Analysis of Randomness of GAEN keys

April Sheeran, MCS

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science

Supervisor: Stephen Farrell

April 2024

Analysis of Randomness of GAEN keys

April Sheeran, Master of Science in Computer Science
University of Dublin, Trinity College, 2024

Supervisor: Stephen Farrell

...ABSTRACT...

Acknowledgments

Thank you Mum & Dad.

April Sheeran

University of Dublin, Trinity College

April 2024

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Here are summaries of the papers I have read thus far

1.1 Background for GAEN keys

Background

The first paper is Contact Tracing App Privacy: What Data Is Shared By Europe's GAEN Contact Tracing Apps (Leith and Farrell (2021)). This paper discusses the data sent to back end servers of various contact tracing apps in Europe. The apps have two components: the 'client' app which is controlled by the national public health authority and the GAEN service that, on android, is managed by Google as part of the Google Play services. The paper investigates both of these components. It found that the Google Play services component continuously sends requests to the Google servers which contain information such as the phone IMEI, handset hardware serial number, SIM serial number, etc. This may be considered intrusive , however most users would have accepted this data sharing if that had previously enabled Google Play services. This data sharing is not specific to the covid tracing app. There may be a way to identify a user is specific data is sent in every request. There are experiments run to test the data being sent by different apps.

The paper CoAvoid: Secure, Privacy-Preserved Tracing of Contacts for Infectious Diseases (Li et al. (2022)) discusses the privacy and security concerns of covid tracing apps and proposes its own CoAvoid app which improves on these issues. It talks about how inappropriate uses of the GAEN API may expose all information about confirmed patients to servers and relevant users. This may allow someone to gather lots of information about patients and discover their identities, daily routines or social relationships. Due to a limitation of BLE, it may be possible for attackers to send users excess false alarms and put further strain on the public health system (wormhole attack). It talk about how the BLE to calculate if two devices came into contact can be affected by factors such as environment, transmitting power, receiving sensitivity, etc. The GAEN design is vulnerable to profiling, possibly de-anonymising infected persons and wormhole attacks. It describes how the GAEN app works: It randomly generates a Daily Tracking Key (DTK), a unique identifier used for 24 hours. It uses function f to derive the Rolling Proximity Identifier (RPI) based on the DTK. These identifiers are sent in Bluetooth Advertisements, which will be replaced every 20 minutes. The apps simultaneously collects and stores other users RPIs locally. If a user test positive, their DTK is uploaded to a central server. Other download these DTKs and reconstruct the RPIs. They compare these to their local list of keys. The authenticity of the information being downloaded cannot be verified and thus wormhole attacks may occur. The paper further describes two types of potential attacks: Wormhole and Privacy Analysis attack. It may be possible to identify things about a user by tracking the DTK uploads.

The paper Digital Contact Tracing Solutions: Promises, Pitfalls and Challenges (Nguyen et al. (2022)) analyses digital contact tracing solutions. It discusses the security and privacy issues with GAEN. It proposes its own solution called TraceCorona. Apple and Google collaborated to create a decentralised contact tracing interface calle Exposure Notification API (GAEN). Access to the API has been given to only one organisation authorised by the government. BLE is used for sensing the proximity between individual devices. The phones send out information like temporary identifiers (TempIDs) that can be sensed by other devices. It also records signal strength in an attempt to estimate the distance of the encounter. It discusses requirements of accuracy, superspreader and accountability for a digital contact tracing system to be acceptable.

The article Privacy and Integrity Threats in Contact Tracing Systems and Their Mitigations (Avitabile et al. (2023)) reports privacay and integrity threats in GAEN and proposes a new system called Pronto-B2. Threats to security and privacy include the possibility of tracing and deanonymising citizens using passive devices and injecting false at-risk alerts. An attacker may trace a user by linking locations visited by the same user. They may also try to deanonymise users by linking locations visited by users to their real identities. 'Paparazzi Attack': using passive devices, the attacker can catch and store the pseudonyms of a target user. They can link together the passive devices recieved the pseudonyms belonging to the same user. The attacker can obtain information about the habits of an infected user and use it for economical gain. 'Brutus Attack': The attacker colludes with the server and the health authority to discover which user uploaded certain data. Pseudonyms in GAEN are called rolling proximity identifiers (RPIs). A short random secret called Temporary Exposure key (TEK) is generated each day by the smartphone. All RPIs of a given day of a user are generated by running a PRF on the input TEK.

The paper October 2020 Survey of GAEN App Key Uploads (Farrell (2020)) is a survey of the TEKs published across 8 European regions. It estimates the number of users uploading TEKs and compares this to the expected number based on population, number of active users and covid case counts. It reports a shortfall of uploads in a number of regions. The efficacy of these apps remains unclear.

1.2 Background for how GAEN keys are Generated

Here is how the keys are generated.

The documentation from Google and Apple (Apple (2020)) found online details how

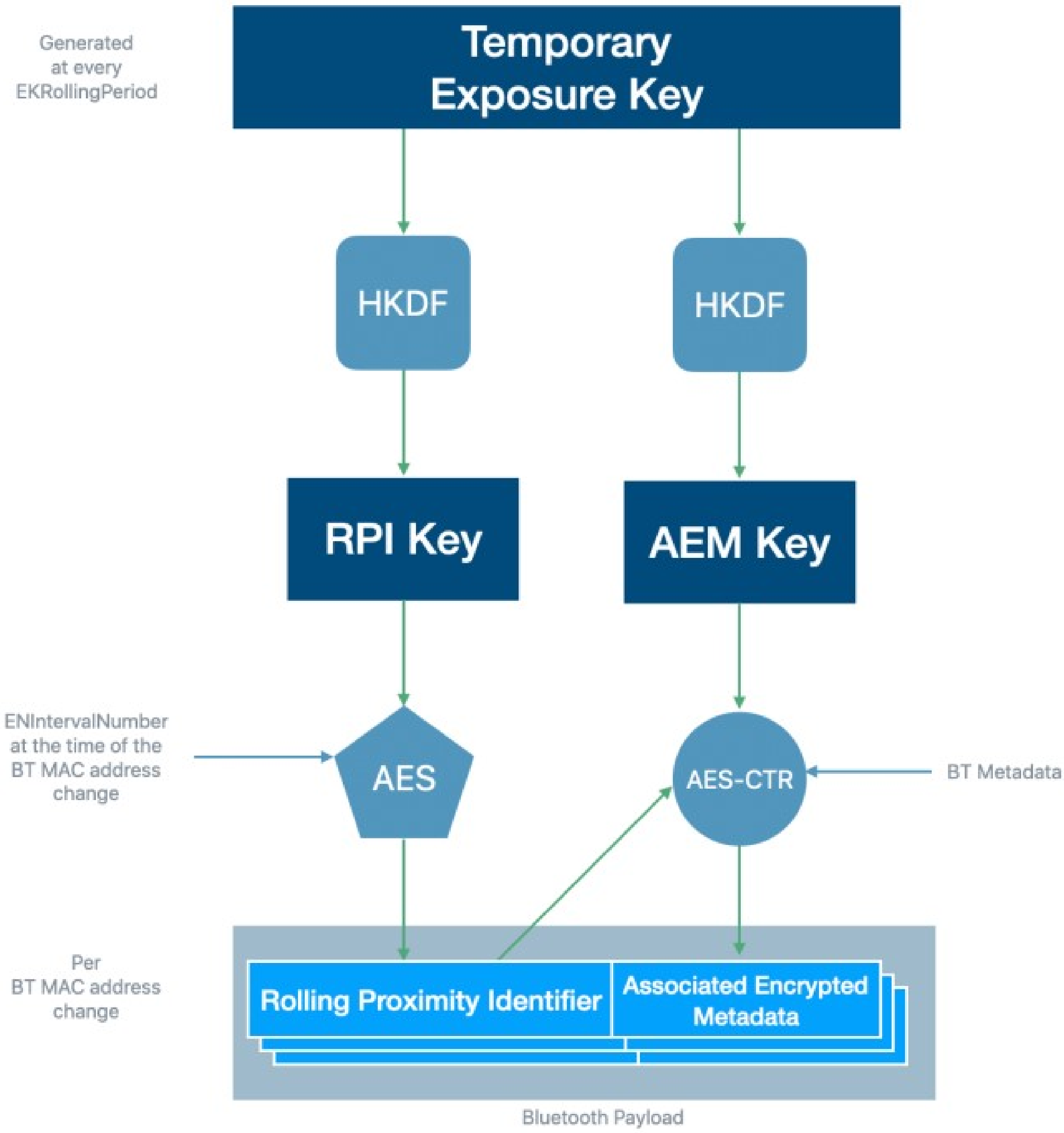

Figure 1.1: How the GAEN key is generated

the GAEN keys are encrypted. An ENIntervalNumber is calculated every 10 minutes, it is a 32-bit unsigned little endian value. The TEKRollingPeriod is how long the Temporary Exposure Key is valid, it is 24 hours. The Temporary Exposure Key is generated and associated with an ENIntervalNumber. It is generated as a 16-byte value using CRNG, which is a cryptographic number generator. At the end of every TEKRollingPeriod, a new key is generated. The Rolling Proximity Identifier Key (RPIK) is derived from the TEK. It is derived by RPIKi ← HKDF(teki, NUL L, UTF8("EN-RPIK"),16), where the HKDF function is as detailed in (Krawczyk and Eronen (2010)), which uses the SHA-256 hash functions. The Rolling Proximity Identifiers are privacy preserving identifiers that are broadcast in Bluetooth payloads. Each time the MAC address changes a new Rolling Proximity Identifier is generated using the Rolling Proximity Identifier Key using RPIi, j ← AES128(RPIKi, PaddedDataj). The use of 16-byte identifiers result in a low probability of collisions and limits the risk of false positive matches. The associated encrypted metadata is encrypted with the Rolling Proximity Identifier.

1.3 Background for Tests for Randomness

The paper Randomness testing of modern encryption techniques in cloud environment (Mohamed et al. (2012)) compares different modern encryption techniques on a traditional desktop and in a cloud environment. It tests 8 algorithms, including AES which is used in GAEN. It uses the NIST statistical test package which contains 15 tests, listed in the paper. It implements the tests in Java. It runs the NIST tests which produces P-values, which are rejected if they are less than 0.01. There is a max rejection rate, 4 in this paper, there is an equation to calculate this. It runs these tests on all the algorithms and outputs its results. It doesn't find any problems with the algorithms but there are differences when they are ran on the desktop environment versus in the cloud.

The paper Effectiveness Analysis of Encrypted and Unencrypted Bit Sequence Identification Based on Randomness Test (Wu et al. (2015)) uses statistical tests to identify whether a bit sequence is encrypted or unencrypted. Randomness tests are used to evaluate the security of cipher algorithms. It uses the SP800-22 rev1a standard. The standard contains 15 randomness tests but this paper uses 5 of them, frequency test, frequency test within a block, runs test, longest runs of one in a block and cumulative sums test. It provides analysis on some of these tests, frequency, runs, frequency within a block. It describes the P-value used and how it is calculated. It somewhat accurately classifies a bit sequence as encrypted or unencrypted.

The paper Analyzing of Chaos based Encryption with Lorenz and Henon Map (Anandkumar and Kalpana (2018))

The paper Statistical Analysis of Enhanced SDEx Encryption Method Based on SHA-512 Hash Function (Hłobaż (2020)) performs statistical analysis on an enchanced SDEx method based on the SHA-512 hash function. It uses NIST and carries out four tests: frequency, block frequency, cumulative sum and runs. The algorithm passes all the test, it gets a P-value of ¿ 0.01. It could not do the compression tests from NIST as the size of the test files was too small so it uses WINRAR. It says that of the file is a string of random bits then it should be able to be compressed. It checks that the size of the file before and after compression using WINRAR is the same. The algorithm passes this test also. References a paper about the cryptanalysis of the SDEx method, should read as this paper only does the statistical analysis.

The paper Analysis of the Randomness Performance of the Proposed Stream Cipher Based Cryptographic Algorithm (Brosas et al. (2020)) does randomness testing on an adapted version of the Vernam Cipher. It uses NIST to do cumulative sums, runs, longest run of ones and frequency tests. It mentions other test suits called NIST STS, Dieharder and TestU01. Mentions Strict Avalanche Criterion (SAC) testing but i do not think this is relevant to the GAEN keys. The cumulative sum is a test concentrates on determining the maximal excursion from zero of the random walk described by the increasing amount of adjusted (-1, +1) digits in the sequence. The Runs Test was utilized to verify whether the total numbers of runs of ones and zeros of various lengths areas projected for a random sequence. The longest run was to define whether the longest run of ones within an M-bit block of the tested sequence is consistent with the length of the longest run of ones anticipated in a random sequence of M bits. For a pseudorandom number generator, the number of ones and zeros in the output must be equal. A test that determines the proportions of zeros and ones for the entire sequence is the Frequency test.

The paper On the Robustness of RSA-OAEP Encryption and RSA-PSS Signatures Against (Malicious) Randomness Failures (Schuldt and Shinagawa (2017)) analyses the robustness of the RSA-OAEP encryption scheme and the RSA-PSS signature scheme. The paper gives examples of real world randomness failures but it focused on the enc scheme and signature thing rather than the randmoness of the key itself. It's a very technical paper that I do not fully understand and do not think its discussion of randmoness tests is relevant to the GAEN kesy testing. It is very theoretical. Mentions random oracle model and repeated randmoness and collision resistant?

The paper Evolving boolean functions for fast and efficient randomness testing (Mrazek et al. (2018)) is VERY USEFUL paper. Introduces a new boolean function for testing randomness, s a novel method for the statistical randomness testing of cryptographic primitives, which is based on the evolutionary construction of the so-called randomness distinguisher. Each distinguisher is represented as a Boolean polynomial in the Algebraic Normal Form. It talks about evolutionary algorithms and their new Boolean functions operating as simple, but high-quality distinguishers of statistical (non)randomness in the data generated by cryptographic algorithms. These boolean functions can provide the same results as the usual test suites NIST etc, but faster and using less data. The quality of each distinguisher was measured in terms of the so-called Z-score. Mentions another paper using a software package named Ent. In total, they evaluated seven statistics (entropy, compression, hisquared, arithmetic mean, pierror, excess and correlation) associated to five tests, resulting in an observation that studied statistics are not completely independent and could be reduced to five statistics. The paper goes on to describe their function but it is quite confusing and shows their results. Would definitely be worth looking at to test the GAEN keys.

The paper Recommendations on Statistical Randomness Test Batteries for Cryptographic Purposes (Luengo and Villalba (2021)) is a VERY USEFUL paper. It compares the different batteries(test) suits like NIST, Dieharder, TestU01 and more. It lists the tests used in each one and explains them. It contains very useful references. Need to exam the keys and their structure and refer to this paper to begin deciding on a test suite and what test to run. Contains useful tables of pros and cons of each and compares the tests they have.

To ensure the strength and resilience of your encryption solutions (Design (2024)) against various attacks and threats, you need to measure and compare them using methods and tools such as simulations, audits, reviews, and assessments. For example, you might want to conduct security tests on key management to prevent key leakage, loss, or compromise; encryption mode to avoid weaknesses such as padding oracle attacks; cryptanalysis to detect or prevent brute force or side-channel attacks; and compliance with laws, regulations, standards, and best practices. By doing so, you can evaluate the performance and security of your encryption solutions and enhance their quality and effectiveness.

Article by OWASP (OWASP (2024b)) might be useful.

Article by OWASP (OWASP (2024a)) about Insecure Randomness. Insecure randomness errors occur when a function that can produce predictable values is used as a source of randomness in security-sensitive context. Computers are deterministic machines, and as such are unable to produce true randomness. Pseudo-Random Number Generators (PRNGs) approximate randomness algorithmically, starting with a seed from which subsequent values are calculated.

Wikipedia page (Wikipedia (2023)) on Randomness test. Mentions famous PRNG that fails randomness test https://en.wikipedia.org/wiki/RANDU . Though there are commonly used statistical testing techniques such as NIST standards, Yongge Wang showed that NIST standards are not sufficient. Furthermore, Yongge Wang(http://webpages.uncc.edu/yonwa designed statistical-distance-based and law-of-the-iterated-logarithm-based testing techniques. Using this technique, Yongge Wang and Tony Nicol("Statistical Properties of Pseudo Random Sequences and Experiments with PHP and Debian OpenSSL") detected the weakness in commonly used pseudorandom generators such as the well known Debian version of OpenSSL pseudorandom generator which was fixed in 2008. The use of Hadamard transform (https://en.wikipedia.org/wiki/Hadamard-transform) to measure randomness was proposed by S. Kak and developed further by Phillips, Yuen, Hopkins, Beth and Dai, Mund, and Marsaglia and Zaman, On the Complexity of Pseudo-Random Sequences or If You Can Describe a Sequence It Can't be Random. Advances in Cryptology.

Paper (Turan et al. (2008))

1.4 Approaches for Testing for Randomness

Below is a list of potential test approachs for my dissertation, and tests I should consider

- NIST, Dieharder, TestU01 (Luengo and Villalba (2021)) INCLUDE at least 1
- chi-squared test, Entropy measurement INLCUDED above
- Boolean functions (Mrazek et al. (2018)) INCLUDE
- Hilbert curve and heatmap, mentioned in tek transparency reports. Analyse distrubutions, INCLUDE
- OpenSSL, Crypto++, PyCrypto, and JCE potentially (Design (2024)) POSSIBLY include if relevent
- bias testing: Keys generated at specific times POSSIBLY include if I have time
- Hadamard spectral test (Wikipedia (2023)) Used for linear congruential generators, not applicable

1.5 Why was the level of significance was chosen

Choosing the appropriate level of significance involves balancing the risk of false positives against false negatives. A value of a=0.05 implies that the null hypothesis is rejected 5 per cent of the time when it is in fact true. The choice of a is somewhat arbitrary, although in practice values of 0.1, 0.05, and 0.01 are common (NIST (2012)). False positives may lead to us rejecting keys that are actually random and false negatives could mean accepting keys that are non random as random. Larger datasets may allow us to use lower significance levels as more data allows us to detect non randomness better.

Currently using alpa = 0.5. NIST (Bassham et al. (2010)) says . Common values of a in cryptography are about 0.01.

"Because there are so many tests for judging whether a sequence is random or not, no specific finite set of tests is deemed "complete." In addition, the results of statistical testing must be interpreted with some care and caution to avoid incorrect conclusions about a specific generator" https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

Chapter 2

State of the Art

Introduction to chapter

2.1 Background

2.1.1 What are GAEN keys

GAEN Apps

Google and Apple developed the Google/Apple Exposure Notification (GAEN) system to facilitate contact tracing in response to the Covid-19 Pandemic. (Mention conventional contact tracing). Nations across the world used this technology to create contract tracing apps, for example Covid Tracker in Ireland and SwissCovid in Switzerland (Leith and Farrell (2021)).

The way the contact tracing works, if a user enables it, is as follows:

- Every 10-20 minutes the user's device will generate a random 128-bit key, referred to as a Temporary Exposure Key (TEK).
- The user's device will broadcast these keys using Bluetooth Low Energy (BLE).
- The user's device will listen and store the TEKs being broadcasted from other devices within a certain radius. These TEKs are stored locally on the device.

- If a user tests positive for Covid, they can log this into the app. The app will send the user's recent TEKs (around the 14 days) to a central server managed by the local health authority.
- Every approx. 2 hours, the user's device will download the TEKs from the central server.
- The app compares these downloaded TEKs to the TEKs stored locally on the device.
- If there is a match, this means that the user has potentially been exposed to Covid and the app will notify them.
- CITES

How Keys are Generated

Not sure if i will include this or not yet.

Analysis on Apps

There have been numerous studies done on contact tracing apps that use GAEN technology. Google and Apple acknowledge that keeping users' information private and secure is essential to the success of the contact tracing app and claim to have designed their system with this central to the design (Google (2020)). Why is privacy so important for this app? Don't want to share covid status etc.

The major privacy concerns of GAEN apps according to (Nguyen et al. (2022)) are the identification of users, tracking users or extracting the social graph of users. Contact tracing should aim to identify encounters rather than actual users, by doing so they should not leak any information that could be used to identify the user. Similarly, the data collected should not be able to be used to create the social graph of the user, the social connections and relationships of a user. Having this information could potentially result in the user being identified.

Security is also a requirement for a contact tracing app according to (Nguyen et al. (2022)). The system should be resilient to large scale data pollution attacks. These could be fake exposure claims, where users may falsely claim they have been exposed in order to get out of work or another obligation or in an attempt to damage the reputation and credibility of the contact tracing apps. Fake exposure injection, a relay attack, may send users false notifications of potential exposures. The attacker could do this by capturing the TEKs of some user and broadcasting them in another location, leading to people being falsely notified of exposure. This could result in panic among users and the population, putting further strain on the healthcare system by creating a demand for unnecessary tests. It could also damage the trust in the contact tracing apps as their accuracy would be no longer trusted.

(Nguyen et al. (2022)) assess the GAEN apps on a variety of requirements. In terms of effectiveness, GAEN has been found to be imprecise at determining the distances between user devices (do i need to reference an internal reference?). Its use of BLE means scanning of the user's surroundings for other devices can only happen with frequent pauses to save battery life of the device. Many factors like positioning of the device's antenna, obstacles in the way and orientation of the device affect the computation of the distance between devices and the errors are significant. GAEN fails to account for 'superspreaders' of the virus, an individual who is very contagious and infects a number of other people. GAEN also does not have any mechanism for dealing with asymptomatic individuals, people that are infected with the virus and are contagious but do not show symptoms. Unknowingly, these people spread the virus. These individuals are unlikely to get tested and therefore won't log their infection in the app, meaning those that come in contact with them will not be notified of a potential exposure. This significantly impacts the effectiveness of the contact tracing apps.

An investigation into the data shared by Europe's contact tracing apps that use GAEN (Leith and Farrell (2021)) discovered that a significant amount of data was being sent to Google servers. The android implementations of the GAEN systems use Google Play

Services to facilitate GAEN-based contact tracing. The user must enable Google Play Services. It was found that Google Play Services connects to Google servers approximately every 20 minutes, sending requests that include the handset IP address, location data and persistent identifiers to link requests coming from the same device. The data sent to Google in other types of requests also include phone IMEI, device hardware serial number, SIM serial number and IMSI, phone number, WiFi MAC address, user email and Android ID. While sharing data to backend servers is not in itself an intrusion of privacy, the ability to link this data to a real-world user is problematic. Given that the user's IP address is being sent to Google very frequently, this could be used as location tracking. It is possible to de-anonymise this location data and potentially identify the user. Given that the user must enable Google Play Services, and therefore this data sharing, to do contact tracing, this does raise a concern to the privacy of the user.

2.1.2 What is Randomness

In order to test for randomness/non-randomness we must first define what randomness is. A random bit sequence could be explained as the result of flipping an unbiased coin, with two sides 1 and 0, which has an equal chance of 50 per cent of landing on side 1 or side 0. Each flip of the coin does not affect any future coin flips which means the flips are independent of each other. This unbiased coin can therefore be considered a perfect random bit stream generator as the appearances of 1s and 0s will be randomly and uniformly distributed. All elements in the sequence are independent of each other and future elements in the sequence cannot be predicted using previous elements. CITE NIST Special Publication 800-107 Revision 1, Recommendation for Applications Using Approved Hash Algorithms. This simple example gives us an understanding of what it means for a set of keys to be random. The keys must exhibit certain properties in order to be accepted as random: they should be independent and equally likely. (?).

2.2 Literature Review

2.2.1 How to Test for randomness/ Analyses done

2.2.2 Examples of Randomness Failures

Chapter 3

Design

3.1 Chosen Tests

3.1.1 Test Suites

The NIST and Dieharder test suites were chosen to evaluate the GAEN keys.

(May not be used because it wants streams of random numbers as input) NIST is widely used in industry (Hłobaż (2020)) (Brosas et al. (2020)) (Mohamed et al. (2012)) and is accepted as a standard. It contains tests that are recommended by NIST for the evaluation of PRNGs used in cryptography.

Dieharder is another widely used battery of tests for randomness created by Robert G. Brown. It is an extension of the Diehard suite of tests created by George Marsgalia.

3.1.2 Other Tests

Some tests outside of the test suites were performed. These include chi-squared test, lagplot and hilbert curve. The counts of the numbers of 1s and 0s in each bit position were also recorded and this data was plotted to allow for quick inspection.

This is a non-exhaustive list of possible tests. There are endless tests that can be ran to build confidence that the keys are random, however it is never certain if the data is

Dieharder Test	**Descriptions**
Birthdays Test	Item 1
OPERM5 Test	Item 1
32x32 Binary Rank Test	Item 1
6x8 Binary Rank Test	Item 1
Bitstream Test	Item 1
OPSO	Item 1
DNA Test	Item 1
Count the 1s stream Test	Item 1
Count the 1s byte Test	Item 1
Parking Lot Test	Item 1
Minimum Distance 2d circle Test	Item 1
Minimum Distance 3d sphere Test	Item 1
Squeeze Test	Item 1
Runs Test	Item 1
Craps Test	Item 1
Tang and Marsgalia GCD Test	Item 1
STS Monobit Test	Item 1
STS runs Test	Item 1
STS serial Test	Item 1
RGB Bit Distribution Test	Item 1
RGB Generalised Minimum Distance Test	Item 1
RGB Permutations Test	Item 1
RGB Lagged Sum Test	Item 1
RGB Kolmogorov-Smirnov Test	Item 1
Byte Distribution	Item 1
DAB DCT	Item 1
DAB Fill Tree Test	Item 1
DAB Fill Tree 2 Test	Item 1
DAB Monobit 2 Test	Item 1

Table 3.1: Tests in Dieharder test suite

truly random or not. The tests done here detect deviations from randomness rather than prove randomness.

Lagplot "A lag plot checks whether a data set or time series is random or not. Random data should not exhibit any identifiable structure in the lag plot. Non-random structure in the lag plot indicates that the underlying data are not random" (NIST (2010))

Hilbert Curve why ??

Spectral Test (Discrete FFT) Note that this description is taken from the NIST documentation http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf . The focus of this test is the peak heights in the Discrete Fourier Transform of the sequence. 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. The intention is to detect whether the number of peaks exceeding the 95per cent threshold is significantly different than 5per cent.

Chi-squared A chi-square test checks how many items you observed in a bin vs how many you expected to have in that bin. It does so by summing the squared deviations between observed and expected across all bins (expand)

Plot of Counts Counts number of 0s and 1s in each bit position. Should be 50-50 1s and 0s. A plot of the data quickly shows any potential biases.

3.2 Closely-Related Work

3.2.1 Aspect #1

3.2.2 Aspect #2

3.3 Summary

Summarize the chapter and present a comparison of the projects that you reviewed.

	Aspect #1	**Aspect #2**
Row 1	Item 1	Item 2
Row 2	Item 1	Item 2
Row 3	Item 1	Item 2
Row 4	Item 1	Item 2

Table 3.2: Caption that explains the table to the reader

Bibliography

Anandkumar, R. and Kalpana, R. (2018). Analyzing of chaos based encryption with lorenz and henon map. In 2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2018 2nd International Conference on, pages 204–208.

Apple, G. (2020). Exposure notification, cryptography specification.

Avitabile, G., Botta, V., Iovino, V., and Visconti, I. (2023). Privacy and integrity threats in contact tracing systems and their mitigations. IEEE Internet Computing, 27(2):13–19.

Bassham, L. E., Rukhin, A. L., Soto, J., Nechvatal, J. R., Smid, M. E., Barker, E. B., Leigh, S. D., Levenson, M., Vangel, M., Banks, D. L., Heckert, N. A., Dray, J. F., and Vo, S. (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications:.

Brosas, D. G., Sison, A. M., Hernandez, A. A., and Medina, R. P. (2020). Analysis of the randomness performance of the proposed stream cipher based cryptographic algorithm. In 2020 11th IEEE Control and System Graduate Research Colloquium (ICSGRC), pages 76–81.

Design, T. (2024). How do you test and evaluate the performance and security of your encryption solutions?

Farrell, S. (2020). October 2020 survey of gaen app key uploads.

Google, A. (2020).

Hłobaż, A. (2020). Statistical analysis of enhanced sdex encryption method based on sha-512 hash function. In 2020 29th International Conference on Computer Communications and Networks (ICCCN), pages 1–6.

Krawczyk, D. H. and Eronen, P. (2010). HMAC-based Extract-and-Expand Key Derivation Function (HKDF). RFC 5869.

Leith, D. J. and Farrell, S. (2021). Contact tracing app privacy: What data is shared by europe's gaen contact tracing apps. In IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, pages 1–10.

Li, T., Yin, S., Yu, R., Feng, Y., Jiao, L., Shen, Y., and Ma, J. (2022). Coavoid: Secure, privacy-preserved tracing of contacts for infectious diseases. IEEE Journal on Selected Areas in Communications, 40(11):3191–3206.

Luengo, E. A. and Villalba, L. J. G. (2021). Recommendations on statistical randomness test batteries for cryptographic purposes. ACM Comput. Surv., 54(4).

Mohamed, E. M., El-Etriby, S., and Abdul-kader, H. S. (2012). Randomness testing of modern encryption techniques in cloud environment. In 2012 8th International Conference on Informatics and Systems (INFOS), pages CC–1–CC–6.

Mrazek, V., Sys, M., Vasicek, Z., Sekanina, L., and Matyas, V. (2018). Evolving boolean functions for fast and efficient randomness testing. In Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, page 1302–1309, New York, NY, USA. Association for Computing Machinery.

Nguyen, T. D., Miettinen, M., Dmitrienko, A., Sadeghi, A.-R., and Visconti, I. (2022). Digital contact tracing solutions: Promises, pitfalls and challenges. IEEE Transactions on Emerging Topics in Computing, pages 1–12.

NIST (2010). Lagplot.

NIST (2012). Critical values and p values.

OWASP (2024a). Insecure randomness.

OWASP (2024b). Testing for weak encryption.

Schuldt, J. C. and Shinagawa, K. (2017). On the robustness of rsa-oaep encryption and rsa-pss signatures against (malicious) randomness failures. In Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS '17, page 241–252, New York, NY, USA. Association for Computing Machinery.

Turan, M., Doğanaksoy, A., and Boztas, S. (2008). On independence and sensitivity of statistical randomness tests. volume 5203, pages 18–29.

Wikipedia (2023). Randomness test.

Wu, Y., Wang, T., and Li, J. (2015). Effectiveness analysis of encrypted and unencrypted bit sequence identification based on randomness test. In 2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC), pages 1588–1591.