams (tcpcrypt), RFC 8548.
- *Cryptology ePrint Archive*, community crypto research site, https://eprint.iacr.org.
- Data Encryption Standard, FIPS Pub. 46-1 (Washington, D.C.: National Bureau of Standards, 1987).

- Davis, Michael; Bodmer, Sean; & LeMasters, Aaron, *Hacking Exposed™ Malware & Rootkits Security Secrets & Solutions* (New York: McGraw-Hill, 2010).
- A Description of the Camellia Encryption Algorithm, RFC 3713.
- Diffie, Whitfield & Hellman, Martin, New Directions in Cryptography, *IEEE Transactions on Information Theory*, vol. 22 (1976), pg 644-654.
- Diffie, Whitfield, and Landau, Susan, *Privacy on the Line* (Cambridge, MA: MIT Press, 1998).
- Diffie-Hellman Key Agreement Method, RFC 2631.
- Digital Signature Standard, FIPS Publication 186-3, National Institute of Standards and Technology (NIST), June 2009.
- Encryption Cracking Tools, http://books.gigatux.nl/mirror/wireless/0321202171/ch06lev1sec1.html.
- ESET, corporate website, 2019, https://www.eset.com/us/business/endpoint-security/encryption/.
- Franksen, Ole Immanuel, *Mr. Babbage's Secret* (London: Prentice-Hall, 1985).
- Gardner, Martin, *Codes, Ciphers, and Secret Writing*, (Mineola, NY: Dover, 1984).
- GOST R 34.12-2015: Block Cipher "Kuznyechik," RFC 7801.
- Green, Matthew, A few thoughts on cryptographic engineering, blog, https://blog.cryptographyengineering.com.
- Guide to Industrial Control Systems (ICS) Security: Supervisory Control and Data Acquisition (SCADA) Systems, Distributed Control Systems (DCS), and Other Control System Configurations such as Programmable Logic Controllers (PLC), Special

Publication 800-82 revision 2, National Institute of Standards and Technology (NIST), May 2015.

- Guide to SSL VPNs, Special Publication 800-113, National Institute of Standards and Technology (NIST), July 2008.
- Guide to Storage Encryption Technologies for End User Devices, Special Publication 800-111, National Institute of Standards and Technology (NIST), November 2007.
- Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithm, RFC 7696.
- Guidelines for Cryptographic Key Management, RFC 4107.
- Guideline for Using Cryptographic Standards in the Federal Government: Directives, Mandates and Policies, Special Publication 800-175A, National Institute of Standards and Technology (NIST), August 2016.
- Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms, Special Publication 800-175B, National Institute of Standards and Technology (NIST), August 2016.
- Hellman, Martin E., An Extension of the Shannon Theory Approach to Cryptography, *IEEE Transactions on Information Theory*, vol. 23 (1977), pg. 289-294.
- Hellman, Martin E., The Mathematics of Public Key Cryptography, *Scientific American*, vol. 241 (August 1977).
- Hioureas, Vasilios, Encryption 101: How to break encryption, Malwarebytes Labs blog, https://blog.malwarebytes.com/threat-analysis/2018/03/encryption-101-how-to-break-encryption/.

- ID Quantique, corporate website, 2019, https://www.idquantique.com/quantum-safe-security/overview/qkd-technology/.
- Internet X.509 Public Key Infrastructure: Additional Algorithms and Identifiers for DSA and ECDSA, RFC 5758.
- Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280.
- Ironclad Encryption, corporate website, 2019, https://www.ironcladencryption.com.
- Joyce, James B., *Methods and Apparatus for Heuristic Firewall*, U.S. Patent 6,519,703, 2000.
- Joyce, James B., *Methods and Apparatus for Heuristic/Deterministic Finite Automata*, U.S. Patent Application , 2006.
- Kahn, David, *The Codebreakers* (New York: Scribner, 1996).
- Kahn, David, *Seizing the Enigma* (London: Arrow, 1996).
- Korn, Granino A. & Korn, Theresa M., *Mathematical Handbook for Scientists and Engineers* (Mineola, NY: Dover, 2000).
- Kugler, Otto, One time pad encryption, Mils Electronic, Austria, 2019, https://www.cryptomuseum.com/manuf/mils/files/mils_otp_proof.pdf.
- Kurose, James F. & Ross, Keith W., *Computer Networking – A Top-Down Approach*, (New York: Addison-Wesley, 5th Edition, 2010).
- The Legion of the Bouncy Castle, http://www.bouncycastle.org.
- Leon-Garcia, Alberto & Widjaja, Indra, *Communication Networks: Fundamental Concepts and Key Architectures* (New York: McGraw-Hill, 2nd Edition 2004).

- Luenberger, David G., *Information Science* (Princeton: Princeton University Press, 2006).
- Mayhan, Robert J., *Discrete-Time and Continuous-Time Linear Systems* (Reading, MA: Addison-Wesley, 1984).
- The MD5 Message-Digest Algorithm, Request for Comment (RFC) 1321.
- Menezes, Alfred J.; van Oorschot, Paul C.; & Vanstone, Scott A., *Handbook of Applied Cryptography* (Waterloo, Canada: CRC Press, 1996).
- Mermin, David N., *Quantum Computer Science: An Introduction* (Cambridge, UK: Cambridge University Press, 2007).
- Naor, Moni & Yung, Moti, Universal One-Way Hash Functions and their Cryptographic Applications, 21st ACM Symposium on Theory of Computing, revised May 1995, http://www.wisdom.weizmann.ac.il/~naor/PAPERS/uowhf.pdf.
- Newton, David E., *Encyclopedia of Cryptology* (Santa Barbara: ABC-Clio, 1997).
- OpenPGP Message Format, RFC 4880.
- Post-Quantum, corporate website, 2019, UK, https://www.post-quantum.com.
- The PPP Triple-DES Encryption Protocol (3DESE), RFC 2420.
- Pressman, Roger S., *Software Engineering: A Practitioner's Approach* (New York: McGraw-Hill, 7th Edition 2010).
- Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1, RFC 3447.
- Quintessence Labs, corporate website, 2019, https://www.quintessencelabs.com.
- Recommendation for Key Management - Part 1 (Revised), Special Publication 800-57, National Institute of Standards and Technology (NIST), March 2007.

- Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography (Revised), Special Publication 800-56A, National Institute of Standards and Technology (NIST), March 2007.
- Recommendation for Random Bit Generator (RBG) Constructions: 2nd Draft, Special Publication 800-90C, National Institute of Standards and Technology (NIST), April 2016.
- Report on Post-Quantum Cryptography, Interagency/Internal Report (NISTIR) 8105, National Institute of Standards and Technology, 28 April 2016.
- Rijmenants, Dirk, CIPHER MACHINES AND CRYPTOLOGY: One-time Pad, 2004-2019, http://users.telenet.be/d.rijmenants/en/onetimepad.htm
- Rivest, Ronald L.; A. Shamir; & L. Adleman, A Method for Obtaining Digital Signatures and Public Key Cryptosystems, *Communications of the ACM 21*, (1978), pg. 120-126.
- RSA, corporate website, https://www.rsa.com, 2019.
- RSA Code-Breaking Contest Again Won by Distributed.Net and Electronic Frontier Foundation, RSA, 1999, http://www.rsasecurity.com/press_release.asp?doc_id=462&id=1034.
- Schneier, Bruce, *Applied Cryptography: Protocols, Algorithms, and Source Code in C* (New York: John Wiley & Sons, 1996).
- Schneier, Bruce, Schneier on Security, 2004-2019, http://www.schneier.com.
- Secure Hash Standard, FIPS Publication 180-3, National Institute of Standards and Technology (NIST), October 2008.

- Security and Privacy Controls for Federal Information Systems and Organizations, Special Publication 800-53 revision 4, National Institute of Standards and Technology (NIST), April 2013.
- Security Implications of Using the Data Encryption Standard (DES), RFC 4772.
- SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, FIPS Publication 202, National Institute of Standards and Technology (NIST).
- Shannon, Claude E., *The Mathematical Theory of Communication* (Urbana: University of Illinois Press, 1949).
- Shannon, Claude E., Communication Theory of Secrecy Systems, *Bell System Technical Journal 28*, (1949), pg. 656-715.
- Singh, Simon, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography* (New York: Random House, 1999).
- A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Special Publication 800-22, National Institute of Standards and Technology (NIST), April 2010.
- Symantec, corporate website, US, https://www.symantec.com/products/encryption, 2019.
- Tiwari, Harshvardhan, Merkle-Damgård Construction Method and Alternatives: A Review, JIOS, vol.. 41, no. 2, 2017.
- Transitioning the Use of Cryptographic Algorithms and Key Lengths, Special Publication 800-131A Rev2, National Institute of Standards and Technology (NIST), March 2019.
- The Translations and KGB Cryptographic Systems, in *The Venona Story,* National Security Agency, Fort Meade, Maryland, 15 Jan. 2004, pp. 26–27.
- The Transport Layer Security (TLS) Protocol Version 1.2, RFC 5246.

- Tripathi, A.N., *Linear Systems Analysis* (New Dehli: New Age International Publishers, 2nd edition 1998).
- UNSW Newsroom, "200 times faster than ever before: the speediest quantum operation yet" University of New South Wales, 18 July, 2019, https://newsroom.unsw.edu.au/news/science-tech/200-times-faster-ever-speediest-quantum-operation-yet.
- US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF), RFC 6234.
- Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS), RFC 3565.
- Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS), RFC 5753.
- Using SHA2 Algorithms with Cryptographic Message Syntax, RFC 5754.
- Whitewood Security, corporate website, 2019, http://whitewoodsecurity.com.
- Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Standard (ECDSA), X9.62-2005, November, 2005.
- Yoshigoe, Kenji & Al, Murat, Adaptive Confidentiality Mechanism for Hierarchical Wireless Sensor Networks, *2008 IEEE Globecom Workshops*, 2008.
- Zimmerman, Philip R., *The Official PGP User's Guide* (Cambridge, MA: MIT Press, 1996).

ADDITIONAL SUPPLEMENTAL READING

- Revisiting the OTP, http://ijns.jalaxy.com.tw/contents/ijns-v6-n1/ijns-2008-v6-n1-p94-102.pdf.
- Crypto Overview, Perfect Sec OTP, https://www.ics.uci.edu/~stasio/fall04/lect1.pdf.
- Shannon Impossibility, proofs etc., https://cs.nyu.edu/~dodis/ps/one-time-pad.pdf.
- Cryptography chapter 20, https://introtcs.org/public/lec_19_cryptography.pdf.
- Disproof, https://techcrunch.com/2015/08/16/a-disproof-of-the-one-time-pad/.
- OTP – unifying Shannon material, https://arxiv.org/pdf/0803.0046.pdf.
- Essential Shannon, https://eprint.iacr.org/2000/059.pdf.
- Perfect Ciphers, https://secgroup.dais.unive.it/teaching/cryptography/shannon-theory-on-perfect-ciphers/.
- OTP – Protecting Everything, https://www.linkedin.com/pulse/protecting-everything-one-time-pad-encryption-internet-julian-murguia.
- OTP – Port Hop, http://introspectivenetworks.com/history-of-the-one-time-pad/.
- Patent for Port Hopping OTP, http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=1&f=G&l=50&co1=AND&d=PTXT&s1=%22streaming+one+time+pad%22.TI.&OS=TTL/.
- Bell Labs – Claude Shannon, https://www.bell-labs.com/claude-shannon/.
- Perfect Adversarial Indistinguishability, http://www.cs.miami.edu/home/burt/learning/Csc609.162/perfect-secrecy-notes.pdf.
- Burton Rosenberg CV, http://www.cs.miami.edu/home/burt/papers/cv.html.
- Shannon's paper, http://netlab.cs.ucla.edu/wiki/files/shannon1949.pdf.

- Pigeon OTP story, http://www.circleid.com/posts/20160504_writing_the_next_chapter_for_the_historic_one_time_pad/.
- Frank Miller, https://www.nytimes.com/2011/07/26/science/26code.html.