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

State of the Art

Give introduction

1.1 Background

Introduction

1.1.1 What are GAEN Keys

Introduction

Contact Tracing 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

Studies on GAEN-based 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.

(Avitabile et al. (2023)) examines several potential threats and privacy concerns of GAEN apps. They introduce a 'paparazzi attack' which involves using passive Bluetooth devices to capture the keys being broadcasted from a targeted user. If this user tests positive, the attacker can match their locally stored keys to those made publically available on the central server and learn that the user is positive for covid. This is a form of deanonymization. Similarly, an attacker could exploit the movements of a targeted user to gain money by linking the locations of the passive devices to the keys and sell this data to interested parties.

The effectiveness of these GAEN contact tracing apps is undetermined (Leith and Farrell (2021)) (Nguyen et al. (2022))

Numerous alternative to the GAEN system have been proposed such as TraceCorona by (Nguyen et al. (2022)) in an attempt to remedy the above privacy and security concerns.

1.1.2 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 percent 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 (Dang (2012)). 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 meaning no previously generated keys affect a new key. Equally likely meaning that the probability of a 0 or 1 appearing at any point in the key is equal to 1/2. Scalable meaning that if the key is random, then any extracted subsequence is also random. (Cortez et al. (2020))Any indication of a dependency or bias within the data would indicate nonrandomness.

1.2 Literature Review

Introduction

1.2.1 How to Test for Randomness

It is important to note that you can not say for certain whether something is random or not, you can only find evidence against non-randomness. It is not possible to give theoretical proof of randomness of a sequence. (Turan et al. (2008)) Various statistical tests can be performed on the data in an attempt to compare and evaluate the data against a truly random sequence since the outcome when a statistical test is applied to a truly random sequence is known. (Bassham et al. (2010)).

A challenge when testing for randomness is that there is no agreed upon complete set of statistical tests to deem a sequence random (Bassham et al. (2010)). There is an infinite number of tests that you could run in order to find the presence or absence of a pattern or bias within the data. The existence of a pattern or bias within the data would indicate that it is non randomness.

Hypothesis Testing

Statistical testing is used to test against a defined null hypothesis (h0). The null hypothesis in this case is that the keys being tested are random. The alternative hypothesis (h1) is that the keys are not random. The challenge here is to determine which of these hypotheses can be accepted (Luengo and Villalba (2021)) . For each statistical test run on the data, the result accepts or rejects the null hypothesis.

The following table shows the possible results on a hypothesis test: ¡table¿ (Bassham et al. (2010))

The above situations are somewhat unknown but some control can be gained by knowing the probability of each of the error situations. The probability of Error Type 1 is defined as a, the level of significance (Luengo and Villalba (2021)). This value is typically 0.01, 0.05 or 0.10. The probability of Error Type 2 is defined as B, referred to as contrast power and is usually used as 1-B. (Luengo and Villalba (2021)). If the data is truly random, rejecting the null hypothesis, determining that the data is non-random, will occur a small percentage of the time. For example if a is 0.01, it would be expected that 1 sequence in 100 sequences is rejected (Bassham et al. (2010)).

In practice, p-values are used to reject or accept the null hypothesis. In the context of this project, a p-value can be defined as the probability that a key produced is less random than the keys previously tested, given the kind of non-randomness the test is assessing (Bassham et al. (2010)). NEED TO FIX. A p-value equal to 1 indicates that the data is perfectly random while a p-value equal to 0 indicates that the data is completely non random. If the p-value is greater than or equal to a, the null hypothesis is accepted and the data appears to be random. If the p-value is less than a, the null hypothesis is rejected and the data is deemed non random.

Maybe K-S stuff here or later on TALK ABOUT test suites and what they are, not specific to one

1.2.2 Studies on Randomness Testing

Intro something like many examples of randomness testing being used on cryptographic techniques/applications. Multiple applications

Given that encryption is essential for maintaining data security in cloud computing, (Mohamed et al. (2012)) performed randomness testing on eight modern encryption tech-

niques, including AES, MARS and DES. They tested on two different platforms, desktop computer and Amazon EC2 Micro Instance. They evaluated the encryption techniques implemented as Pseudo Random Number Generators (PRNGs). They used the NIST Test Suite to perform the randomness testing. With a significance level of 0.01, any p-value less than 0.01 meant that sequence was rejected. They found no strong evidence of any statistical non randomness across the 8 encryption algorithms however some differences were found between them on the two different platforms.

Statistical analysis has been run on an enhanced SDEx encryption method based on the SHA-512 hash function (Hłobaż (2020)). Using various tests like frequency, cumulative sums and runs, with a significance level of 0.01, it was found that this encryption algorithm was sufficiently random and passed the tests. They concluded that this SDEx method based on the SHA-512 hash function was quicker and equally or more secure than AES with a 256-bit key. They hope to use this method to secure end-to-end encryption for data transfer.

Similarly, statistical tests for randomness have been run on new algorithms, like a proposed stream cipher cryptographic algorithm based on the popular Vernam Cipher (Brosas et al. (2020)). The algorithm had a success rate of 99.5 percent across the statistical tests performed on it, which included frequency and longest runs of one's tests. Due to this success, the paper deemed the proposed algorithm effective in producing a random ciphertext sequence and detailed further work of implementing it to help secure medical records.

SHA256 is vulnerable to length extension attacks which involve misusing particular hashes as authentication codes and using them to include extra information. (**?**) introduces a new and improved padding scheme and hashing process for SHA256 to deal with this issue. To verify that the solution is cryptographically secure, statistical tests for

randomness are performed on the output of the Message Digest. Tests such as monobit frequency, frequency within a block and runs were carried out on the data. The results validate that the number of ones and zeros are randomly distributed in the final hash value.

Statistical tests were also used to identify encrypted and unencrypted bit sequences (Wu et al. (2015)). Unencrypted bit sequences are less random than encrypted ones. From the SP800-22 rev1a standard, five tests were selected and a significance level of 0.01 was chosen. If the sequence passes more than 3 of the tests, it was concluded that that sequence was encrypted. Otherwise the sequence was concluded as unencrypted. The results of the experiment were that 89 percent of the time, unencrypted sequences were identified correctly and 99 percent of the time encrypted sequences were identified correctly.

1.2.3 Examples of Randomness Failures

Randomness failures pose a serious threat to cryptographic security (Schuldt and Shinagawa (2017)). The consequences can be severe and there are many examples of real-world incidents.

There are many examples of pseudo random number generators (PRNGs) failing and being guessable. Notably, the Debian Linux vulnerability in 2008 that left cryptographic keys to be guessable. It was caused by the code used to gather entropy, used to seed the PRNG used to create private keys, were removed. This resulted in only 32,768 possible keys meaning the connections made with these keys were insecure. CITE 11111

In 2015, Juniper Networks announced that there were multiple security vulnerabilities due to unauthorised code in their operating system, for their NetScreen VPN routers, called ScreenOS. CITE 22222 These vulnerabilities were due to Juniper's use of Dual

EC (Elliptical Curve) as a PRNG. Dual EC had a weakness that was exploited in the Juniper incident. It was possible for an attacker, who knew the discrete logarithm of an input parameter Q with respect to a generator point, to see a number of consecutive bytes from the output and hence calculate the internal state of the generator. This allowed the attacker to predict all the future output of the generator. They were able to exploit this lack of randomness and passively decrypt VPN traffic.

Bitcoin thefts in 2013 were due to a compromised PRNG used in Android wallets CITE 9999. Applications on Android using Java Cryptography Architecture (JCA) for key generation, signing and generating random numbers were not receiving cryptographically strong values because of an improper initialization of the underlying PRNG SecureRandom on Android devices CITE 991. The predictability of the values being generated by SecureRandom was exploited and attackers were able to guess the private keys used in Bitcoin Wallets and steal the Bitcoins the wallet contained. Again, attackers were able to exploit the lack of randomness.

1.3 Summary

Chapter 2

Design

2.1 Challenges

2.1.1 Review of Test Suites

2.1.2 Interpretation of Results

2.2 Methodology

2.2.1 Data Preparation

The data used in this project was sourced from the Testing Apps for COVID-19 Tracing (TACT) project by Farrell and Leith CITE 2. The TACT project was a study on whether the BLE used in GAEN-based contact tracing applications was effective at identifying users who were in proximity for long enough to be deemed as exposed to covid, if one of the users was later positive for the virus. The project ran from April 2020 until September 2023 and a number of reports were written on the findings CITE some examples.

While the project was ongoing, the TEKs being published in 33 regions, including Ireland, Germany and Brazil, were downloaded hourly. This resulted in a huge amount

of data and allowed for insight into the functioning of these apps. The TEKs downloaded were in a large number of zip files.

In order to get the keys in the correct format to test, I first extracted keys from zipped files. Following this, duplicate keys were identified and removed, resulting in a file composed of only the unique keys. This significantly reduced the data size, from an initial 56GB of all the keys in the zip files, down to 4GB, removing a substantial amount of duplicate keys. The final dataset consists of a total of 129 million unique TEKs in ascii-hex format??, allowing for efficient analysis of the keys.

2.2.2 Chosen Test Suite

2.2.3 Other Tests

2.2.4 Random Dataset

Bibliography

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.

Cortez, D. M. A., Sison, A. M., and Medina, R. P. (2020). Cryptographic randomness test of the modified hashing function of sha256 to address length extension attack. In Proceedings of the 2020 8th International Conference on Communications and Broadband Networking, ICCBN '20, page 24–28, New York, NY, USA. Association for Computing Machinery.

Dang, Q. (2012). Recommendations for applications using approved hash algorithms.

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.

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.

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.

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.

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.

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.