Collisions of uniformly distributed identifiers with an application to MAC address anonymization

Jean-François Determe*, Sophia Azzagnuni*, Utkarsh Singh*, François Horlin*, and Philippe De Doncker**

September 22, 2020

Disclaimer — This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.

Abstract

The main contribution of this paper consists in theoretical approximations of the collision rate of $n$ random identifiers uniformly distributed in $m (> n)$ buckets—along with bounds on the approximation errors. A secondary contribution is a decentralized anonymization system of media access control (MAC) addresses with a low collision rate. The main contribution supports the secondary one in that it quantifies its collision rate, thereby allowing designers to minimize $m$ while attaining specific collision rates. Recent works in crowd monitoring based on WiFi probe requests, for which collected MAC addresses should be anonymized, have inspired this research.

I Introduction

Widely used structures in computer science associate inputs with outputs that are approximately uniformly distributed in the set of all possible outputs. Hash tables [1, Chap. 11], cryptographic hash functions and token generators (used for anonymization or security purposes) are examples of such structures [2, Sec. 9.7.1]. An issue is that these structures may generate collisions, that is, two different inputs being mapped onto the same output [3, Sec. 9].

The main contribution of this paper is a set of numerically stable estimates of the collision rate—the average number of collisions divided by the number of inputs—in a hash-based or token-based system; our estimates assume that the hash function or token generator yields uniformly distributed outputs. We also derive bounds on the error of these estimates.

The main contribution supports our secondary one, a decentralized anonymization procedure for media access control (MAC) addresses. Specifically, our main contribution quantifies the collision rate of our secondary one, thereby making it possible to tune the parameters of the anonymization procedure so as to attain arbitrarily small collision rates (at the cost of higher bandwidth and storage requirements).

We present our secondary contribution through the lens of a crowd monitoring system based on WiFi signals. Anonymizing MAC addresses from WiFi probe requests (PRs) [4, Fig. 4-52] is indeed required in networks of sensors used for crowd monitoring [5,7]. [3, Sec. 7]. Our scheme prevents user tracking and time synchronization accuracy is no issue on modern networks.

* All authors are with the OPERA Wireless Communications Group, Université libre de Bruxelles, 1050 Brussels, Belgium. Corresponding e-mail: Jean-Francois.Determe@ulb.be. Innoviris funded Jean-François Determe and Utkarsh Singh.

** All authors are with the OPERA Wireless Communications Group, Université libre de Bruxelles, 1050 Brussels, Belgium. Corresponding e-mail: Jean-Francois.Determe@ulb.be. Innoviris funded Jean-François Determe and Utkarsh Singh.
The main contribution is general and could be of interest to researchers and engineers pursuing endeavors other than our secondary contribution.

The authors of [9, Sec. 5] succinctly mentioned using random binary sequences appended to the MAC addresses prior to hashing (or to replace MAC addresses with tokens, more specifically, universally unique Identifiers (UUIDs) [10]). Our secondary contribution uses a similar idea, except that we prepend random sequences a central server partially generates and then shares with time-synchronized sensors. Each sequence is used simultaneously by all sensors during one minute, a time after which the server and the sensors erase it. Thus, brute force attacks consist in recovering a pepper of high entropy instead of MAC addresses, whose entropy is too low to withstand such attacks [9, 11, 12]. We also split peppers into two parts, with one unknown to the server.

Section II focuses on the main contribution, which lays the foundations for a theoretical validation of our secondary contribution, which Section III presents.

II Expected number of collisions with uniformly distributed identifiers

Variable $m$ denotes a number of possible outputs, such that $\log_2(m) \in \mathbb{N}$, and $\{0, 1\}^\gamma$ denotes the set of all binary sequences of $\gamma$ bits. We consider a function $h : \mathcal{X} \rightarrow \{0, 1\}^{\log_2(m)}$ (with $n := \text{card}(\mathcal{X})$). Hereafter, $h$ is either a hash function or a token generator, whose output is approximately uniformly distributed in $\{0, 1\}^{\log_2(m)}$ [2, Sec. 9.7.1].

Following standard terminology in the study of hash tables, we refer to $m$ and $n$ as the number of buckets and the number of inserts, respectively. Similarly, $\alpha := n/m$ is called the load factor. Finally, $Y^{(n,m)}$ denotes the (random) number of collisions when inserting $n$ values into $m$ buckets (with the uniform distribution assumption). Theorem 1 provides the exact—yet numerically unstable—formula of $\mathbb{E}[Y^{(n,m)}]$. The numerical instability appears for sufficiently high values of $m$.

Theorem 1. For $n$ inserts into $m$ buckets, the collision rate, $\mathbb{E}[Y^{(n,m)}]/n$, is

$$\frac{\mathbb{E}[Y^{(n,m)}]}{n} = 1 - \frac{m}{n} \left(1 - \left(\frac{m-1}{m}\right)^n\right),$$

where the uniform distribution assumption has been used.

Proof. See the Appendix. \qed

Theorem 2 proposes three approximations of $\mathbb{E}[Y^{(n,m)}]/n$.

Theorem 2. For a degree of approximation $K \geq 2$, a number of inserts $n \geq 2$, and a load factor $\alpha \leq 1$, there exist error terms $\delta(\alpha, n)$ and $R_{K-1}(\alpha)$ such that

$$\frac{\mathbb{E}[Y^{(n,m)}]}{n} = 1 - \alpha^{-1} (1 - \exp(-\alpha)) + \delta(\alpha, n)$$

$$= \sum_{k=1}^{K-1} \frac{\alpha^k (-1)^{k+1}}{(k+1)!} + \delta(\alpha, n) + R_{K-1}(\alpha)$$

$$= \frac{\alpha}{2} + \delta(\alpha, n) + R_2(\alpha),$$

where

$$-\sqrt{\frac{\alpha^2}{n^2} - \alpha^2 \left(\frac{\pi^2}{6} - 1\right)} \leq \delta(\alpha, n) \leq 0,$$
\[ |R_{K-1}(\alpha)| \leq \frac{\alpha^K}{(K+1)!}, \]  

and, in particular,
\[ \frac{|R_1(\alpha)|}{\alpha/2} \leq \frac{\alpha}{3}. \]  

Proof. See the Appendix. \qed

Equation (2) yields a first approximation that is not numerically stable for low values of \( \alpha \). Equation (3) provides a numerically stable approximation whose precision is controlled through \( K \).

The error term \( \delta(\alpha, n) \) quantifies to what extent \( (1 - \alpha/n)^n \) accurately approximates \( \exp(-\alpha) \). The term \( R_{K-1}(\alpha) \) bounds the error tied to approximating \( \exp(-\alpha) \) using its \( K \)th-order Taylor polynomial.

For low values of \( \alpha \) (e.g., \( \alpha \leq 10^{-3} \)), \( (4) \) is an accurate approximation because \( |R_1(10^{-3})|/(10^{-3}/2) \leq 10^{-3}/3 \) (see (7)), i.e., the error \( |R_1(\alpha)| \) is less than 0.1\% of the approximated value \( \alpha/2 \). For \( \alpha \leq 1 \) and for \( n \) high enough (say, \( n \geq 100 \)), \( \alpha^2/(n^2 - \alpha^2) \simeq \alpha^2/n^2 = 1/m^2 \). Thus, with \( m \geq 2^{64} \), \( |\delta(\alpha, n)| \leq m^{-1} \cdot 0.8031 \leq 5 \cdot 10^{-20} \).

III The probe request anonymization procedure

III.A System overview

We now turn to the anonymization procedure of MAC addresses, whose theoretical validation relies on Theorem 2. As depicted in Figure 1, we designed a system i) comprising several time-synchronized WiFi sensors collecting PRs in their respective locations and ii) a central server collecting as well as processing PRs. The sensors may have overlapping ranges; thus, in order to detect identical PRs, sensors must generate source address (SA) identifiers that are identical for a given MAC address and time instant. Within the framework of crowd monitoring, the central server computes the rate at which PRs are sent (over time frames of one minute) and then derives an estimate of the number of people in the area covered [6].

There are four requirements our system should meet; SA identifiers should i) be identical across all sensors at any time instant, ii) not allow anyone to recover the original MAC address from the corresponding identifier alone, iii) not allow tracking for more than one minute, and iv) have a collision rate of less than \( 10^{-9} \) for \( 10^7 \) MAC addresses per time frame. (A collision is defined as two SAs being mapped onto the same SA identifier.) The fourth point means that the collision rate remains negligible for up to \( 10^7 \) WiFi devices.

Requirements ii) and iii) guarantee privacy. Requirements i) and iv) enable the central server to compute accurate attendee counts. Should Requirement i) not be met, sensors would return different SA identifiers for identical devices simultaneously detected (because of overlapping detection ranges), thereby inducing a positive counting bias. Requirement iv) ensures a negligible probability of two devices being identified as a single one (which creates a negative counting bias).

We use the SHA-256 hash function in conjunction with a pepper and truncate its output to 64 bits. Thus, \( h : \mathcal{X} \to \{0, 1\}^{64} \) is a truncated SHA-256 hash function whose inputs are 48-bit MAC addresses (\( \mathcal{X} = \{0, 1\}^{48} \)). On the server, we could also generate uniformly distributed identifiers from the hashed identifiers; in this case, the server waits a while until all PRs for a given time frame have been transmitted.

We prepend a time-varying pepper to every MAC address before hashing it. With + denoting the concatenation operation, and mac_address and global_pepper representing respectively the
MAC address to be anonymized and the pepper prepended, $h(\text{global pepper} + \text{mac address})$ generates the SA identifier. As shown in Figure 2, sensors collect a timestamp, a received signal strength indicator (RSSI), and a source address (the MAC address).

The pepper consists in a concatenation of a fixed 128-bit sensor pepper and a time-varying 128-bit server pepper. The central server maintains an up-to-date array of 20 server peppers for a duration of 20 minutes that sensors periodically fetch using an HTTPS link with transport security layer (TLS). Sensors use each server pepper for a specific one-minute time frame. Server peppers are generated using a pseudo random number generator (PRNG) (e.g., /dev/urandom or /dev/random on Linux). If this PRNG is deemed not secure (see [13]), hardware PRNG generators are alternatives too [14, 15]. We can also generate a specific set of peppers for each cluster of sensors.

The server and the sensors delete server peppers once they become outdated—in particular, the sensors erase the volatile memory chunk storing server peppers before updating it with new peppers retrieved from the server.

The fixed sensor pepper forms a last line of defense in case the server peppers get compromised. It is written in a file or in the codebase of the sniffer, and it is never stored on the server.

Let us now prove that our four requirements are met.
III.B Requirement 1: peppers are identical across all sensors at a given time instant

This requirement depends on the accuracy of time synchronization. We propose to use network time protocol (NTP), which implies accurate time synchronization on low-latency networks (e.g., 4G networks with timing errors lower than 10 ms [16]). There could be synchronization-related mismatches at the frontiers of consecutive one-minute time frames but only for 20 ms/60000 ms = 0.033 % of their duration.

III.C Requirement 2: impossibility to recover the original MAC address from anonymous identifiers

Cryptographic hash functions like SHA-256 cannot be directly reversed—in practice, reversing consists in trying many of the possible inputs until finding one whose hash is the output to be reversed. Assuming an attacker knows the input MAC address of a particular entry in the list of PRBs, brute forcing the pepper entails testing many of the 256-bit sequences that exist (on average, half of them should be tested). For example, 1 million Nvidia RTX 2080 SUPER Founders Edition graphics cards can compute roughly 5700 SHA-256 TeraHashes per second [17]—this implies that testing all 256-bit peppers (approximately 1.16 $10^{65}$ TeraHashes) takes $2.04 \times 10^{61}$ seconds, i.e., 6.47 $10^{53}$ years. Should one of the two 128-bit peppers be known to an attacker, testing all 128-bit sequences still takes roughly $1.90 \times 10^{15}$ years. We point out that relying on a regular SHA-256 hash function without peppers is not safe (see [9, 12] and [11, Sec. VI]) as the entropy of MAC addresses is too low to resist brute force attacks. Moreover, using computationally intensive hashes like bcrypt [18] and Argon2 [19] would imply unreasonable computational requirements for sensors (see also [9, Sec. 5]).
III.D  Requirement 3: preventing tracking for more than one minute

This requirement is linked to server peppers being updated between consecutive time frames of one minute (as mentioned in Section III.A). In particular, the avalanche effect of SHA-256 hash functions makes hashing with different peppers return incomparable SA identifiers for a given MAC address.

III.E  Requirement 4: a collision rate of less than $10^{-9}$ for $10^7$ MAC addresses

We have $m = 2^{64} \approx 1.84 \times 10^{19}$, which means that we truncate SHA-256 hashes to 64 bits. This corresponds to a load factor $\alpha = 10^7(1.84)^{-1}10^{-19} \approx 10^{-12}$ for $n = 10^7$ MAC addresses. As shown in the paragraph below, it yields a collision rate of about $10^{-12.5}$, and it makes computer implementations of the system straightforward (most of the databases on the market support 64-bit integer/BIGINT fields).

Figure 3 shows that the collision rate is approximately equal to $10^{-12.5}$. For $\alpha$ sufficiently low, (e.g., $\alpha \leq 10^{-3}$), the approximation becomes [4], which explains why the level sets in Figure 3 appear to be linear slopes.

Note that approximation errors are negligible. Our load factor $\alpha \approx 10^{-12}$ implies (for $K \geq 2$) $|R_{K-1}(\alpha)| \leq 10^{-24}$. Moreover, as already pointed out in our comment of Theorem 2 for $m \geq 2^{64}$, $|\delta(\alpha,n)| \leq 5 \times 10^{-20}$.

![Figure 3: Levels sets of the approximation (3) of the collision rate as a function of the number of inserts $n$ and the number of buckets $m$.](image)

Acknowledgments

We thank Innoviris for funding this research through the MUFINS project. We also thank the Icity.Brussels project and FEDER/EFRO grant for the support they provided.
Appendix: Proofs

.A Proof of Theorem 1

Let $p_j$ denote the probability the $j$th ($1 \leq j \leq m$) bucket be empty after $n$ inserts. All inserts have equal probabilities to fall within each bucket and whether an insert ends up in one bucket is independent of which buckets are already occupied. As a result, we have $p_j = ((m-1)/m)^n$ (n inserts and, for each insert, a probability $(m-1)/m$ that it ends up in any bucket except the $j$th one). The expectation of the number of empty buckets is equal to $\sum_{j=1}^{m} E[A_j] = m((m-1)/m)^n$, where $A_j = 1$ if the $j$th bucket is empty and equals 0 otherwise. Hence, the expectation of the number of occupied buckets is $m - m((m-1)/m)^n$. As the number of collisions is equal to $n - \text{“number of occupied bucket”}$, the proof is complete.

.B Lemmas for Theorem 2

To prove Theorem 2, we shall first derive two lemmas. Lemma 1 quantifies to what extent $(1 - \alpha/n)^n$ is a good approximation of $\exp(-\alpha)$.

Lemma 1. For $n \geq 1$ and $\alpha < n$,

$$(1 - \frac{\alpha}{n})^n = \exp(-\alpha)F(\alpha, n),$$

where

$$\exp\left(-\alpha^2 \sqrt{\frac{1}{n^2} - \alpha^2 \left(\frac{\pi^2}{6} - 1\right)}\right) \leq F(\alpha, n) \leq 1.$$ 

Proof. For $0 \leq \alpha/n < 1$,

$$(1 - \frac{\alpha}{n})^n = \exp\left(n \log\left(1 - \frac{\alpha}{n}\right)\right) = \exp\left(-n \sum_{k=1}^{\infty} \frac{(\alpha/n)^k}{k}\right) = \exp\left(-\alpha \left(1 + \sum_{k=1}^{\infty} \frac{(\alpha/n)^k}{k+1}\right)\right).$$

Defining $f^{(K)}(\alpha, n) := \sum_{k=1}^{K} (\alpha/n)^k/(k+1)$, we have, $0 < f^{(1)}(\alpha, n) < f^{(2)}(\alpha, n) < \cdots$ so that if for all $K$, $f^{(K)}(\alpha, n) \leq \xi(\alpha, n)$. The sum in $f^{(K)}(\alpha, n)$ is the inner product between vectors $((\alpha/n)^k)_{1 \leq k \leq K}$ and $(1/(k+1))_{1 \leq k \leq K}$. Cauchy-Schwarz inequality yields:

$$f^{(K)}(\alpha, n) \leq \left\|(\alpha^n/n^k)_{1 \leq k \leq K}\right\|^2_2 \left\|(1/(k+1))_{1 \leq k \leq K}\right\|^2_2.$$
We have, using an asymptotic expression for geometric series,

\[
\left\| \left( \frac{\alpha^k}{n^k} \right)_{1 \leq k \leq K} \right\|_2^2 = \sum_{k=1}^K \left( \frac{\alpha}{n} \right)^k \\
= \sum_{k=0}^K \left( \frac{\alpha}{n} \right)^k - 1 \\
\leq \sum_{k=0}^\infty \left( \frac{\alpha}{n} \right)^k - 1 \\
= \frac{1}{1 - \alpha^2/n^2} - 1 \\
= \frac{\alpha^2}{n^2 - \alpha^2}.
\]

Moreover,

\[
\left\| \left( \frac{1}{k+1} \right)_{1 \leq k \leq K} \right\|_2^2 = \sum_{k=1}^{K+1} \frac{1}{k^2} - 1 \\
\leq \sum_{k=1}^\infty \frac{1}{k^2} - 1 \\
= \zeta(2) - 1,
\]

where \( \zeta(2) \) is the Riemann zeta function evaluated at 2, which is equal to \( \pi^2/6 \). Therefore, the upper bound \( \xi(\alpha, n) \) may be

\[
\xi(\alpha, n) := \alpha \sqrt{\frac{1}{n^2 - \alpha^2}} \sqrt{\frac{\pi^2}{6} - 1}.
\]

Injecting these results in (8) as well as noticing that \( \sum_{k=1}^\infty (\alpha/n)^k/(k+1) \geq 0 \) conclude the proof.

We now turn to a lemma focusing on the accuracy of a polynomial approximation of \( \alpha^{-1}(1-\exp(-\alpha)) \).

**Lemma 2.** For \( 0 < \alpha < 1 \), \( K \geq 1 \) and \( g : [0, 1] \subset \mathbb{R} \to [0, \infty) : \alpha \mapsto g(\alpha) = \alpha^{-1}(1-\exp(-\alpha)) \),

\[
g(\alpha) = \sum_{k=0}^{K-1} \frac{\alpha^k}{(k+1)!}(-1)^k + R_{K-1}(\alpha)
\]

where

\[
|R_{K-1}(\alpha)| \leq \frac{\alpha^K}{(K+1)!}.
\]

**Proof.** With \( \ell(\alpha) := 1 - \exp(-\alpha) \), it is easy to compute that

\[
\frac{d\ell}{d\alpha}(x) = (-1)^{k+1} \exp(-x).
\]

Thus,

\[
\max_{x \in [0,1]} \left| \frac{d\ell}{d\alpha}(x) \right| = 1.
\]
Taylor’s theorem [*20*, Theorem 5.15] shows that the $K$th-order Taylor polynomial of $\ell(\alpha)$ around zero has a remainder $R_K(\alpha)$, for which $|R_K(\alpha)| \leq \alpha^{K+1}/(K+1)!$. The desired $(K-1)$th-order polynomial approximation is:

$$\alpha^{-1}(1 - \exp(-\alpha)) = \alpha^{-1}\left(1 - \sum_{k=0}^{K} \frac{\alpha^k}{k!} (-1)^k\right)$$

$$= \sum_{k=0}^{K-1} \frac{\alpha^k}{(k+1)!} (-1)^k,$$

and the $(K-1)$th-order remainder is $R_{K-1}(\alpha) = \alpha^{-1}R_K(\alpha)$ and satisfies $|R_{K-1}(\alpha)| \leq \alpha^K/(K+1)!$.

### C Proof of Theorem 2

With $\alpha = n/m$, Theorem 1 and Lemma 1, we derive

$$\frac{\mathbb{E}[Y^{(n,m)}]}{n} = 1 - \frac{m}{n}\left(1 - \left(\frac{m-1}{m}\right)^n\right)$$

$$= (1 - \alpha^{-1})(1 - \exp(-\alpha/n))$$

$$= 1 - \alpha^{-1}(1 - \exp(-\alpha)\theta(n,m))$$

For $n \geq 2$ and $\alpha < 1$, $\frac{\alpha^2}{n^2 - \alpha^2} \frac{\pi^2}{6} - 1$ is monotonically decreasing with $n$ and monotonically increasing with $\alpha$, and it is approximately equal to $0.4637 < 1$ for $n = 2$ and $\alpha = 1$. Hence, for $n \geq 2$, $1 - x \leq \exp(-x)$ (for $x < 1$) and

$$\left(1 - \alpha^{-1}\sqrt{\frac{1}{n^2 - \alpha^2} \left(\pi^2/6 - 1\right)}\right) \leq \theta(n,m) \leq 1.$$ 

Therefore,

$$\theta(n,m) = \frac{\mathbb{E}[Y^{(n,m)}]}{n} - (1 - \alpha^{-1}(1 - \exp(-\alpha))) \leq 1,$$

where, for $\alpha \in [0,1]$,

$$\theta(n,m) := -\alpha^{-1}\exp(-\alpha)\sqrt{\frac{1}{n^2 - \alpha^2} \left(\pi^2/6 - 1\right)}$$

$$\geq -\sqrt{\frac{\alpha^2}{n^2 - \alpha^2} \left(\pi^2/6 - 1\right)}$$

because $-\exp(-\alpha) \geq -\exp(0) = -1$ for $\alpha \in [0,1]$. Equation (2) of Theorem 2 is now proven. Lemma 2 implies

$$1 - \alpha^{-1}(1 - \exp(-\alpha)) = 1 - \sum_{k=0}^{K-1} \frac{\alpha^k}{(k+1)!} (-1)^k + R_{K-1}(\alpha)$$

$$= \sum_{k=1}^{K-1} \frac{\alpha^k}{(k+1)!}(-1)^{k+1} + R_{K-1}(\alpha),$$

which proves (3). Deriving (4) and (7) is straightforward.
References

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to algorithms. MIT press, 2009.

[2] A. J. Menezes, J. Katz, P. C. Van Oorschot, and S. A. Vanstone, Handbook of applied cryptography. CRC press, 1996.

[3] M. J. Wiener, “The full cost of cryptanalytic attacks,” Journal of Cryptology, vol. 17, no. 2, pp. 105–124, 2004.

[4] M. Gast, 802.11 wireless networks: the definitive guide. ” O’Reilly Media, Inc.”, 2005.

[5] M. Uras, R. Cossu, E. Ferrara, A. Liotta, and L. Atzori, “PmA: A real-world system for people mobility monitoring and analysis based on Wi-Fi probes,” Journal of Cleaner Production, p. 122084, 2020.

[6] J.-F. Determe, U. Singh, F. Horlin, and P. De Doncker, “Forecasting Crowd Counts With Wi-Fi Systems: Univariate, Non-Seasonal Models,” IEEE Transactions on Intelligent Transportation Systems, 2020.

[7] U. Singh, J.-F. Determe, F. Horlin, and P. De Doncker, “Crowd Forecasting based on WiFi Sensors and LSTM Neural Networks,” IEEE Transactions on Instrumentation and Measurement, 2020.

[8] U. Singh, J.-F. Determe, F. Horlin, and P. D. Doncker, “Crowd monitoring: State-of-the-art and future directions,” IETE Technical Review, 2020. [Online]. Available: https://doi.org/10.1080/02564602.2020.1803152

[9] L. Demir, M. Cunche, and C. Lauradoux, “Analysing the privacy policies of Wi-Fi trackers,” in Proceedings of the 2014 workshop on physical analytics, 2014, pp. 39–44.

[10] P. Leach, M. Mealling, and R. Salz, “A universally unique identifier (UUID) URN namespace,” 2005.

[11] L. Demir, A. Kumar, M. Cunche, and C. Lauradoux, “The pitfalls of hashing for privacy,” IEEE Communications Surveys & Tutorials, vol. 20, no. 1, pp. 551–565, 2017.

[12] M. Marx, E. Zimmer, T. Mueller, M. Blochberger, and H. Federrath, “Hashing of personally identifiable information is not sufficient,” SICHERHEIT 2018, 2018.

[13] Y. Dodis, D. Pointcheval, S. Ruhault, D. Vergniaud, and D. Wichs, “Security analysis of pseudo-random number generators with input: /dev/random is not robust,” in Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, 2013, pp. 647–658.

[14] M. Stipčević and B. M. Rogina, “Quantum random number generator based on photonic emission in semiconductors,” Review of scientific instruments, vol. 78, no. 4, p. 045104, 2007.

[15] Z. Zheng, Y. Zhang, W. Huang, S. Yu, and H. Guo, “6 gbps real-time optical quantum random number generator based on vacuum fluctuation,” Review of Scientific Instruments, vol. 90, no. 4, p. 043105, 2019.
[16] R. Miškinis, D. Jokubauskis, D. Smirnov, E. Urba, B. Malyško, B. Dzindzelėta, and K. Svirskas, “Timing over a 4G (LTE) mobile network,” in *2014 European Frequency and Time Forum (EFTF)*. IEEE, 2014, pp. 491–493.

[17] “Nvidia RTX 2080 SUPER FE Hashcat Benchmarks,” [https://gist.github.com/epixoip/47098d25f171ec1808b519615be1b90d](https://gist.github.com/epixoip/47098d25f171ec1808b519615be1b90d), accessed: 2020-08-13.

[18] N. Provos and D. Mazieres, “A future-adaptable password scheme.” in *USENIX Annual Technical Conference, FREENIX Track*, 1999, pp. 81–91.

[19] A. Biryukov, D. Dinu, and D. Khovratovich, “Argon2: new generation of memory-hard functions for password hashing and other applications,” in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 292–302.

[20] W. Rudin *et al.*, *Principles of mathematical analysis*. McGraw-hill New York, 1964, vol. 3.