# NISK 2018

## PROCEEDINGS

*NORSK INFORMASJONSSIKKERHETS-KONFERANSE*

## The 11th Norwegian Information Security Conference

19-20 SEPTEMBER 2018
LONGYEARBYEN, SVALBARD

Samlokalisert med NIKT


Denne proceedings publiseres i NISK tidsskriftserie
gjennom Open Journal Systems hos Bibsys

# Message from the Program Chairs

This compilation is the pre-proceedings of the papers to be presented at NISK 2018: the 11th Norwegian Information Security Conference 2018 held on September 18-20, 2018 in Longyearbyen, Svalbard. All presented papers will be published in the NISK open access series of proceedings (level 1) by the Open Journal Systems at Bibsys.

The aim of the NISK conference series is to be the principal Norwegian research venue for presenting and discussing developments in the field of ICT security and privacy, and bringing together people from universities, industry, and public authorities. We invite both national and international contributions by researchers, practitioners, and PhD- and Master thesis students presenting new problems and solutions within topics on ICT security. All papers must be original and not simultaneously submitted to another conference or journal. Contributions can be written in English or Norwegian. The authors will retain all rights to the published article, no copyright transfer agreement is requested.

The programme committee included 25 experts covering the broad field of information security. We were able to recruit our PC members both from universities, research institutes, industries, and public organisations. The main load of review work took place in the time period of April and May 2018. The review process was performed in a timely manner thanks to the very good responsiveness of the committee members. We used the Easychair web-based system for the submission, review, and selection process.

The call for papers was published in February, and we received by end of April in total 28 submissions authored by altogether 58 researchers, distributed over these countries: Norway (42), Germany (8), Canada (2), Turkey (2), Bangladesh (1), Israel (1), UK (1), USA (1). Three double blind reviews were done for each regular research paper, whereas short papers received two independent reviews, also double blind. The programme chairs accepted 9 of the submissions as regular research papers, and 3 of the submissions as short papers. Finally, 2 submissions were accepted as a one page abstract and poster. All of these 14 contributions are scheduled for presentations at the two day conference programme, divided into five topical sessions: Crypto-primitives, Crypto-protocols, Security Analysis, Biometrics, and Malware.

The NISK keynote theme for this year was motivated by the recent important European policy impact of personal data protection and privacy regulations. The invited keynote talk is by Elise K. Lindeberg, director of the security department at the Norwegian Communication Authority. The title of the talk is *Spenningsfeltet innovasjon, digitalisering og kommunikasjonsvern* (Editor's translation: The Tension Field of Innovation, Digitization, and Communication Protection.)

We will also note the contributions of our national PhD student research school of computer and information security (COINS), headed by Hanno Langweg. COINS provided financial support for PhD student participation, and organised a PhD-student workshop the day before the conference. The NIKT conference budget sponsored one masterstudent presenting a NISK paper.

Over 60 registered conference participants this year sets a grand attendance record for the annual NISK conference series. Thank you very much to Ingrid Chieh Yu and her staff at the University of Oslo for an excellent organisation of all practical arrangements for the multi-conference NIKT. You made it smooth sailing!

Trondheim, September 11, 2018 *Stig Frode Mjølsnes and Ragnar Soleng*

# Program Committee

| | |
|---|---|
| Stig F. Mjølsnes (chair) | NTNU |
| Ragnar Soleng (co-chair) | UiT |
| | |
| Anders Andersen | UiT |
| Martin Eian | Mnemonic AS |
| László Erdödi | UiO |
| Danilo Gligoroski | NTNU |
| Tor Helleseth | UiB |
| Martin Gilje Jaatun | SINTEF |
| Audun Jøsang | UiO |
| Geir M. Køien | UiA |
| Hanno Langweg | HTWG Konstanz |
| Chunlei Li | UiB |
| Marie Moe | SINTEF |
| Leif Nilsen | Thales Norway AS |
| Nils Nordbotten | FFI |
| Vladimir Oleshchuk | UiA |
| Ruxandra F. Olimid | Univ. of Bucharest |
| Tord Reistad | Statens Vegvesen |
| Sondre Rønjom | NSM |
| Einar Snekkenes | NTNU |
| Christian Tellefsen | Thales Norway AS |
| Mohsen Toorani | UiB |
| Svein Willassen | Confrere AS |
| Øyvind Ytrehus | SimulaUiB |
| Lasse Øverlier | FFI |

# Table of Contents

**5. MALWARE**

*Chair: Stig F. Mjølsnes*

# Author Index

# Keyword Index

# A Successful Subfield Lattice Attack on a Fully Homomorphic Encryption Scheme

Martha Norberg Hovd

Simula@UiB
University of Bergen
martha.hovd@uib.no

**Abstract**

We present the application of a known subfield lattice attack on a fully homomorphic encryption scheme based on NTRU. It is known that for this attack to be successful, a parameter of the scheme must satisfy a lower bound. We derive a second lower bound on the same parameter and show that this bound must be respected if the scheme is to be functional, and furthermore that, in all practical instances of the scheme, the derived second lower bound is greater than the lower bound required for the attack to be applicable. Hence, the scheme is shown to be susceptible to the subfield lattice attack, and furthermore that this susceptibility is inevitable given the current structure of the scheme.

## 1 Introduction

Fully homomorphic encryption (FHE) schemes are encryption schemes with the following property: for any function $f$, $\mathrm{Dec}(f(c)) = f(\mathrm{Dec}(c))$, where $c = \mathrm{Enc}(m)$ for a message $m$. The first such scheme was presented in 2009 [4], and several schemes have been presented since. They mostly follow the same structure and have the same starting point: an encryption scheme where both multiplication and addition of **freshly generated** ciphertexts are homomorphic: $\mathrm{Dec}(\mathrm{Enc}(m_1) + \mathrm{Enc}(m_2)) = m_1 + m_2$, and $\mathrm{Dec}(\mathrm{Enc}(m_1)\mathrm{Enc}(m_2)) = m_1 m_2$.

All these schemes add bounded randomness to the plaintext to obscure it; this randomness is also referred to as 'noise'. The problem is that as more operations are performed on a ciphertext, this noise may grow until it no longer respects the required bounds. At this point, the noise is said to have become unmanageable, as we have no guarantee of correct decryption and the scheme is merely somewhat homomorphic.

In order to have a fully homomorphic scheme, we must reduce the noise, which is usually achieved through a combination of operations. These may stunt the growth of noise, or reduce it slightly. When these no longer suffice, bootstrapping is applied: a homomorphic evaluation of the decryption algorithm. Bootstrapping reduces the noise sufficiently to allow for homomorphic evaluation of any function - it is, however, a very time-consuming procedure. It is, therefore, preferable to construct a fully homomorphic scheme by relying on other strategies and using bootstrapping only as a last resort, as a scheme heavily dependent on bootstrapping is very impractical.

In some cases, the somewhat homomorphic 'starting schemes' are based on previously known schemes, but with different parameter settings, which may result in a less secure scheme. We present one such case, namely a FHE scheme based on NTRU, presented in [7]. We also present an employable attack, described fully in [1]. However, the article [1] only describes the attack, without noting that it may be applied to the NTRU-based fully homomorphic encryption scheme discussed here. We show this, and also how the susceptibility of this attack is inevitable. This is done by rigorously deriving a lower bound on a crucial parameter of the encryption scheme,

and showing that this bound must be satisfied if the scheme is to be functional. Furthermore, it is noted that another lower bound on the same parameter must be met for the attack to be applicable, and finally that this lower bound is typically smaller than the lower bound derived from the scheme itself. It follows that the scheme may be attacked in all practical instances of it.

It is important to note that it is the derived lower bound which makes the scheme vulnerable to the subfield lattice attack, and furthermore that this lower bound is a result of the additional noise-reducing operations of the scheme. In particular, this means that the original NTRU encryption scheme does not share the same vulnerability to this attack.

# 2 Preliminaries

## 2.1 Notation

All vectors are row vectors and will be denoted with bold lower case letters: $\mathbf{v}, \mathbf{w}$, while matrices will be denoted using bold upper case letters: $\mathbf{A}, \mathbf{B}$. Elements of either a vector, a matrix or a (polynomial) ring will be denoted with a lower case letter in italics: $a, b$. Vectors will be written as $\mathbf{a} = [a_1, a_2, \ldots, a_n]$, whereas sets will be denoted by $\{0, 1, \ldots\}$.

Multiplication of integers, or an integer and a vector or polynomial is denoted by simple juxtaposition: $ab, a\mathbf{v}, af(x)$. Multiplication of a vector and a matrix will be denoted by a single dot: $\mathbf{v} \cdot \mathbf{A}$, and finally, the multiplication of two polynomials will be denoted by an asterisk: $f * g$. Furthermore, this polynomial multiplication always takes place in some polynomial ring, and the main motivation of the multiplicative notation is to serve as a reminder of this during computations. It should be clear from the context whether or not a given element is a polynomial, and any polynomial $f$ will therefore, with very few exceptions, not be written $f(x)$.

Let $\mathbf{v}, \mathbf{w}$ be vectors of the same length $k$ over a polynomial ring $R$. We then define the dot product of these two vectors as $\langle \mathbf{v}, \mathbf{w} \rangle = \sum_{i=1}^{k} v_i * w_i \in R$. In addition, we have the following notation: for any two polynomials $a = \sum_{i=0}^{n-1} a_i x^i$, $b = \sum_{i=0}^{n-1} b_i x^i$, let $[a, b]$ denote the vector $[a_0, \ldots, a_{n-1}, b_0, \ldots, b_{n-1}]$.

We operate with the two following modular reductions: $p = r \mod q$ reduces $p$ modulo $q$ to $r \in (-q/2, q/2]$, whilst $p = r \mod q$ denotes the modular reduction to $r \in [0, q-1]$. Note that the only difference in the notation is the underlining of the first mod. In both cases, we may also write $p \equiv r \mod q$ or $p \equiv r \mod q$ if we wish to stress that $p$ is equivalent to $r$ modulo $q$: $p = r + kq$, for $k \in \mathbb{Z}$. This generalizes to vectors and polynomials.

The Euclidean norm of a vector $\mathbf{v}$ is denoted by $\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$, whilst $\| \cdot \|_\infty$ denotes the infinity norm: $\|\mathbf{v}\|_\infty = \max_i\{|v_i|\}$. Supposing $f$ is a polynomial, $\|f\|, \|f\|_\infty$ refers to calculating either norm of the coefficient vector of $f$.

For a probability distribution $\chi$, $x \leftarrow \chi$ refers to drawing $x$ according to $\chi$. Furthermore, any logarithm log will be to the base 2.

Finally, throughout this paper, the following lemma will prove quite useful. Here, $R = \mathbb{Z}[x]/f(x)$ for a polynomial $f$ of degree $n$.

**Lemma 2.1.** *The following bound holds for any two elements $a, b \in R$:*

$$\|a * b\|_\infty \leq n\|a\|_\infty\|b\|_\infty.$$

*Proof.* Seeing as $a_i \leq \|a\|_\infty,\ b_i \leq \|b\|_\infty\ \forall i \in \{0, 1, \ldots, n-1\}$, it follows that $a_i b_j \leq \|a\|_\infty\|b\|_\infty$. Since every coefficient of $a*b$ is the sum of $n$ terms $a_i b_j$, it holds that $\|a*b\|_\infty \leq n\|a\|_\infty\|b\|_\infty$. □

## 2.2 Lattices

**Definition 2.2.** *Let $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\eta\}$ be a set of linearly independent vectors, with $\mathbf{v}_i \in \mathbb{R}^m$ $\forall i \in \{1, \ldots, \eta\}$. The lattice $\mathcal{L}$ generated by $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\eta$ is the set of linear combinations of these vectors with coefficients in $\mathbb{Z}$:*

$$\mathcal{L} = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_\eta\mathbf{v}_\eta : a_1, a_2, \ldots, a_\eta \in \mathbb{Z}\}.$$

A basis for the lattice $\mathcal{L}$ is any set of independent vectors that generates $\mathcal{L}$, and any two such sets will have the same dimension. Suppose $m = \eta$, we may then represent a basis by a square matrix (where the basis vectors form the rows of the matrix) and so we may calculate the determinant of it. There are of course many possible bases of a lattice $\mathcal{L}$, but as Proposition 7.14 of [6] shows, any two bases of a lattice are related by an integer matrix with determinant $\pm 1$. It follows from this that for any two basis matrices $\mathbf{B}, \mathbf{B}'$ we have: $|\det(\mathbf{B})| = |\det(\mathbf{B}')|$. Based on this, we have the following lattice invariant:

**Definition 2.3.** *Let $\mathcal{L}$ be a lattice of dimension $\eta$ with basis $B = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_\eta\}$, where $\mathbf{v}_i \in \mathbb{R}^\eta \quad \forall i \in \{1, 2, \ldots, \eta\}$. The determinant of $\mathcal{L}$ is defined as*

$$\det(\mathcal{L}) = |\det(\mathbf{B})|.$$

Any vector $\mathbf{v} \in \mathcal{L}$ has a (Euclidean) length. Given this property, we may formulate the shortest vector problem of a lattice $\mathcal{L}$ [6]:

The shortest vector problem (SVP): Find a shortest nonzero vector in a lattice $\mathcal{L}$, i.e. find a nonzero vector $\mathbf{v} \in \mathcal{L}$ that minimizes $\|\mathbf{v}\|$.

It may be shown that solving SVP is NP-hard under the randomized reduction hypothesis [6]. Due to this proven hardness, SVP is used in cryptographic settings, so that breaking an encryption scheme requires solving SVP for a certain instance. However, solving SVP precisely is not always necessary; in some cases, it may suffice to compute merely an approximation of the vectors in question; that is, solving the following problem [6]:

Approximate-SVP: Let $\psi(\eta)$ be a function of the lattice dimension $\eta$ of a lattice $\mathcal{L}$, with $\|\mathbf{v}_0\|$ the length of the shortest vectors in $\mathcal{L}$. Find a vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \leq \psi(\eta)\|\mathbf{v}_0\|$.

Of course, the length of the shortest vector $\mathbf{v}_0 \in \mathcal{L}$ is not always given, but an upper bound on $\|\mathbf{v}_0\|$ is always given by the following theorem:

**Theorem 2.4** (Hermite's Theorem (Theorem 7.25 [6]))**.** *Every lattice $\mathcal{L}$ of dimension $\eta$ has at least one vector $\mathbf{v} \in \mathcal{L}$ satisfying $\|\mathbf{v}\| \leq \sqrt{\eta}\det(\mathcal{L})^{1/\eta}$.*

Another result by Hermite is that for every lattice $\mathcal{L}$ of dimension $\eta$ there exists a Hermite constant, $\gamma_\eta$, such that $\|\mathbf{v}_0\| \leq \sqrt{\gamma_\eta}\det(\mathcal{L})^{1/\eta}$. This constant is generally not known. However, we may use the expression to rephrase the approximate-SVP into the Hermite Shortest Vector Problem [3]:

HSVP: Given a lattice $\mathcal{L}$ and an approximation factor $\alpha > 0$, find a non-zero vector $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{v}\| \leq \alpha \det(\mathcal{L})^{1/\eta}$.

The approximation factor $\alpha$ may be expressed as $\delta^\eta$, in which case $\delta$ is known as the Hermite root factor.

Of course, a solution to any of these problems is seldom apparent given a basis $B$ for a lattice, and the most efficient way of solving any of the presented problems is to find a basis which contains the solution of either stated problem. This is known as basis reduction, and the main algorithms are LLL and a generalisation of it, BKZ, both of which are HSVP-algorithms [3].

LLL works by swapping two vectors in the basis and performing a reduction, whereas BKZ works similarly, only with more than two vectors. The number of vectors BKZ works with is known as the block size, denoted by $\beta$. The larger $\beta$ is, the more precise the result of BKZ will be. Although the algorithms are not fully understood, it is known that BKZ outperforms LLL. BKZ also performs much better, both with respect to time and the resulting approximation factors, than any theoretical bound predicts.

## 2.3 An Introduction to NTRU and its Security

The original NTRU encryption scheme [5] is defined over the polynomial ring $\mathbb{Z}[x]/(x^N - 1)$ for an integer $N$. The integer $q > 1$ is an additional parameter of the scheme, as most operations are performed modulo $q$.

A complete description of the NTRU-based FHE scheme to be presented in section 4 may be found in [7]. The scheme follows the general structure of NTRU quite closely. The secret key of the scheme is a polynomial $f \leftarrow \chi$, for a distribution $\chi$ over $R$, and we require $f$ to be invertible modulo $q$. The public key is defined as $h = f^{-1} * g \bmod q$, for $g \leftarrow \chi$. The main difference in the setup is that this scheme is defined over the polynomial ring $R = \mathbb{Z}[x]/(x^n + 1)$, where we require $n = 2^k$.

A possible attack is to try to find the secret key $f$ based solely on the public information $q$ and $h$, and one way to do this is to reformulate the problem into one related to lattices. This is done by constructing a $2n \times 2n$ basis matrix for a lattice $\mathcal{L}_{\text{NTRU}}$. For an NTRU public key polynomial $h(x) = h_0 + \cdots + h_{n-1}x^{n-1}$, the basis matrix of the lattice $\mathcal{L}_{\text{NTRU}}$ is:

$$\mathbf{B}_{\text{NTRU}} = \begin{bmatrix} 1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{n-1} \\ 0 & 1 & \dots & 0 & -h_{n-1} & h_0 & \dots & h_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & -h_1 & -h_2 & \dots & h_0 \\ 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{bmatrix}.$$

Recall that $h = g * f^{-1}$ and $f * f^{-1} = 1 + qf'$, so we must have $f * h = g + qu$ for some polynomial $u = g * f'$.

**Proposition 2.5.** *For the polynomials $f, g$ and $u$ described above, we have: $[f, -u] \cdot \mathbf{B}_{\text{NTRU}} = [f, g]$.*

*Proof.* The $n$ first coefficients of the resulting vector of $[f, -u] \cdot \mathbf{B}_{\text{NTRU}}$ are obviously $f$. Coefficient $n+1+k$, for $k \in \{0, 1, \dots n-1\}$ is expressed as:

$$\sum_{\substack{i,j=0 \\ i+j=k}}^{n-1} f_i h_j - \sum_{\substack{i,j=0 \\ i+j=k+n}}^{n-1} f_i h_j - q u_k = g_k + q u_k - q u_k = g_k,$$

where the fact that $x^n \equiv -1$ in $R = \mathbb{Z}[x]/(x^n+1)$ has been applied. Hence, $[f, -u] \cdot \mathbf{B}_{\text{NTRU}} = [f, g]$, meaning $[f, g]$ belongs to $\mathcal{L}_{\text{NTRU}}$, as the vector may be expressed as a linear combination of the basis vectors of $\mathcal{L}_{\text{NTRU}}$ using only integers. □

Supposing $[f, g]$ is among the shortest vectors in the lattice $\mathcal{L}_{\text{NTRU}}$, it follows that if an adversary is able to solve SVP in $\mathcal{L}_{\text{NTRU}}$, she is able to compute $f$ based solely on public information, and thus break the scheme. Furthermore, any pair of polynomials $[\bar{f}, \bar{g}]$ with sufficiently small coefficients satisfying the relation $\bar{f} * h = \bar{g} \bmod q$ will also suffice, as will probably any solution to approximate-SVP for an approximation factor smaller than $\sqrt{n}$ [6]. Thus, recovering the secret key $f$ of the encryption scheme reduces to solving approximate-SVP or HSVP for the lattice $\mathcal{L}_{\text{NTRU}}$. We stress again that for this strategy to work, we require the vector $[f, g]$ to be among the shortest vectors in the lattice $\mathcal{L}_{\text{NTRU}}$.

# 3 Subfield Lattice Attack

There may be more efficient attacks than merely applying LLL or BKZ on the lattice basis, depending on the properites of the scheme. We present one such attack here, described fully in [1].

## 3.1 Algebraic Background

Let $\mathbb{K} = \mathbb{Q}[\omega]$ be a field, for a root of unity $\omega$ of order $2n$, for $n$ a power of 2, and let $\mathbb{L}$ be a subfield of $\mathbb{K}$ such that $\mathbb{L} = \mathbb{Q}[\omega']$, for $\omega'$ a root of unity of order $2n'$, where $n'$ also is a power of 2, and define $\rho = n/n'$. These fields will have rings of integers $\mathbb{Z}[\omega]$ and $\mathbb{Z}[\omega']$, respectively. These rings of integers may be shown to be isomorphic to the polynomial rings $R = \mathbb{Z}[x]/(x^n+1)$ and $R' = \mathbb{Z}[x]/(x^{n'}+1)$.[1]

We know from Galois theory that there is a Galois group $G'$ of automorphisms $\{\varphi_i\}$ on $\mathbb{K}$ that fixes $\mathbb{L}$ pointwise [1]. Using these automorphisms, we may define the norm function $\mathrm{N}_{\mathbb{K}/\mathbb{L}} : \mathbb{K} \to \mathbb{L}$, as $\mathrm{N}_{\mathbb{K}/\mathbb{L}}(a) = \prod_{\varphi_i \in G'} \varphi_i(a)$.

## 3.2 The Attack

Given an instance of an NTRU-based encryption scheme, with $sk = f$ and $pk = h = f^{-1} * g$, we define $f' = \mathrm{N}_{\mathbb{K}/\mathbb{L}}(f), g' = \mathrm{N}_{\mathbb{K}/\mathbb{L}}(g), h' = \mathrm{N}_{\mathbb{K}/\mathbb{L}}(h)$ and a new lattice $\mathcal{L}'_{\text{NTRU}}$ defined by $h'$ and $q$ as described in Subsection 2.3. The approach of the attack is to find a short vector $[x', y'] \in \mathcal{L}'_{\text{NTRU}}$ by performing LLL on the basis $B'_{\text{NTRU}}$ and lift this vector up to $[x, y]$ in the original lattice, using the canonical inclusion map. If the vector $[x', y']$ satisfies certain properties, the vector $[x, y]$ will be short in $\mathcal{L}_{\text{NTRU}}$, and might therefore function as a secret key.

The actual attack rests on the following heuristic:

**Heuristic 3.1** (Heuristic 1 [1]). *For any $n$ and any $f, g \in R$ with reasonable isotropic distribution of variance $\sigma^2$ and any constant $c > 0$, there exists a constant $C$ such that $\|f'\| \leq (\sigma n^C)^\rho$ and $\|g'\| \leq (\sigma n^C)^\rho$, except with probability $\mathcal{O}(n^{-c})$.*

Moreover, Theorem 1 of [1] assures us of the existence of a lattice reduction algorithm with block-size $\beta$ which is able to find a vector $[x', y'] \in R'$ such that $\|[x', y']\| \leq \beta^{\Theta(2n'/\beta)}\|\mathbf{v}_0\|$ when applied to the basis of the lattice $\mathcal{L}'_{\text{NTRU}}$, with $\|\mathbf{v}_0\|$ the length of the shortest vectors in the lattice. When combined with the observation that $\|\mathbf{v}_0\| \leq \|[f', g']\|$ and Heuristic 3.1, we conclude that there exists a lattice reduction algorithm able to find a vector $[x', y'] \in R'$ such that

$$\|[x', y']\| \leq \beta^{\Theta(n/\beta\rho)}\|[f', g']\| \leq \beta^{\Theta(n/\beta\rho)}(n\sigma)^{\Theta(\rho)}.$$

Furthermore, we also have the following theorem:

**Theorem 3.2** (Theorem 2 [1]). *Let $f', g' \in R'$ be such that $\langle f' \rangle$ and $\langle g' \rangle$ are coprime ideals[1] and $h' * f' = g' \bmod q$ for some $h' \in R'$. If $[x', y'] \in \mathcal{L}'_{\text{NTRU}}$ has length satisfying*

$$\|[x', y']\| < \frac{q}{\|[f', g']\|}, \tag{1}$$

*then $[x', y'] = v[f', g']$ for some $v \in R'$.*

Based on the result derived from Heuristic 3.1 and Theorem 1 from [1], we conclude that for bound (1) to hold, and therefore for the attack to succeed, we must require

$$\beta^{\Theta(n/\beta\rho)}(n\sigma)^{\Theta(\rho)} \leq q. \tag{2}$$

Once the vector $[x', y']$ has been found, we lift $x', y' \in R'$ to $R$ using the canonical inclusion map $L : \mathbb{L} \to \mathbb{K}$:

$$\begin{aligned} x &= L(x') = L(v) * L(f'), \\ y &= L(y') * h/L(h') \bmod q = L(v) * L(g') * h/L(h') \bmod q, \end{aligned}$$

Here, $v$ is as in Theorem 3.2. For simplicity, we set $\tilde{f} = L(f')/f$, $\tilde{g} = L(g')/g$ and $\tilde{h} = L(h')/h$; we then have

$$\begin{aligned} x &= L(v) * \tilde{f} * f \bmod q \\ y &= L(v) * L(g')/\tilde{h} = L(v) * g * \tilde{g}/\tilde{h} = L(v) * \tilde{f} * g \bmod q \\ \Rightarrow [x, y] &= u * [f, g] \in \mathcal{L}_{\text{NTRU}} \quad \text{with } u = L(v) * \tilde{f} \in R. \end{aligned}$$

In other words: the subfield attack yields a (small) multiplicative of $[f, g]$ under certain reasonable assumptions.

# 4 A Fully Homomorphic Encryption Scheme based on NTRU

## 4.1 The Somewhat Homomorphic Encryption Scheme

As mentioned in Section 2.3, the encryption scheme presented here is fully described in [7] and is defined over the polynomial ring $R = \mathbb{Z}[x]/(x^n + 1)$, for $n$ a power of 2. We also have the

[1] The authors of [1] note that the probability of $\langle f' \rangle$ and $\langle g' \rangle$ being coprime is roughly 3/4, and also that it does not seem strictly necessary for the attack to be successful in practice.

integer parameters $q, p$, chosen such that $q \gg p \geq 2$ and $\gcd(p, q) = 1$. Given these integers, we define the rings $R_p = \mathbb{Z}_p[x]/(x^n + 1)$ and $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. In addition, we have a probability distribution $\chi$, which will typically be some discrete Gaussian distribution. The scheme consists of the following operations:

KeyGen: Draw $f \leftarrow \chi$ such that $f \equiv 1 \mod p$ and $\exists f^{-1} \underline{\mod} q$. Draw $g \leftarrow \chi$ as well, and output $pk = h = g * f^{-1} \underline{\mod} q$ and $sk = f$.

Enc($pk = h, m \in R_p$): Draw $e, r \leftarrow \chi$ such that $e \equiv m \mod p$.
Output $c = pr * h + e \underline{\mod} q$, $d = 1$.

Dec($sk = f, c \in R_q$, $d$): Compute $\bar{b} = f^d * c \underline{\mod} q$ and lift this to the integer polynomial $b \in R$ with coefficients in $(-q/2, q/2]$. Output $m = b \underline{\mod} p$.

EvalAdd($c_0, c_1, d_0, d_1$): Output: $c = c_0 + c_1 \underline{\mod} q$, $d = \max(d_0, d_1)$.

EvalMult($c_0, c_1, d_0, d_1$): Output: $c = c_0 * c_1 \underline{\mod} q$, $d = d_0 + d_1$.

The two last operations are the homomorphic operations, and it is also these that necessitate the notion of the degree $d$ of a ciphertext, which denotes the power of $f^{-1}$ in the ciphertext. Note that $f^k * b = m \underline{\mod} p$ for any power $k \geq 0$, whilst this is not necessarily the case for $f^{-1}$, as there is no guarantee that $f^{-1} = 1 \underline{\mod} p$. Therefore, the decryption procedure will decrypt any ciphertext of degree at most the given $d$, assuming $f^d * c = f^k * b \underline{\mod} q$, which is the reason $d = \max(d_0, d_1)$ in EvalAdd.

The polynomials $f, g, r$ and $e$ must be chosen so that they ensure correct decryption - meaning $\chi$ should have parameters ensuring that these polynomials are "short enough". What precisely this entails will be discussed at some length throughout this section. Essentially: we will derive bounds on the coefficients of these polynomials to ensure correct decryption, even after the noise reducing operations have been performed on a ciphertext. The resulting bounds will be used to derive a final bound on $q$.

We start with the lower bound imposed on the coefficients of the polynomials $f, g, r$ and $e$ to ensure correct decryption of a freshly generated ciphertext.

**Proposition 4.1.** *In order for the presented encryption scheme to correctly decrypt a freshly generated ciphertext, we require every coefficient of the polynomials $f, g, r$ and $e$ to be strictly less than $\sqrt{\frac{q}{4pn}}$.*

*Proof.* The decryption of $c = pr * h + e \underline{\mod} q$ proceeds as follows, when viewed as an operation in $R$, as opposed to $R_q$:

$$\begin{aligned}\bar{b} = f * c = f * (pr * h + e) = pf * r * g * f^{-1} + f * e \\ = pq * r * g * f' + pr * g + f * e,\end{aligned}$$

where $f * f^{-1} = qf' + 1$. Consider the polynomial $pr * g + f * e$ in $R$. To ensure correct decryption, we need every coefficient of this polynomial to be of absolute value less than $q/2$, or else we get $b = pr * g + f * e - q \sum_{i=0}^{n-1} a_i x^i$ where some $a_i \neq 0$ and hence, $b \underline{\mod} p$ need not equal $m$. We therefore require $\|pr * g + f * e\|_\infty < q/2$. Using the triangle inequality and Lemma 2.1, we may compute:

$$\begin{aligned}\|pr * g + f * e\|_\infty &\leq \|pr * g\|_\infty + \|f * e\|_\infty \\ &\leq pn\|r\|_\infty\|g\|_\infty + n\|f\|_\infty\|e\|_\infty\end{aligned}$$

$$\leq pn\|r\|_\infty\|g\|_\infty + pn\|f\|_\infty\|e\|_\infty \leq 2pnB^2, \tag{3}$$

for $B$ a bound on the largest coefficient of $r, g, f$ and $e$. If we assume (3) is less than $q/2$, then any fresh ciphertext will decrypt correctly. This requires the polynomials $r, g, f$ and $e$ to be sampled from a distribution $\chi$ which ensures that any coefficient is strictly less than $\sqrt{\frac{q}{4pn}}$. □

## 4.2 Noise reductions

The scheme in [7] uses key switching, ring reduction, modulus switching and bootstrapping as strategies to reduce the noise of a ciphertext, and thus turn the somewhat homomorphic scheme into a fully homomorphic encryption scheme. However, only key switching and modulus switching are being performed after every multiplication; hence we shall only focus on these two operations.

Note that this, strictly speaking, only makes the scheme presented here somewhat homomorphic, as we need bootstrapping to make it truly fully homomorphic. Nevertheless, we refer to the scheme presented here as a fully homomorphic scheme, mainly to separate it from the "starting scheme" presented in Subsection 4.1, and refer the interested reader to [7] to study the bootstrapping procedure.

### 4.2.1 Key Switching

Key switching converts a ciphertext of degree at most $d$ encrypted under $f_1$ into a ciphertext of degree 1 encrypted under the secret key $f_2$. This procedure requires a hint, namely $a_{1\to 2} = \bar{a} * f_1^d * f_2^{-1} \bmod q$, for $\chi \to \bar{a} \equiv 1 \mod p$. Based on this hint, the actual key switching is the procedure

KeySwitch$(c_1, a_{1\to 2})$: Output: $c_2 = a_{1\to 2} * c_1 \bmod q$.

**Proposition 4.2.** *Suppose $c_1$ is an encryption of $m$ under $f_1$ of degree $d$ which decrypts correctly:* $\mathrm{Dec}(f_1, c_1, d) = m$. *In order to ensure* $\mathrm{Dec}(f_2, c_2, 1) = m$, *for $a_{1\to 2}$ and $c_2$ generated according to the above procedure, we require every coefficient of $f_1, f_2, g, r, e$ and $\bar{a}$ to be strictly less than* $(\frac{q}{2^{d+1}p^d n^{2d}})^{\frac{1}{2d+1}}$.

*Proof.* Decryption of $c_2$ results in:

$$\begin{aligned}\bar{b}_2 &= f_2 * c_2 = f_2 * a_{1\to 2} * c_1 = f_2 * \bar{a} * f_1^d * f_2^{-1} * c_1 \\ &\equiv \bar{a} * f_1^d * c_1 \equiv \bar{a} * \bar{b}_1 \bmod q\end{aligned}$$

In order to have correct decryption, we require $\|\bar{a} * \bar{b}_1\|_\infty < q/2$, to ensure that $b_2 = \bar{a} * b_1 = \bar{a} * m = m \bmod p$. Seeing as $c_1$ is a ciphertext of degree $d$, it must be the result of $d-1$ multiplications, wlog let $\bar{b}_1 = f_1^d * (pr * g * f_1^{-1} + e)^d \bmod q$. For the assumption $\|\bar{a} * \bar{b}_1\|_\infty < q/2$ to hold, we must have:

$$\begin{aligned}\|\bar{a} * f_1^d * (pr * g * f_1^{-1} + e)^d\|_\infty &= \|\bar{a} * f_1^d * \sum_{i=0}^{d} \binom{d}{i} p^i r^i * g^i * f_1^{-i} * e^{d-i}\|_\infty \\ &= \|\bar{a} * \sum_{i=0}^{d} \binom{d}{i} p^i r^i * g^i * f_1^{d-i} * e^{d-i}\|_\infty \\ \text{By Lemma 2.1} \qquad &\leq n^{2d}\|\bar{a}\|_\infty \sum_{i=0}^{d} \binom{d}{i} p^i \|r\|_\infty^i \|g\|_\infty^i \|f\|_\infty^{d-i} \|e\|_\infty^{d-i}\end{aligned}$$

$$\leq p^d n^{2d} B^{2d+1} \sum_{i=0}^{d} \binom{d}{i} \leq 2^d p^d n^{2d} B^{2d+1} < q/2.$$

Here, $B$ is a bound on the largest coefficient in $\bar{a}, r, g, f$ and $e$, and the requirement of this being strictly less than $(\frac{q}{2^{d+1} p^d n^{2d}})^{\frac{1}{2d+1}}$ to ensure correct decryption after switching keys immediately follows. □

It follows that key switching should be performed after every multiplication to minimize this bound. In the case $d = 2$ we have:

$$B^5 < \frac{q}{8p^2n^4}. \tag{4}$$

### 4.2.2 Modulus Switching

Modulus switching converts a ciphertext from modulus $q$ to a smaller modulus, $\bar{q} = q/q'$ for some factor $q'$ of $q$, which reduces the underlying noise by a factor of approximately $q'$. Seeing as $q'|q$, it follows that $\gcd(q', p) = 1 \Rightarrow \exists v$ s.t. $v = (q')^{-1} \bmod p$. The procedure $\text{ModSwitch}(c, q, q')$ is performed as follows:

1. Compute a short $\varrho \in R$ such that $\varrho = c \bmod q'$.
2. Compute a short $\Delta \in R$ such that $\Delta = (q'v - 1)\varrho \bmod (pq')$.
3. Let $\varrho' = c + \Delta \bmod q$. Note that $q'$ divides $\varrho'$ by construction.
4. Output $c' = (\varrho'/q') \in R_{\bar{q}}$.

Note that the final step indirectly multiplies $\varrho$ with $v$, which may easily be compensated for by either multiplying with $q'$ in the final step of the decryption procedure or ensuring that $q' \equiv 1 \mod p$.

**Proposition 4.3.** *Suppose $c$ is an encryption of degree 1 of the message $m$ under the secret key $f$. Let $c' = \text{ModSwitch}(c, q, q')$; then, in order to have $v\text{Dec}(f, c, 1) = \text{Dec}(f, c', 1)$, we require every coefficient of $f, g, r$ and $e$ to be less than or equal to $B$, which again needs to satisfy $\frac{1}{q'}(2pnB^2 + nB\frac{pq'}{2}) < \frac{q}{2q'}$*

*Proof.* Let $\bar{q} = q/q'$. As $\varrho = c \bmod q'$ and $v = (q')^{-1} \bmod p$, we may write

$$\varrho = c - q'l \qquad \text{for } l \in R, \qquad q'v = 1 + pk \qquad \text{for } k \in \mathbb{Z}.$$

Following the procedure, we have[2]:

$$\begin{aligned}
(q'v - 1)\varrho &= (pk + 1 - 1)(c - q'l) = pk(c - q'l) = pkc - pq'kl. \\
\Rightarrow \Delta &= pkc - pq's \text{ for } s \in R, \quad \text{as} \quad R \ni \Delta = (q'v - 1)\varrho \equiv pkc \bmod pq'. \\
\varrho' &= c + \Delta \bmod q = c + pkc - pq's = (1 + pk)c - pq's \\
&\equiv q'vc - pq's \bmod q. \\
c' &= \varrho'/q' \equiv vc - ps \bmod \bar{q}.
\end{aligned}$$

[2] Throughout this proof, $pk$ denotes $p$ multiplied with $k$, not the public key.

In order to guarantee correct decryption, we require $\|vc - ps\|_\infty < \bar{q}/2$, so that:

$$\begin{aligned} f * c' &= vf * c - pf * s = v(pr * g(qf' + 1) + f * e) - pf * s \in R \\ &\equiv vpg * r + vf * e - pf * s \bmod \bar{q} \end{aligned}$$

If we are to have equality rather than equivalence, we must have $\|vf*e+vpg*r-pf*s\|_\infty < \bar{q}/2$, so that: $(f * c' \bmod \bar{q}) \bmod p = vm$. Thus, for the decryption of $c'$ to be successful, we need to have the following:

$$\|f * c'\|_\infty = \|f * (c + \Delta)/q'\|_\infty \leq \frac{1}{q'}(\|f * c)\|_\infty + \|f * \Delta\|_\infty)$$

We use $c = p * r * g * f^{-1} + e$ as well as Lemma 2.1 and derive:

$$\begin{aligned} \frac{1}{q'}(\|pg * r + f * e\|_\infty + \|f * \Delta\|_\infty) &\leq \frac{1}{q'}(2pnB^2 + nB\|\Delta\|_\infty) \\ &\leq \frac{1}{q'}(2pnB^2 + nB\frac{pq'}{2}) < q/2q'. \end{aligned} \tag{5}$$

□

### 4.3 ComposedEvalMult and the Growth of $q$

The operation ComposedEvalMult is simply the sequential execution of EvalMult, KeySwitch and ModSwitch.

**Proposition 4.4.** *Suppose $c_0, c_1$ are two ciphertexts encrypted under the public key $h = g*f_1^{-1}$, both of degree 1. Correctness of ComposedEvalMult means*

$$\mathrm{Dec}(f_2, \mathrm{ComposedEvalMult}(c_0, c_1), 1) = \mathrm{Dec}(f_1, c_0, 1) * \mathrm{Dec}(f_1, c_1, 1),$$

*where $f_2$ is the new secret key after KeySwitch has been performed. To achieve this, we require the polynomials $f_1, f_2, g, r_0, r_1, e_0, e_1$ and $\bar{a}$ to all be drawn from a distribution $\chi$ so that their largest coefficient is smaller than $B$, and that $B$ satisfies*

$$\frac{1}{q'}(4p^2n^4B^5 + nB\frac{pq'}{2}) < \frac{q}{2q'}.$$

*Proof.* Based on the proofs of propositions 4.2 and 4.3, it follows that

$$f_2 * \mathrm{ComposedEvalMult}(c_0, c_1) \equiv \bar{b} \bmod \bar{q},$$

where $\bar{b} = m_0 * m_1 \bmod p$. What needs to be calculated is the bound we require on the drawn polynomials so that the noise added during multiplication and switching keys is sufficiently lowered by switching the modulus. The ciphertext ComposedEvalMult$(c_0, c_1)$ outputs is on the form $c = \frac{1}{q'}(a_{1\to 2} * c_0 * c_1 + \Delta)$ for a factor $q'$ of $q$. We have the following:

$$\begin{aligned} f_2 * c &= f_2 * \frac{1}{q'}(a_{1\to 2} * c_0 * c_1 + \Delta) \\ &= \frac{1}{q'} f_2 * (\bar{a} * f_2^{-1} * f_1^2 * (pr_0 * g * f_1^{-1} + e_0)(pr_1 * g * f_1^{-1} + e_1) + \Delta) \end{aligned}$$

10

$$= \ldots \equiv \frac{1}{q'}(p^2\bar{a} * r_0 * r_1 * g^2 + p\bar{a} * r_0 * g * f_1 * e_1 + p\bar{a} * r_1 * g * f_1 * e_0 + \bar{a} * f_1^2 * e_0 * e_1 + f_2 * \Delta) = b' \equiv \bar{b} \bmod \bar{q}.$$

We need $\|b'\|_\infty < \bar{q}/2$, so $b' = \bar{b}$. To achieve a bound on the coefficients of the polynomials, we use Lemma 2.1 and set

$$\|\bar{a}\|_\infty = \|r_0\|_\infty = \|r_1\|_\infty = \|g\|_\infty = \|f_1\|_\infty = \|f_2\|_\infty = \|e_0\|_\infty = \|e_1\|_\infty = B,$$

and we compute:

$$\begin{aligned} \|b'\|_\infty &\leq \frac{1}{q'}(p^2n^4B^5 + 2pn^4B^5 + n^4B^5 + nB\|\Delta\|_\infty) \\ &\leq \frac{1}{q'}(4p^2n^4B^5 + nB\frac{pq'}{2}), \end{aligned} \tag{6}$$

which has to be smaller than $\frac{q}{2q'}$ if ComposedEvalMult is to be correct. □

Given this final bound all the coefficients of the noise-inducing polynomials need to satisfy, we may use this to at last derive a bound on $q$. This bound will depend on other parameters of the scheme and the probability distribution $\chi$.

Suppose any of the polynomials affecting the noise level are drawn from a discrete Gaussian distribution of parameter $r$, and set $w$ as an assurance measure; meaning it should be practically impossible for any polynomial drawn from this distribution to have a Euclidean length greater than $rw$. It follows that we may set a bound on the infinity norm of any such distributed polynomial as $\frac{rw}{\sqrt{n}}$. Using this bound and expression (6), we require

$$\frac{1}{q'}(4p^2n^4(\frac{rw}{\sqrt{n}})^5 + n\frac{rw}{\sqrt{n}}\frac{pq'}{2}) = \frac{1}{q'}(4p^2n^{1.5}r^5w^5 + \frac{1}{2}pq'\sqrt{n}rw) < q/2q'$$

for decryption to still be correct after a call to ComposedEvalMult.

Suppose $4p^2n^{1.5}r^5w^5 < q'$, we then have $1 + \frac{1}{2}p\sqrt{n}rw < q/2q'$, where the value $q/q' \geq q_1$, a single factor of $q$, and thus the smallest possible ciphertext modulus. In theory, $q$ could have a factor $q_1$ significantly smaller than the rest, as we could set this as the final ciphertext modulus, which would not be subjected to a modulus switching. We would therefore only require such a factor to be large enough to decrypt ciphertexts that have been subjected to $D$ modulus switchings. A more practical approach however, is to set the following universal bound for any factor of $q$, as the authors of [7] do:

$$q_i > 4p^2r^5w^5n^{1.5}. \tag{7}$$

We may therefore conclude that ensuring any factor of $q$ satisfies bound (7) results in an encryption scheme which reduces the noise sufficiently to guarantee correct decryption of any freshly generated ciphertext and output of ComposedEvalMult, given that the input ciphertexts has at most the same noise level as any freshly generated ciphertexts for the current ciphertext modulus $\bar{q}$. As a result, $q$ should satisfy the following lower bound, as anything else might result in decryption failure:

$$q > (4p^2r^5w^5n^{1.5})^{D+1}. \tag{8}$$

# 5 Subfield Lattice Attack on the NTRU-based Fully Homomorphic Encryption Scheme

## 5.1 Applicability and Success of the Attack

It remains to be shown that the attack of section 3 is at all applicable to the scheme in section 4, and also that it will be successful.

We note first that the authors of [1] state in particular that Heuristic 3.1 holds for the Gaussian distribution, which is the distribution suggested by the authors of [7] for the encryption scheme. It is therefore possible to apply the attack on this scheme.

However, as noted in the last paragraph of subsection 2.3, an attack based on solving SVP or approximate-SVP for the lattice $\mathcal{L}_{\text{NTRU}}$ rests on the assumption that $[f, g]$ is among the shortest vectors in this lattice. We therefore need this assumption to hold, if the attack described in section 3 is to produce a vector which may be used as a secret key. As the following proposition shows, the necessary assumption holds:

**Proposition 5.1.** *With overwhelming probability, the vector $[f, g]$ is one of the shortest vectors in the lattice $\mathcal{L}_{\text{NTRU}}$.*

*Proof.* Recall Theorem 2.4: the length of the shortest vector in any lattice $\mathcal{L}$ is at most $\sqrt{\eta}\det(\mathcal{L})^{1/\eta}$. In the case of $\mathcal{L}_{\text{NTRU}}$, we get $\|\mathbf{v}_0\| \leq \sqrt{2n}(q^n)^{1/2n} = \sqrt{2nq}$.

We may in addition calculate a bound on $\|[f, g]\|$, using the upper bound $\|f\|_\infty, \|g\|_\infty < \sqrt{\frac{q}{4pn}}$, derived in the proof of Proposition 4.1:

$$\|[f, g]\| = \sqrt{f_0^2 + \dots f_{n-1}^2 + g_0^2 + \dots + g_{n-1}^2} \leq \sqrt{2n\left(\sqrt{\frac{q}{4pn}}\right)^2} = \sqrt{\frac{q}{2p}}.$$

Comparing the two bounds, we have: $\sqrt{\frac{q}{2p}} \Big/ \sqrt{2nq} = \sqrt{\frac{1}{4pn}} \ll 1$. Thus, seeing as the bound on $\|[f, g]\|$ is much smaller than the Hermite bound, it is highly probable that $[f, g]$ is one of the shortest vectors in $\mathcal{L}_{\text{NTRU}}$. □

Thus, the attack is applicable to the scheme at hand, and will produce a vector usable as a secret key should it be successful.

With regards to the success of the attack: recall bound (2) of subsection 3.2:

$$\beta^{\Theta(n/\beta\rho)}(n\sigma)^{\Theta(\rho)} \leq q,$$

which needs to be satisfied if the attack is to succeed. This is obviously more likely as $q$ grows in relation to $n$ - in other words, the more factors $q$ consists of, allowing for more CompEvalMult operations to be performed.

If we wish to allow for $D$ modulus switchings to be applied - meaning at most $D$ multiplications are possible without calling on bootstrapping to reduce the noise - $q$ will be of size $(4p^2r^5w^5n^{1.5})^{D+1}$, in accordance with bound (8). The success of the attack therefore depends on whether or not these parameters force $q$ to satisfy bound (2). As the next subsection shows, the attack is successful for an extensive range of parameters, and setting the parameters in such a way that the attack fails results in a rather impractical encryption scheme.

### 5.2 Results

The authors of [1] also carried out experiments to test their attack on actual systems, which is a necessity due to lack of understanding of the nature of the basis reduction algorithms LLL and BKZ. The experimental attacks were carried out on NTRU bases over the ring $R = \mathbb{Z}[x]/(x^n+1)$, for $n$ a power of 2 - meaning the experimental results are transferable to the scheme presented here. We may therefore employ the experimental data given in [1] on the scheme to judge how successful such an attack may be. We set the following values: $p = 2, r = w = 6$, which are the parameter values the authors of [7] suggest.

For example, a successful attack was carried out in 3.5 hours for $n = 2^{11}$ when $\log(q) \geq 165$, which in the given scheme corresponds to $D = 3$, for $q = (4p^2r^5w^5n^{1.5})^{D+1}$. To achieve the same success by running BKZ on the full lattice (that is, not exploiting the possibility of using the sub-field strategy), one would have to run BKZ with block size 27 to achieve $\delta = 1.0141$. For this block-size, BKZ is still considered practical, and the subfield lattice attack might therefore not be too big an improvement in this specific instance [2].

The highest dimension the attack was carried out in was $n = 2^{12}$, with success for $\log(q)$ as low as 190, yet again corresponding to $D = 3$, with the same parameter values as previous. This attack took 120 hours; a direct attack on the full lattice would require running BKZ with block size 131 to achieve $\delta = 1.0081$, an attack that seems unfeasible at this point, as $\beta = 131$ is much too large a block-size to be practical [2].

It follows from these utilizations of the attack that the scheme must be considered insecure if the scheme is also to make meaningful use of the noise reduction strategies presented. Note also that the subfield attacks used LLL to reduce the subfield basis. Therefore it seems reasonable to expect better attacks if BKZ was used on these bases instead, as BKZ consistently outperforms LLL.

## 6 Conclusions

It has been shown that a subfield lattice attack described in [1] may be applied to an NTRU-based fully homomorphic encryption scheme presented in [7]. It has also been shown that the attack requires the integer parameter $q$ of the encryption scheme to satisfy a lower bound in order to be successful. At the same time, utilization of necessary operations that reduce the level of noise in a ciphertext *also* requires $q$ to satisfy a second lower bound, which is typically much larger than the one required for the attack to be applicable. For this to not be the case and the scheme to be safe from the attack, the parameters of the scheme make it very impractical, and essentially unusable, as it would result in a scheme overly dependent on bootstrapping. Thus, we must conclude that the susceptibility of the described attack is inevitable, for all intents and purposes, if the scheme is to make meaningful use of its noise reducing operations.

Further work may include carrying out the subfield lattice attack with BKZ instead of LLL, as this may improve the attack further. It is also possible that this may affect the security of other schemes based on NTRU.

## References

[1] Martin R. Albrecht, Shi Bai, and Léo Ducas. “A Subfield Lattice Attack on Overstretched NTRU Assumptions - Cryptanalysis of Some FHE and Graded Encoding Schemes”. In: *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Confer-*

*ence, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*. 2016, pp. 153–178. URL: http://dx.doi.org/10.1007/978-3-662-53018-4_6.

[2] Yuanmi Chen and Phong Q. Nguyen. "BKZ 2.0: Better Lattice Security Estimates". In: *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*. 2011, pp. 1–20. URL: https://doi.org/10.1007/978-3-642-25385-0_1.

[3] Nicolas Gama and Phong Q. Nguyen. "Predicting Lattice Reduction". In: *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*. 2008, pp. 31–51. URL: http://dx.doi.org/10.1007/978-3-540-78967-3_3.

[4] Craig Gentry. "A fully homomorphic encryption scheme". PhD thesis. 2009. ISBN: 9781109444506. URL: https://search.proquest.com/docview/305003863?accountid=8579.

[5] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. "NTRU: A Ring-Based Public Key Cryptosystem". In: *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*. 1998, pp. 267–288. URL: https://doi.org/10.1007/BFb0054868.

[6] Jeffrey Hoffstein et al. *An introduction to mathematical cryptography*. Second edition. Springer, 2014, pp. 373–454. ISBN: 9781493917105.

[7] Kurt Rohloff and David Bruce Cousins. "A Scalable Implementation of Fully Homomorphic Encryption Built on NTRU". In: *Financial Cryptography and Data Security - FC 2014 Workshops, BITCOIN and WAHC 2014, Christ Church, Barbados, March 7, 2014, Revised Selected Papers*. 2014, pp. 221–234. URL: http://dx.doi.org/10.1007/978-3-662-44774-1_18.

# Improving the generalized correlation attack against stream ciphers by using bit parallelism

Slobodan Petrović

Norwegian University of Science and Technology (NTNU), P.O. box 191, N-2802 Gjøvik, Norway
slobodan.petrovic@ntnu.no

**Abstract**

It is well known that a pseudorandom generator scheme employing irregularly clocked Linear Feedback Shift Registers (LFSRs) can be broken using the generalized correlation attack, in which the constrained edit distance between the intercepted noised output sequence of the generator and the output sequence of a particular independent LFSR inside the generator is computed. The most time-consuming part of the attack is the very constrained edit distance computation, whose time complexity is quadratic in the length of the intercepted sequence. In this paper, we apply constrained bit-parallel approximate search in the attack instead of the edit distance computation. We discuss the advantages and the challenges of using this technique in cryptanalysis.

## 1 Introduction

Pseudorandom keystream generators employing irregularly clocked Linear Feedback Shift Registers (LFSRs) are often used to ensure confidentiality of today's high speed data communication. In these generators, the output sequences of several irregularly clocked LFSRs are usually combined in a non-linear Boolean function. This scheme can be broken using the *generalized correlation attack* [1] in which, to compare binary sequences of different lengths, constrained edit distance is used. The constraints are the maximum length of a run of deletions and the order of particular edit operations (deletions precede substitutions). The bottleneck in this attack is the quadratic time complexity (in the length of the intercepted noised output sequence) of the computation of the constrained edit distance.

In this paper, we show how constrained edit distance computation can be replaced by constrained approximate bit-parallel search (see, for example, [2, 3, 4]). We define the appropriate constraints for the generalized correlation attack and discuss the advantages and challenges of application of this technique in the generalized correlation attack.

## 2 The generalized correlation attack

A pseudorandom sequence generator employing irregularly clocked LFSRs combines their output sequences in a (nonlinear) Boolean function $F$. If the function $F$ has a property that there is correlation between the output sequence and the output sequence of one of the LFSRs generating the input sequences, then it is possible to launch the generalized correlation attack against that LFSR separately. The statistical model of the equivalent generator in that case is shown in Fig. 1.

In this paper, we assume that irregular clocking of $\mathrm{LFSR}_i$ is performed by a single LFSR ($\mathrm{LFSR}_{i0}$). Then $D_n^i$ is a binary sequence, which means that maximum one bit of the output sequence of the clocked LFSR ($\mathrm{LFSR}_i$) can be skipped at a time. We call such irregular clocking 0/1 clocking.

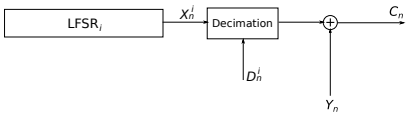



Figure 1: The statistical model used in the generalized correlation attack

The generalized correlation attack against the pseudorandom generator scheme, whose statistical model is presented in Fig. 1 proceeds as follows [1]:

For each non-zero initial state of LFSR$_i$:

1. Generate the output sequence $X_n$ of length $2L$, where $L$ is the length of the intercepted output sequence $C_n$.
2. Compute the constrained edit distance $d$ between $X_n$ and $C_n$.
3. If $d < T$, where $T$ is a threshold chosen in advance, then the initial state of LFSR$_i$ is taken to be a candidate for the true initial state of LFSR$_i$.

Edit distance is defined as the minimum number of elementary edit operations needed to transform one sequence into another. The constraints that are introduced in this definition reflect the way of transforming the output sequence $X_n$ of the clocked LFSR (LFSR$_i$) without irregular clocking into the noised intercepted output sequence $C_n$, starting from any initial state of LFSR$_i$. We assume 0/1 clocking. Then the constraints are:

1. Only deletions and substitutions are used.
2. The maximum number of consecutive deletions is $E = 1$.
3. Deletions occur before substitutions.

To compute the (constrained) edit distance, we usually use the well known dynamic programming method (see, for example, [4]), whose time complexity is quadratic in the length of the intercepted output sequence. To ensure good separation between the values of the constrained edit distance obtained for the candidate initial states of LFSR$_i$ and those obtained for other initial states, the intercepted sequence $C_n$ must be long [1, 5], which implies a very large number of operations needed to compute the constrained edit distance. This may limit the success of the generalized correlation attack.

# 3 Constrained approximate bit-parallel search

Let $S$ be the search string and $w$ the search pattern. The search task is to find all occurrences of $w$ in $S$ (overlap is allowed). If we allow $k$ errors in search then we have the approximate search problem. The errors can occur due to insertions, deletions, or substitutions. Most search algorithms simulate finite state machines of some type. In this paper, we focus on simulating the Nondeterministic Finite Automaton (NFA) assigned to any search pattern. In theory, NFA can make as many transitions as necessary in parallel for every input symbol from the search string $S$. In practice, the search algorithms have to limit the number of transitions performed in parallel at a time. Every time a new symbol from the search string $S$ is processed, the NFA tries to make a transition from the state in which it found itself before that. At the same time, a new copy of this machine is created, being in the state 0 and that new copy also tries to match the current symbol. If in some copy of the NFA a match of a new symbol is not possible such a machine becomes *inactive*. Otherwise, the machine is considered *active*. If any copy of the NFA reaches the final state, an *occurrence* of the search pattern $w$ is reported.

The fact that each new copy of the NFA that is created is active and it starts from the state 0 is usually taken into account by adding a loop in the state 0.

We can represent copying of the NFA at each new input symbol appearance by considering some *states* of the machine active. For this purpose, we introduce the concept of $\varepsilon$-transition, which is a transition that does not consume any input symbol. The NFA starts in the state $I$ and from that state it can reach any state of the NFA that corresponds to the ending symbol of a substring of $w$ (Fig. 2).

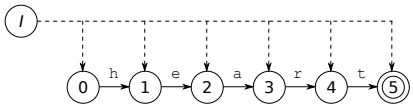



Figure 2: The NFA with $\varepsilon$-transitions

It can be shown [2] that only the status, active or inactive, can be used to completely characterize the machines created in the search process. The status bits can be kept in a single computer word, which is updated for each symbol from the search string. The update is performed in parallel for maximum as many machines as the length of the computer word. This possibility is called *bit parallelism*.

The $\varepsilon$-transitions are used for representing the NFAs used in (constrained) approximate search as well. An NFA of this kind used in our cryptanalytic attack is presented in Fig. 3. A horizontal transition represents a match. A diagonal solid transition represents a substitution. A diagonal dashed transition is an $\varepsilon$-transition that represents a deletion.

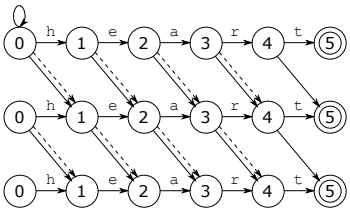



Figure 3: The NFA used in the attack (see text)

The 0-th row of the NFA from Fig. 3 corresponds to an exact match, the 1-st row corresponds to a match with 1 error, etc. The update formula for the automaton is given by the expression (1). The bit mask array $B$ is obtained in advance [2] and depends on the search pattern only.

The NFA from Fig. 3 and the update formula (1) take into account the constraints needed in the proposed cryptanalytic attack. The absence of dashed diagonal transitions in the next-to-last column corresponds to the ordering constraint. The additional bit masks used in (1) in the *del* section correspond to the constraint on the maximum length of a run of deletions.

$$
\begin{aligned}
&dm_i = 1^m, i = 1, \ldots, k \\
&R_0^{'} = (R_0 << 1) \text{ OR } B[S_j] \\
&del = 1^m \\
&R_i^{'} = \text{match AND sub AND del}
\end{aligned}
\qquad
\begin{aligned}
\text{match} &= (R_i << 1) \text{ OR } B[S_j] \\
\text{sub} &= ((R_{i-1} << 1) \text{ OR NOT } B[S_j]) \\
\text{del} &= (R_{i-1} << 1) \text{ OR} \\
&\text{NOT } ((\text{del} << 1) \text{ OR } 0^{m-1}1) \text{ OR} \\
&\text{NOT } ((\text{dm}_{i-1} << 1) \text{ OR } 0^{m-1}1) \text{ OR} \\
&(0^{m-1}1 << (m-1)) \\
&dm_i^{'} = \text{del} \\
&i = 1, \ldots, k
\end{aligned}
\qquad (1)
$$

# 4 Advantages and challenges of using bit parallelism

In the attack, we replace dynamic programming with search. The intercepted noised output sequence of the generator is the search pattern and the output sequence of the attacked LFSR without irregular clocking is the search string. A straightforward advantage of using search instead of dynamic programming is linear time complexity instead of the quadratic one. Another advantage of using constrained approximate search instead of the constrained edit distance computation is in that search implies ignoring the run of deletions before matching the first symbol of the search pattern (due to the existence of the loop in the state 0 of the NFA). Then the search tolerance $k$ can be kept small, which speeds up the computation even more.

The principal challenge when using constrained bit-parallel search is the limited length of the computer word. As commented in Section 2, the intercepted output sequence must be long. Consequently, we have to update our attack infrastructure by concatenating a very large number of computer words, which slows down the attack procedure. Then a trade-off has to be achieved between the length of the intercepted sequence and the time complexity of the attack. A smaller length of the intercepted sequence implies a higher number of false positives. Alternatively, it is possible to use specialized hardware and design a search processor with very long words. In fact, today's FPGA technology allows design of custom CPUs with very long registers at a very low cost. Because of that, this alternative approach is much more promising than using an office computer.

The second challenge is the problem of the reconstruction of the clocking sequence. In the classical attack, this requires backtracking through the dynamic programming array. In the case of using bit-parallelism, backtracking through the NFA has not been studied and some effort is necessary to define the appropriate algorithm.

# 5 Conclusion

In this paper, using the constrained bit-parallel approximate search in the cryptanalysis of pseudorandom generator schemes employing irregular clocking has been proposed instead of computing the constrained edit distance in the dynamic programming manner. The appropriate constraints have been proposed and the corresponding NFA simulation update formula has been presented. Advantages and challenges of using bit-parallelism in such an attack have been discussed. The challenges are possible to overcome by technical means such as using specialized hardware, which makes the proposed attack promising.

# References

[1] J. Golić and M. Mihaljević. A generalized correlation attack on a class of stream ciphers based on the Levenshtein distance. *J. Cryptology*, 3(3):201–212, 1991.

[2] R. Baeza-Yates and G. Gonnet. A new approach to text searching. *Comm. ACM*, 35(10):74–82, 1992.

[3] S. Faro and T. Lecroq. Twenty years of bit-parallelism in string matching. In J. Holub, B. Watson, and J. Žďárek, editors, *Festschrift for Bořivoj Melichar*, pages 72–101, 2012.

[4] G. Navarro and M. Raffinot. *Flexible pattern matching in strings: Practical on-line search algorithms for texts and biological sequences*. Cambridge University Press, 2002.

[5] S. Jiang and G. Gong. On edit distance attack to alternating step generator. In J. No, H. Song, T. Helleseth, and P. Kumar, editors, *Mathematical Properties of Sequences and Other Combinatorial Structures*, pages 85–92. Springer Verlag, 2003.

4

# Distributed Personal Password Repository using Secret Sharing

Merete Løland Elle*, Stig F. Mjølsnes, and Ruxandra F. Olimid

Department of Information Security and Communication Technology, Norwegian University of Science and Technology - NTNU, Trondheim, Norway
{sfm,ruxandra.olimid}@ntnu.no

*The work was mostly performed during M.L.Elle MSc studies at NTNU

**Abstract**

Secret sharing based systems can provide both data secrecy and recoverability simultaneously. This is achieved by a special cryptographical splitting of the data, where the parts, called *shares*, are distributed among a group of entities. A classical solution would be to first encrypt the data (*confidentiality*), then to copy and store the result for backup (*recoverability*). However, by using a secret sharing system, the complete data can be recovered even when only a sufficiently sized subset of shares can be supplied, while any smaller subset of shares does not leak any information about the original data (*perfect secrecy*). For instance, the shares can be distributed across several distinct cloud providers, thus enabling a secure and recoverable storage. Following this idea, we design and propose a novel application for secure and recoverable management of personal passwords by distributing secret shares to cloud storage entities. We have made an experimental smartphone implementation that validates the expediency of the design. The Android application implementation distributes the shares to three cloud providers (Dropbox, Google Drive and Microsoft OneDrive). We note that several mobile password managers exist, but they mostly use the classical solution of encrypted data for storage.

## 1 Introduction

### 1.1 Motivation

Authentication is the most popular method to allow access to resources and services. Despite the tremendous effort to replace the usage of passwords by introducing other mechanisms (e.g.: electronic tokens, one-time keys, biometrics), passwords remain the most common, but also the most vulnerable factor in a personal authentication process. This might be caused by the implementation of the password authentication itself, but very often is caused by bad practices at the user's side. A common method to alleviate people's problem of remembering many passwords is to use the same password for multiple platforms. To prevent people from using this and other bad practices, the very practical question of how to securely manage all our private passwords comes naturally. A solution can be a *password manager*, which should provide at least *confidentiality* of the stored passwords and *recoverability* in case of incidents and faults.

### 1.2 Contribution

The contributions presented in this paper are based on the Android application design and implementation described in Elle's master's thesis work [5], where we investigated the usability of secret sharing for password storage, as an alternative to the classical solution that requires both encryption (for data *confidentiality*) and backup (for data *recoverability*). Unlike this

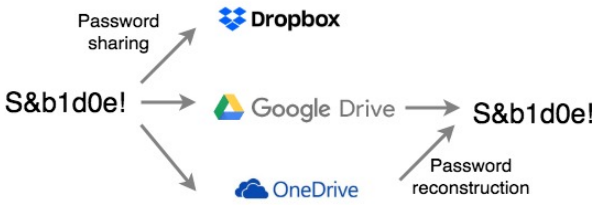



Figure 1: A password is divided in three shares, where each share is stored in a cloud; Reconstruction is possible from any two out of three shares

classical backup method that creates copies, secret sharing does not increase the risk of exposure, but facilitates secrecy by dividing trust between multiple entities. This approach is thus a possible solution for secure distributed cloud storage, where the clouds are located with distinct providers.

Even though the concept of secret sharing has been around for almost 40 years now and has been considered for long-term secure storage, it has been given very little attention for password storage. By using the concept of storing by secret sharing, and implementing it in a mobile application, one could create an application that acts like the conventional password storage manager while providing a solution that is information-theoretically secure. Considering secret sharing as a basic for a distributed personal password repository and implementing a working Android experimental application for sharing passwords among cloud providers brings novelty to this work. This particular application distributes secret shares among three clouds (Dropbox, Google Drive and Microsoft OneDrive) in a theoretically-secure manner. Using Shamir's threshold secret scheme, we obtain recoverability from any two out of three shares. See Figure 1 for a schematic representation of password sharing and reconstruction.

In our literature survey, we found only one other mobile application that uses secret sharing for password management [21]. That Android application is apparently under development, but with no software update since 2016. Moreover, its design does not distribute the shares to third party cloud servers, but to other personal devices, which makes that application's functionality dependent on the availability of the personal devices at any given time.

### 1.3 Outline

The rest of the paper is organized as follows. Next section introduces the theoretical background in terms of secret sharing and the two different approaches for secure storage. Section 3 introduces the implementation background in terms of clouds usage and connectivity, as well as some basic Android implementation notions. Section 4 presents the experimental Android application that implements the distributed password repository among clouds. Last section concludes.

## 2 Theoretical Background

### 2.1 Secret Sharing

Secret sharing is a method to divide a secret into parts, called *shares* that are distributed among a group of entities such that the secret can only be reconstructed when an *authorized*

set of participants combine their shares. Individual shares or combinations of shares belonging to *unauthorized* sets of participants do not reveal the secret. A large variety of secret sharing schemes exist in the literature and are classified based on the secrecy level they guarantee for unauthorized sets, the structure of the authorized sets, the type of the algorithms used for generating the shares or reconstructing the secret, and others.

We will only describe Shamir's scheme [16] here because it satisfies the necessities for our work, and it is further used in the implementation. It is a *threshold* secret sharing scheme, which means that the secret can be reconstructed from any set of at least $k$ shares, where $k$ is a given threshold. This implies redundancy directly: if at least $k$ shares remain accessible, the secret can be reconstructed even if the other shares are lost or destroyed. Moreover, Shamir's scheme is *perfectly secure*, which means that, theoretically, less than $k$ shares reveal absolutely no information about the secret. Shamir's scheme is based on the fact that a $(k-1)$-degree polynomial $f(x)$ is uniquely defined by (at least) $k$ distinct pairs $(x, f(x))$. In the sharing phase, the free coefficient is set to the value of the secret, and the rest of the coefficients of the polynomial are randomly chosen modulo $q$, where $q$ is a large prime number. Notice that the secret itself must lie in $\mathbb{Z}_q = \{0, \ldots, q-1\}$, because it is also acts like a coefficient. This might seem a limitation, but is not the case, because any sequence of bits can be encoded into one or more values in $\mathbb{Z}_q$ that can be shared independently. The $i$-th share is simply set to $f(i)$, the value of the polynomial in $i$. Reconstruction is possible from any $k$ (or more) shares by polynomial interpolation.

### 2.2 Secure Storage Types

The classical solution for secure storage assumes strong encryption, and so decryption becomes *computationally* infeasible in the absence of the decryption key. This means that the data remains confidential to any adversary with bounded computational power that gains access to the ciphertext. Contrastingly, a solution that uses a *perfect* secret sharing scheme (such as Shamir's) maintains confidentiality against any adversary, even with unbounded computational power, under the assumption that the adversary can gain access to up to $k-1$ shares.

The classical solution uses backup or other replication mechanisms to assure data redundancy in case of incidents and faults, while threshold secret sharing based solutions provide data recoverability by construction: the data can be fully reconstructed as long as a minimal number of shares (equal to the threshold) are accessible.

For a more detailed comparison of the two approaches, see [17, 18, 15]. Although many solutions for long-term data storage that use secret sharing mechanisms have been proposed in the literature, surprisingly few investigations have been done to employ secret sharing mechanisms for passwords management.

## 3 Implementation Background

### 3.1 Clouds

Three of the major cloud storage providers on the market today are Dropbox, Google Drive and Microsoft OneDrive [1]. They provide from 2 to 15 GB of free storage. Using an API (Application Programming Interface), an application can interconnect to a cloud by sending and receiving data in a secure manner, using encryption for the data in transit [20]. All the three mentioned clouds use the standard OAuth 2.0 authentication scheme, a framework that

enables applications to obtain limited access to resources, without the users having to share their log-in credentials with the application [11].

In our implementation, the clouds are used to store the shares. The amount of necessarily storage space occupied by the shares is very small compared to the overall free storage space of a user, so there is no negative impact with respect to storage capacity. Storing the shares to the cloud does not introduce any additional costs and in principle, gives the user high flexibility in selecting its own preferable providers. Although the experimental work of the current application allows connectivity to the above-mentioned clouds only, in principle the application can be extended to allow connectivity to other cloud providers as well.

The code used for connecting to the Dropbox API is provided by the Dropbox Core SDK (Software Development Kit) for Java 6+ [3]. Similarly, the code used for connecting to the OneDrive API for Android is provided by the sample code from the OneDrive SDK [13], while the code used for connecting to the Google Drive API for Android is provided by Google [2].

### 3.2 Android

Android is an open-source mobile OS (Operating System) owned by Google [10]. The popularity of Android devices (approximately 69% of the global market in April 2018 [14]) and the ease to program makes it a good choice for developing mobile applications.

The Android Studio is the official Integrated Development Environment (IDE), providing tools for building applications for all Android devices [6].

The security mechanisms for Android includes code signing, application isolation, permission model, file system encryption, and other security protections. The purpose of the code signing is to prove the source of the software, by signing the application with the developer's private key, and only allowing updates to be made if the signature can be verified. Furthermore, each Android application resides in its own security sandbox, which isolates the application data and code execution from other applications [8]. If the application needs to use resources outside of the sandbox, the application has to request permission. The permissions are declared by listing them in the *manifest file* [9].

Note that this work did not go through all steps required for releasing an application because our Android experimental application was considered for testing purposes in a limited environment. The application was made available for download outside the Play Store. The manifest file includes access to the Internet, the network state, reading and writing to the external storage, and managing access to documents. It also contains additional metadata, such as the API key for the Google API and the authentication activity for Dropbox with the application key.

## 4 The Application

The distributed password repository is implemented in the shape of an Android application that uses Shamir's secret sharing scheme with threshold $k = 2$. The application divides each password into three shares and distributes each to three different cloud providers (Dropbox, Google Drive and Microsoft OneDrive). Android Studio has been used in the developing process, and the connectivity to the clouds was performed as previously explained.

### 4.1 Functionality

We describe next the basic functionality of the experimental implementation in terms of user actions and results. We keep in mind that our fundamental goal is not to describe the imple-

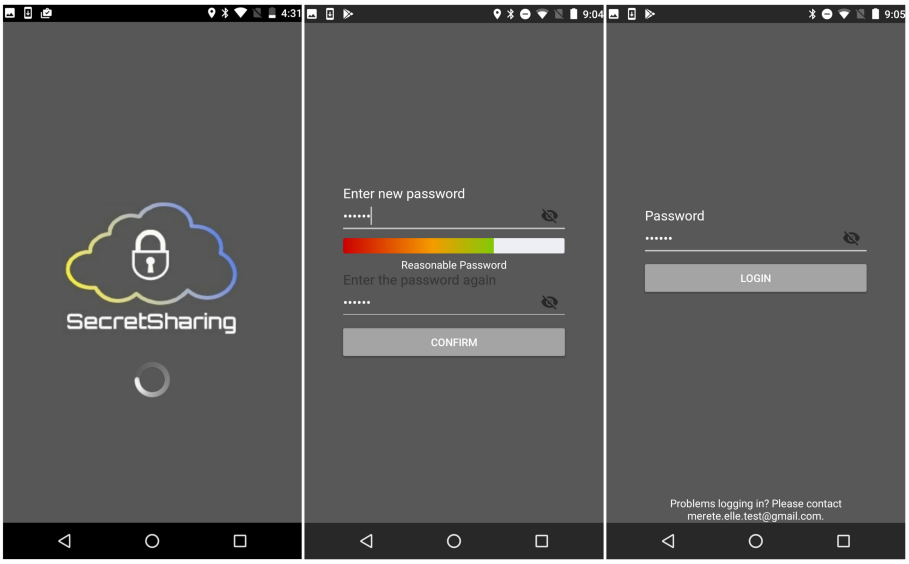



Figure 2: Screenshot of the application; left: welcome screen; middle: creating a new application password; right: log-in screen

mentation itself, but to consider secret sharing as a basis for a distributed personal password repository, so we also mention capabilities that have not been implemented in the application, improvements and future work.

**Login.** After the welcome screen (or *splash screen*), the user is either asked to introduce a password (if no application password has been set before), or to log-in by typing the correct password. When a new password is introduced, a progress bar gives an indication of its strength. The evaluation is based on the length of the password, as well as the number of symbols, digits, uppercase and lowercase letters. Afterwards, the login screen is prompted every time the application is launched, to create an extra barrier for a potential attacker. This would disallow for example direct access to the application to someone that has physical access to the phone. Figure 2 illustrates related screenshots of the application.

**Connectivity to the clouds.** The entry point of the application is the connection status screen, displayed after the user is successfully logged in. Is here where the user has an overview of whether it is connected or disconnected to the clouds, and also if there is Internet connectivity. For each cloud there is displayed a login button. The login buttons contain the names and icons for the clouds, in accordance with the guidelines for branding [4, 12, 7]. If the user is not logged to any cloud, then all of the login buttons are visible and all the clouds are marked with a red cross. When the user logs to any of them (using his personal credentials), the red cross changes to a green tick symbol, indicating that the application is connected to the cloud. If so, the login button becomes invisible. Figure 3 illustrates related screenshots of the application.

The elements in the drop-down menu change depending on how many clouds are connected. If none, the user only has access to change the password for the application. If at least one cloud is connected, the user can in addition log out from the clouds. If at least two clouds are

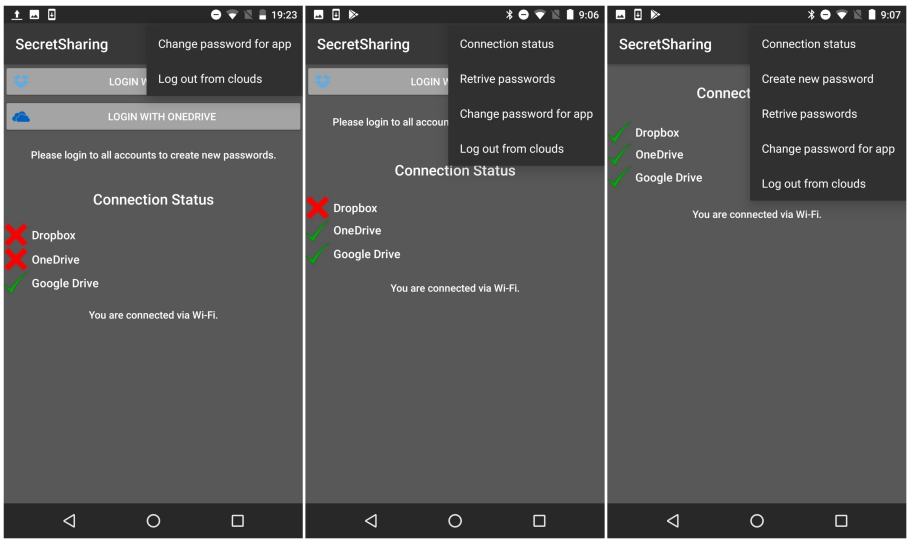



Figure 3: Screenshot of the application; left: drop-down menu when one cloud is connected, middle: drop-down menu when two clouds are connected, right: drop-down menu when all clouds are connected

connected, the user can reconstruct passwords. Only when all clouds are connected, the user has full functionality and can store a new password. The restriction of storing a password only when all clouds are up is to achieve redundancy: a password can be recovered from any two shares, but the splitting is done in three shares. Of course, this is a particular implementation, which is somehow hardcoded. To properly allow flexibility, the user might choose the overall number of shares and the threshold needed for reconstruction. But the implementation should be done with care, do disallow the user to poorly parametrize (e.g.: to get no redundancy).

**Store a password.** To store a new password, the user needs to press the *Create new password* button in the drop-down menu. This action redirects to a new screen, as shown in Figure 4. Here, the *platform name* field corresponds to the system the password is used for, e.g. "Facebook", and the *password* field is the password to be stored. When the save button is pressed, a pop-up is prompted to confirm the action. The application will split the password into shares and distribute to each cloud a share titled with the platform name, which can be later used to reconstruct the password. Again, an implementation decision has been made here not to hide the name of the platform, assuming that only the password protection is important. Of course, to some extend the platforms themselves can be considered private, as they disclose something about the user behavior, so the name of the shares should be independent of the platform name. This can easily be achieved by keeping a file that matches the names of the platforms to random or generic sequences. These sequences are then used for naming the shares in the clouds, and the matching file can be stored split between the clouds by secret sharing. Each reconstruction would then additionally imply first the reconstruction of the matching file, and only after the reconstruction of the password by the corresponding shares. In the absence of this matching file, a similar file containing just the names of the platforms can be stored in the clouds in case of the phone is stolen, lost or dysfunctional and the user needs to reinstall the application on

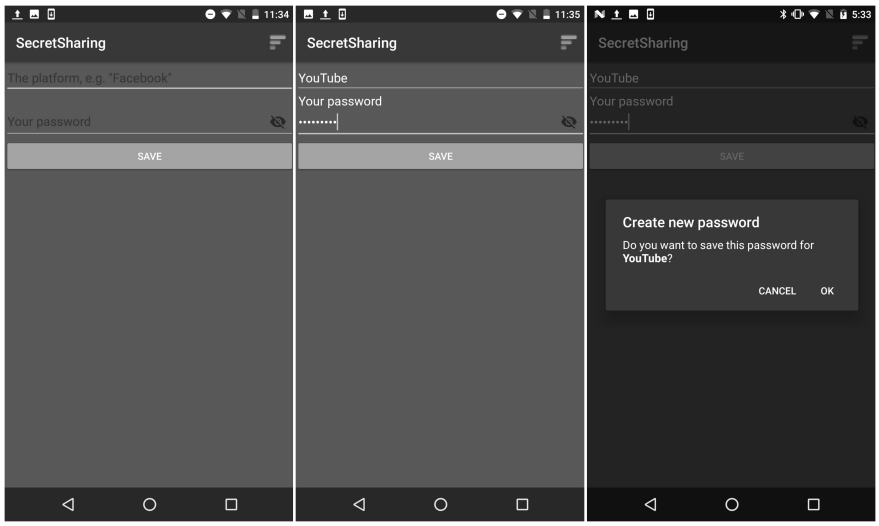



Figure 4: Screenshot of the application; left: create new password, middle: filled in platform name and password, right: pop-up for confirmation

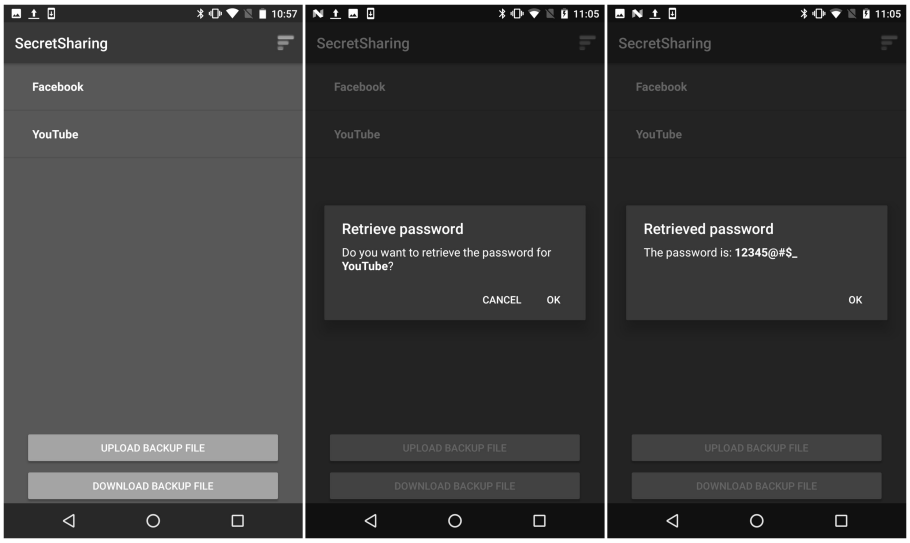



Figure 5: Screenshot of the application; left: retrieve passwords, middle: pop-up asking if the user wants to retrieve the chosen password, left: the password retrieved

another device and restore the passwords. This is what we call the *backup file*.

**Reconstruct a password.** To reconstruct a password, the user needs to press the *Retrieve passwords* button in the drop-down menu. This action redirects to a new screen that contains a list of the stored passwords, as shown in Figure 5. The user selects the platform's name that

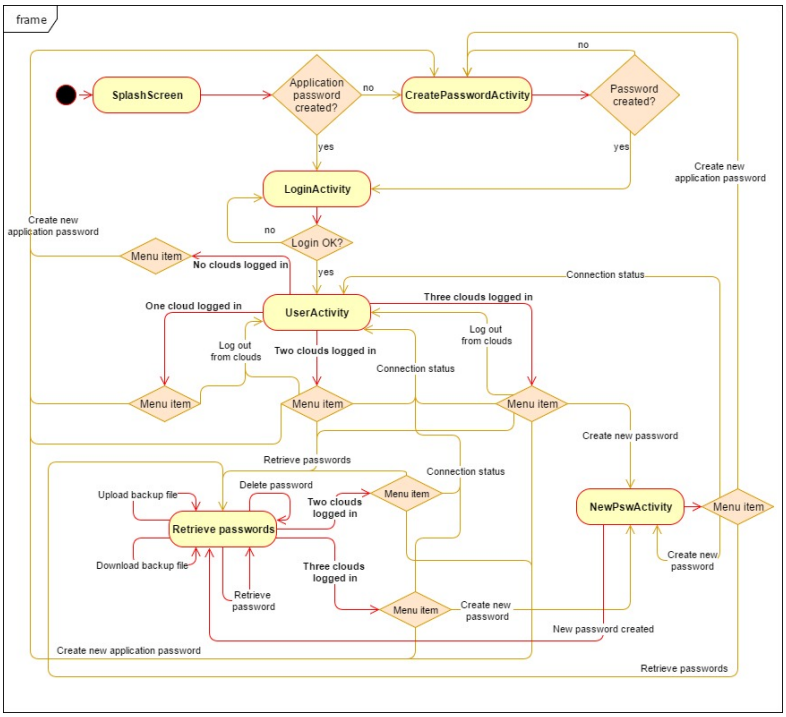



Figure 6: An overall representation of the activities and menu options shown to the user

corresponds to the password he wants to recover and confirms in the pop-up window. This triggers the reconstruction of the password from shares, using the secret sharing scheme.

**Deleting a password.** To delete a password, the user needs to click on the platform name and hold it for a short while before a pop-up with a warning message is displayed. After the user confirms deletion, the application triggers the deletion of all the shares belonging to the given password in the clouds.

Figure 6 is an overall representation of the activities and the options shown to the user. On the Android platform, an activity serves as the entry point for an applications interaction with the user. The implementation contains six activities visible to the user, which are illustrated in the figure.

**Completing the design.** One could immediately observe the absence of the usernames here. Even if usually the username is easier to remember (e.g.: the e-mail address), the user should not be asked to remember them, but they must be managed by the password manager application too. This could be simply solved by concatenating the username to the password (for instance delimited by a space) before sharing. This is if the application prototype implementation is to be followed. A simpler approach would be to store all platforms' names, together with their associated usernames and passwords, in a single file that is to be shared between the clouds. This would imply the storage of a single share (of larger size) to each cloud. Each time a new password needs to be stored, the whole file is reconstructed, the new string *(platform-name,*

*username, password)* is appended to the file and the new version is secret shared, updating the new shares to the clouds. A similar procedure should be followed in case of deletion of passwords. Of course, this solution would require a higher computational cost for a single password storage or deletion, but it should not introduce a significant overload, so it should not affect the usability of the application. It would also eliminate the need of a backup or matching file, and it would keep private the platforms' names by design. For any of the solutions, a versioning system should be put in place, and timestamp of all performed actions must be available.

## 4.2 Sharing and Reconstruction

An available java implementation was used as a base for the secret sharing algorithm [19].

A 384-bit prime $q$ is used for the split and reconstruction operations, and the password is encoded to an integer in $\mathbb{Z}_q$. As already mentioned, this is not a limitation because the password string can be broken into smaller pieces that lie in $\mathbb{Z}_q$. Anyway, this should not be the case, because no usable password can exceed 48 characters in length.

**Share a password.** To split a password into shares, a method called *splitSecretIntoPieces* is called with parameters the password *secret*, the total number of secrets $n$ and the number of shares required for reconstruction $k$. The return value is a string array containing the shares, that will later be distributed to the clouds. Listing 1 shows the splitting of the password into shares.

Listing 1: Splitting password into shares, code from *UploadSharesTask.java* file

```
// Split password into shares.
    final String secret;
    secret = [THE_PASSWORD];
    final int n = 3, k = 2;
    String[] pieces = splitSecretIntoPieces(secret,n,k);
```

**Reconstruct a password.** To reconstruct the password, a method *mergePiecesIntoSecret* is called with parameter a string array obtained from the shares used for reconstruction. As an implementation example, Listing 2 shows the scenario where the password is reconstructed when all three shares are used.

Listing 2: Retrieving password, code from *RetrieveActivity.java* file

```
String[] pieces = new String[3];
pieces[0] = [OneDrive_SHARE];
pieces[1] = [Dropbox_SHARE];
pieces[2] = [Google Drive_SHARE];

// Create arraylist out of the formatted strings,
// shuffle and reconstruct secret.
List<String> list = new ArrayList<String>(Arrays.asList(pieces));
Collections.shuffle(list);
String[] kPieces = list.toArray(new String[0]);
final String reconstructed = mergePiecesIntoSecret(kPieces);
```


Figure 7: The overall score of the user tests: "PASSED" 98,2% and "FAILED" 1,8%

### 4.3 Testing and Feedback

An evaluation was conducted to measure how well the application performed while being used by regular users. The users were able, and encouraged, to comment on the application's interface and behavior. The aim was to have a focus group of five subjects, preferably with various degrees of technical skills.

The subjects were given a test sheet containing 70 steps that covered all aspects of the functionality, including for example logging in, storing and recovering passwords. The users gave each functionality a score of "PASSED" or "FAILED", where "PASSED" means that the application responded as described in the test sheet, and "FAILED" means that there were some deviations from the description in the test sheet. By evaluating these tests, a quantitative score, the *overall score*, was used to describe how well the application performs. Figure 7 shows the overall score of the user test, with a "PASSED" score of 98,2%. The comments provided by the users served as inputs for further development and improvement of the application.

The sharing and reconstruction of the secret is performed fast and hence introduces no disturbing delay for the user. Therefore, from a performance point of view there was no negative feedback with respect to the response time of the application.

## 5 Conclusions

The paper presents a distributed personal password repository prototype that uses secret sharing implemented in the form of an Android experimental application. The usage of secret sharing as an alternative solution for password storage brings novelty. By using Shamir's threshold secret sharing, the application provides redundancy by default. In the presented implementation, two out of three shares are required for reconstruction, but a more flexible approach might allow the user to parametrize the overall number of shares, as well as the shares required for reconstruction. Also, flexibility should be added to allow connectivity to other clouds. In a complete version, usernames must be stored for each platform together with the passwords, and a versioning system could be set into place.

As a risk, there is a certain chance that the shares can be deleted outside the application, directly from the cloud. If this is the case, and the remaining shares are below the reconstruction threshold, then reconstruction is not possible and an error message is displayed.

A testing of the functionalities of the implementation was performed. The goal was not to test the security of the application, but mostly its functionalities and usability. Therefore, future work includes an in-depth security evaluation of the implementation. Furthermore, enhancements on the privacy should be considered. For example, this includes the secrecy of the names of the platforms the user has accounts for, which to some extent represent private information too.

# References

[1] Martyn Casserly. The best cloud storage services. http://www.pcadvisor.co.uk/test-centre/internet/best-cloud-storage-services-2017-uk-3614269/. Accessed: April 2018.

[2] Dropbox. Authorizing Android Apps. https://developers.google.com/drive/android/auth, 2017. Accessed: April 2018.

[3] Dropbox. Dropbox Core SDK for Java 6+. https://github.com/dropbox/dropbox-sdk-java, 2017. Accessed: April 2018.

[4] Dropbox, Inc. Developer branding guide. https://www.dropbox.com/developers/reference/branding-guide. Accessed: April 2018.

[5] Merete Løland Elle. Long-term confidential data storage by distributed secret shares. Master's thesis, NTNU, Department of Information Security and Communication Technology, 7 2017. https://brage.bibsys.no/xmlui/handle/11250/2458158.

[6] Google. Android Studio. https://developer.android.com/studio/index.html. Accessed: April 2018.

[7] Google. Use the Drive Badge and Brand. https://developers.google.com/drive/v3/web/branding. Accessed: April 2018.

[8] Google. Application Fundamentals. https://developer.android.com/guide/components/fundamentals.html, 2017. Accessed: April 2018.

[9] Google. Declaring Permissions. https://developer.android.com/training/permissions/declaring.html, 2017. Accessed: April 2018.

[10] Google. Understanding Android. https://www.android.com/everyone/facts/, 2017. Accessed: April 2018.

[11] Internet Engineering Task Force (IETF). [RFC6749] The OAuth 2.0 Authorization Framework. https://tools.ietf.org/html/rfc6749, 2012. Accessed: April 2018.

[12] Microsoft. Branding guidelines. https://msdn.microsoft.com/en-us/onedrive/dn673556.aspx. Accessed: April 2018.

[13] Microsoft. OneDrive SDK for Android. https://github.com/OneDrive/onedrive-sdk-android, 2017. Accessed: April 2018.

[14] Net MarketShare. Operating System Market Share. https://www.netmarketshare.com/operating-system-marketshare.aspx?qprid=8&qpcustomd=1., 2018. Accessed: April 2018.

[15] Ruxandra F. Olimid and Dragos Alin Rotaru. On the security of a backup technique for database systems based on threshold sharing. *Journal of Control Engineering and Applied Informatics*, 18(2):37–47, 2016.

[16] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[17] Mark W. Storer, Kevin M. Greenan, Ethan L. Miller, and Kaladhar Voruganti. Potshards - a secure, recoverable, long-term archival storage system. *ACM Transactions on Storage (TOS)*, 5(2):5, 2009.

[18] Arun Subbiah and Douglas M. Blough. An approach for fault tolerant and secure data storage in collaborative work environments. In *Proceedings of the 2005 ACM workshop on Storage security and survivability*, pages 84–93. ACM, 2005.

[19] Tim Tiemens. Shamir's Secret Share in Java. https://github.com/timtiemens/secretshare, 2014. Accessed: April 2018.

[20] Davey Winder. How secure are Dropbox, Microsoft OneDrive, Google Drive and Apple iCloud cloud storage services? http://www.alphr.com/apple/1000326/how-secure-are-dropbox-microsoft-onedrive-google-\drive-and-apple-icloud-cloud-storage. Accessed: April 2018.

[21] ETH Zurich. Convenient Password Manager. https://play.google.com/store/apps/details?id=disco.ethz.ch.convenientpasswordmanager&hl=no, 2016. Accessed: April 2018.

12

# The tension between anonymity and privacy

Staal A. Vinterbo

Department of Information Security and Communication Technology
Norwegian University of Science and Technology
Staal.Vinterbo@ntnu.no

**Abstract**

Privacy in the context of information and data is often defined in terms of anonymity, particularly in regulations such as the GDPR. Operationally, it is appealing to define privacy in terms of computable data properties as this makes it possible to verify compliance. A well known example of privacy defined as such is $k$-anonymity. At the same time, uncertainty regarding real-world privacy is increasing with the amount of data collected about us all. We present arguments for why focusing on anonymity or computable properties of data is not very helpful in this regard. In particular, we count exploitable failures of privacy defined in terms of computable properties of $n$-bit data and conclude that these counterexamples to protection cannot be rare.

## 1 Introduction

Many privacy regulations, including the General Data Protection Regulation (GDPR) [1] and the US Health Insurance Portability and Accountability Act (HIPAA) [18], have anonymity as a decision criterion of whether they apply to the contents of dataset or not.

Now, consider a colleague showing you data and asking "Is this dataset anonymous?", effectively asking you if it can be shared without running afoul of privacy regulations. Ethics and potential personal harm aside, getting the answer wrong can have financial and legal consequences. Especially as privacy regulations grow teeth, as they are doing in Europe, where the upcoming GDPR threatens with significant penalties. Unfortunately, relying on a positive answer to this question is problematic.

Uncertainty about anonymization and privacy features prominently among barriers to efficient use of information collected from individuals [16, 17]. A perhaps non-obvious reason is that anonymity and anonymization strongly suggests a focus on prohibiting a one-to-one relation from data to identity, while actually providing what most of us associate with privacy requires prohibiting more general inferences. Intuitively, instead of asking "can I figure out who this data comes from" we have to ask "what new inferences about anyone can I make using this data". As we will see, addressing the latter is difficult and puts additional constraints on how information can be shared. In particular, collecting and sharing anonymized data becomes difficult.

Our goal here is to substantiate the above with simple yet somewhat formal arguments. We also briefly present how differential privacy [9], an emerging standard for data privacy, relates to the identified challenges. From a technical perspective, our main contribution is a quantitative argument that failures of checkable privacy cannot be rare (Section 5.1 and Theorem 5.4).

## 2 Why anonymity

The distinction between the public and private was understood in Greece and China 400 BC [15]. In the early Renaissance in Europe, the notion of the home as a protected and sovereign sphere was documented already in 1499 [21]. This idea is also found in the Fourth Amendment of the US

Constitution, with a focus on protections from the government. In 1890, Warren and Brandeis published the very influential "The right to privacy", where they reacted to the newspapers' overstepping bounds of propriety and decency, particularly with photographs becoming available. Warren and Brandeis declared a "right to be left alone", laying the foundation of what Ruth Gavison much later defends as the right to hide in the crowd [13].

This view of privacy as anonymity has been widely adopted in regulations. For example, the upcoming (May 2018) GDPR explicitly states:

> "The principles of data protection should therefore not apply [...] to personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable."

From a more technical point of view, anonymity is violated by relating a piece of information or trait to a single identity. A way of preventing this is introducing ambiguity. For data about people this can be stated as:

> if enough people share a trait, then it is not identifying.

The beauty of this idea is that we can use data we have to establish a lower bound on how many people share a trait, or several traits. Sweeney's well known $k$-anonymity [22] parameterizes sufficient ambiguity for anonymity as $k$ people having to share all present combinations of values for traits. In particular, introducing ambiguity destroys 1-to-1 relations. This was the explicit motivation behind the definition of $k$-anonymity [23].

As an example, consider the data in Table 1 where the possession and absence of two traits $a$ and $b$ for five people are indicated by 1 and 0, respectively.

**Table 1:** Two traits and five people

| trait | Alice | Bob | Graham | Denise | Eric |
|---|---|---|---|---|---|
| $a$ | 1 | 1 | 0 | 0 | 0 |
| $b$ | 1 | 1 | 1 | 0 | 0 |

We see that for the individual traits, at least two people share both possession and absence. In terms of $k$-anonymity, the data is 2-anonymous in $a$ and $b$ individually. We also see that only Graham exhibits absence of $a$ and possession of $b$. Consequently, his pattern of absence and possession is not "anonymous". If we were to remove Graham from the data, it would be 2-anonymous (in $a$ and $b$ jointly).

In general, definitions of privacy that we can check are called *syntactic*. From the perspective of sharing data, being able to check privacy compliance avoids the burden of having to document data provenance, which in turn could be sensitive.

In summary, anonymity represents a tradition in jurisprudence that is easy to operationalize by syntactical means.

## 3 The insufficiency of prohibiting 1-to-1 relations

Consider the following story of Alice and Bob. Alice works at the local hospital as an analyst. Lately, she has been working with researchers investigating the connection between HIV and diabetes. Specifically, her work has been to answer the question whether the hospital already has

sufficient patient data for a study, and whether recruiting external prospective study participants is necessary. To accomplish this, she has been given access to counts of how many patients in the hospital database have been diagnosed as diabetic, HIV positive and both simultaneously, across gender and age. Last Saturday, she was invited to her neighbor Bob's 40th birthday BBQ. During their chat while Bob was flipping burgers, Bob mentioned that he loves Coca Cola, but that his doctor at the local hospital had told him that he has to be more careful about controlling his diabetes. Prompted by this comment, Alice, more of a beer drinker herself, recalls that there were no male, age 35-40, diabetic, and HIV negative patients in the hospital database a month ago.

The information Alice obtained through hospital data access can be described as the implication $(a \implies b)$, where $a$ is true for males, age 35-40, that are diabetic, and $b$ is being HIV positive. A logically equivalent formulation of the implication is $\neg(a \wedge \neg b)$. This means that if $(a \implies b)$ is true for all elements in the database (note that this is different from $a$ being true for all), then $(a \wedge \neg b)$ is true for none. Neither allows establishing a unique relationship with anyone in the database. Note that $(a \implies b)$ holds for all people in Table 1.

It is only through Bob telling Alice that he is in the database and that $a$ is true for him that Alice is able to infer that Bob is HIV positive. Again, neither of the pieces of information Bob provides represent a one-to-one relationship.

In our above example, $(a \wedge \neg b)$ was true for none. Can we consider the information that something does not exist (in the database), personal information or data? This question is interesting in terms of language semantics, as the GDPR emphasizes that personal data is data related to a real living person.

# 4 Privacy as protection against adversarial inference

As we have seen above, ambiguity as a privacy criterion is not sufficient. What we arguably care about is any inference about any individual. Paraphrasing Dalenius [4], disclosure by information $v$ happens if we can use it to learn something about $x$ we did not already know. We can define this notion a little more formally using probabilities as follows.

**Definition 4.1** (Disclosure)**.** Let $M_r$ be a randomized inference "machine" for a property $r$. Disclosure of $r$ for object $x$ by information $v$ happens if

$$P[M_r(x, v) = r(x)] > P[M_r(x) = r(x)].$$

Furthermore, we can think of disclosure as *direct* if $r(x)$ can be determined from $v$ alone, or *indirect* if more information and inferential machinery is needed.

We can now cast anonymity of data in terms of disclosure as follows. Let property $r$ be identity, i.e., $r_{id} = x \mapsto \text{Identity}$. Then we can say that data $v$ is anonymous if

$$(\forall x)\ P[M_{r_{id}}(x, v) = r_{id}(x)] \leq P[M_{r_{id}}(x) = r_{id}(x)].$$

Generalizing a little, we can think of $r$ as being any sensitive information. Of course, what sensitive information means is subjective. For example, not all HIV positive patients consider their status as sensitive in all contexts [20].

Unfortunately, as Dalenius informally, and Dwork later formally argues [6], eliminating disclosure while providing useful information is impossible. Intuitively, no information computed from data about individuals can be independent of this data and reflect the contents of the data at the same time. Consequently, we can think of privacy as *controlling* disclosure, or

alternatively, controlling how much a particular piece of information aids inferences about someone. Abandoning the prohibition of disclosure also signals a departure from thinking of privacy as controlling access to crisply circumscribed information, a goal of information security.

The notion of indirect disclosure invites the question of what resources an adversary has at her disposal. Knowing what information an adversary can use for inference generally requires omniscience. In order to avoid uncertainty due to assumptions, we have to make the strongest assumption possible. This assumption is that the adversary has enough information to reduce her task to a choice between two alternatives. Importantly, we do not know which alternatives those are.

Not knowing which pair of alternatives the adversary is left to decide between makes it impossible to distinguish between sensitive and non-sensitive properties. We cannot dismiss the possibility that a non-sensitive property allows us to rule out one of the two remaining alternatives. An instance of this problem is deciding which attributes in a data table do not help the adversary when linking to other tables.

One could ask whether such a strong adversary is esoteric enough to not matter in practice. Again, this is hard to know as an adversary will manipulate the context to her advantage. One way of doing this is in terms of an "intersection attack" where a set of a priori known candidate hypotheses is intersected with the hypotheses corresponding to a given response [12]. Furthermore, the specter of Russian manipulation of the 2016 US election could be taken as a cautionary tale against underestimating the resources of an adversary. In particular, this emphasizes the importance of reducing the reliance on assumptions.

# 5 Syntactic privacy

Since only considering one-to-one relationships such as identity is insufficient, can we abandon the anonymity tradition but somehow keep the operational advantage of checkable privacy? Unfortunately, barring a definition where essentially all data is sensitive, we can always find a formal example of privacy breach for any syntactic definition of privacy. Moreover, as we will see, such counterexamples cannot be rare.

We will consider databases and encodings of information somewhat interchangeably as finite bit-strings, i.e., elements from the set $\{0,1\}^*$.

**Definition 5.1** (Syntactic definition of privacy)**.** A function $\sigma : \{0,1\}^* \to \{0,1\}$ that returns 1 if its input is sensitive and 0 otherwise is a *syntactic definition of privacy.*

We will assume that all our syntactic privacy definitions are computable. Importantly, computable $\sigma$ means that syntactic privacy captures the notion of checkable privacy of data. From a disclosure control standpoint, syntactic privacy allows for deciding whether a given statistic (or data) causes disclosures with some likelihood.

Anna has a database that contains sensitive information and wants to answer Ben's query, but without sharing sensitive information. A *sanitizer* is a mechanism by which she can extract the queried information from the data in a safe manner.

**Definition 5.2** (Deterministic sanitizer)**.** Given a non-constant syntactic definition of privacy $\sigma$, an algorithm that computes function $f : \{0,1\}^* \to \{0,1\}^*$ is a *deterministic sanitizer* for $\sigma$ if $\sigma(f(x)) = 0$ for all $x \in \{0,1\}^*$.

We will think of a sanitizer as a response algorithm for a query $q$ on a bit-string database. We generally want the answer that is most useful for that query, i.e., that the function $f$

computed by the sanitizer approximates the function $q$ well. This is difficult in general. For example, finding the least generalized $k$-anonymous database is NP-hard and it is known that an approximation that is not worse than $O(k \log k)$ times than needed can be computed, however with a running time exponential in $k$ [14].

Now consider the situation where Anna has a sensitive database $s0$ where $s$ is a bit string of length $n-1$. Also, let Ben know $s$, meaning he knows all bits of Anna's database except the last one. For sanitizer $f$ to be safe, Anna should be able to give Ben $f(s0)$ without divulging the value of the last bit. If $f$ is deterministic and $f(s0) \neq f(s1)$, then Ben can infer what Anna's database is by checking which of $s0$ and $s1$ yield the value $f(s0)$ he receives from Anna. This leads us to the following definition.

**Definition 5.3** (Counterexample)**.** Let $f$ be a deterministic sanitizer for definition of privacy $\sigma$ on $n$-bit databases. Any pair of databases $x$ and $y$ differing in one bit such that $\sigma(x)+\sigma(y) > 0$ and $f(x) \neq f(y)$ constitutes a *counterexample* of $f$. Furthermore, if $(x, y)$ is a counterexample for any sanitizer for $\sigma$, $(x, y)$ is a counter-example of $\sigma$.

In terms of our discussion of adversaries in Section 4, a counterexample consists of two specific alternative hypotheses that an adversary can reduce to a single correct hypothesis.

Unless otherwise indicated, we consider sanitizers to be deterministic. The reason for this is that if we rely on non-determinism or randomness for privacy, we go beyond what can be checked[1]. We now describe the syntactic definitions of privacy that allow counterexamples.

**Definition 5.4** (Useful syntactic privacy)**.** A non-constant syntactic definition of privacy is *useful* if there are at least two non-sensitive databases.

The reason for Definition 5.4 is that a definition that is not useful, only allows sanitizers that are constant (trivial), and therefore useless.

Variations of the Anna and Ben example above are common in presentations on theory about disclosure control and differential privacy. The following theorem is a formalization of the idea behind these examples, and can therefore be considered a "folklore" theorem.

**Theorem 5.1** (Folklore)**.** *For any useful syntactic definition $\sigma$ of privacy on $n$-bit databases, there exists a counterexample.*

*Proof.* There exists a pair $(x, y)$ of $n$-bit databases for which $\sigma$ yields different values since $\sigma$ is non-trivial. We can create a sequence $(x_0, x_1, \ldots, x_k)$ such that $k \leq n$, $x_0 = x$, $x_k = y$, and $x_{i+1}$ equals $x_i$ with a single bit inverted. Since $\sigma(x_0) \neq \sigma(x_k)$, there must exist $i$ such that $\sigma(x_i) \neq \sigma(x_{i+1})$. Since $\sigma$ is useful, there exists non-sensitive distinct databases $u$ and $v$. Let sanitizer $f$ be such that $u = f(x_i) \neq f(x_{i+1}) = v$. The pair $(x_i, x_{i+1})$ is therefore a counterexample. □

## 5.1 Counting counterexamples

Being completely safe is trivial: respond to queries using a constant sanitizer. This is generally not a helpful observation since sharing information is the the reason we are interested in sanitizers in the first place. Providing utility implies the ability to distinguish between databases. But, from the proof of Theorem 5.1, being able to discriminate between neighboring databases can yield counterexamples. A natural question now is whether we can find a suitable syntactic definition of privacy that allows answering many questions but only has few counterexamples. If

[1] more precisely, a given finite string cannot be proven random [3].

such a definition exists, then we could argue that since the number of counterexamples is low, Theorem 5.1 does not matter in practice. We approach this question by showing that only a negligible fraction of syntactic definitions on $n$-bit databases do *not* exhibit a full complement of counterexamples.

### 5.1.1 Syntactic privacy on the $n$-cube

For each $n$-bit database $x \in \{0,1\}^n$ there are $n$ other $n$-bit databases that differ from $x$ in a single bit. We denote that two databases $x, y \in \{0,1\}^n$ differ in a single bit by $x \sim y$, and call them neighbors. If we take the set of all $n$ bit databases and connect all the neighboring $n$ bit databases by an edge, we get a hypercube graph, or $n$-cube.

Define the weight $w$ of a database $x \in \{0,1\}^n$ to be the number of 1 bits in it, i.e., $w(b_1, b_2, \ldots, b_n) = \sum_{i=1}^{n} b_i$. Now define the balance of $x$ as $\beta(x) = (-1)^{w(x)}$. The balance of $x$ tells us whether the weight of $x$ is odd or even with values -1 and 1, respectively.

**Proposition 5.2.** *For any $S \subset \{0,1\}^n$ such that $\exists\ u, v \notin S\ \beta(u) \neq \beta(v)$, there exists $f : \{0,1\}^n \to \{0,1\}^n$ such that*

$$\forall x \in S, \forall y \in \{0,1\}^n (x \sim y \implies f(x) \neq f(y)).$$

*Proof.* By assumption we can fix $u, v \in \{0,1\}^n$ such that $\beta(u) \neq \beta(v)$. Now define $f : \{0,1\}^n \to \{0,1\}^n$ as

$$f(x) = \begin{cases} u & \text{if } \beta(x) = \beta(u) \\ v & \text{otherwise,} \end{cases}$$

For all $x, y \in \{0,1\}^n$ we have

a. $x \sim y \implies \beta(x) \neq \beta(y)$
   since $x \sim y \implies |w(x) - w(y)| = 1$.
b. $\beta(x) = \beta(f(x))$
   by definition of $f$.
c. $\beta(x) \neq \beta(y) \implies x \neq y$
   since $\beta(x) \neq \beta(y) \implies w(x) \neq w(y)$ and $w(x) \neq w(y) \implies x \neq y$.

Combining a., b., and c., we get

$$x \sim y \overset{a.}{\implies} \beta(x) \neq \beta(y) \overset{b.}{\implies} \beta(f(x)) \neq \beta(f(y)) \overset{c.}{\implies} f(x) \neq f(y). \qquad \square$$

Recall that we have $m = 2^n$ different $n$-bit databases. There are $2^m$ different subsets $S$ of databases, and one of them is empty, and one of them is $\{0,1\}^n$. Consequently, there are $2^m - 2$ non-empty proper subsets of $\{0,1\}^n$. We now turn to how many of these do not fit the requirement for application of Proposition 5.2.

**Proposition 5.3.** *Let $m = 2^n$. Then there are $2^{\frac{m}{2}+1} - 2$ non-empty sets $T \subseteq \{0,1\}^n$ such that $x, y \in T \implies \beta(x) = \beta(y)$.*

*Proof.* Let $U_j = \{x \in \{0,1\}^n | \beta(x) = j\}$. Then

a. $U_1 \cup U_{-1} = \{0,1\}^n$
b. $U_1 \cap U_{-1} = \emptyset$

c. $|U_i| = 2^{\frac{m}{2}}$ for $i \in \{-1, 1\}$

Let $T$ be non-empty such that $x, y \in T \implies \beta(x) = \beta(y)$. Since this means that all elements in $T$ must have the same balance, we have that $T \subset U_i$, where $i$ is this shared balance. The converse is also trivially true. This means that $T$ can be any non-empty subset of either $U_{-1}$ or $U_1$. From c. we have that $U_{-1}$ and $U_1$ each have $2^{\frac{m}{2}}$ subsets, out of which one is empty. Consequently, there are $2(2^{\frac{m}{2}} - 1) = 2^{\frac{m}{2}+1} - 2$ possible non-empty sets $T \subseteq \{0,1\}^n$ such that $x, y \in T \implies \beta(x) = \beta(y)$. □

Since each syntactic privacy definition $\sigma$ is uniquely defined by its set $S$ of sensitive databases, we have that there are $2^m - 2$ non-trivial such definitions. Proposition 5.2 tells us that if $T = \{0,1\}^n - S$ contains two elements that have different balance, we can find a sanitizer that for all elements in $S$ discerns this element from all its neighbors. Proposition 5.3 tells us that there are at most $2^{m/2+1} - 2$ non-empty sets $T$ where all elements have equal balance. Since $1/(x-2) \leq 2/x$ for $x \geq 4$, and since $n \geq 1 \implies m \geq 2$ which in turn implies $2^m \geq 4$, we get

$$\begin{aligned} \frac{2^{m/2+1} - 2}{2^m - 2} &\leq 2 \cdot 2^{m/2} \frac{1}{2^m - 2} \\ &\leq 2^{m/2} \frac{4}{2^m} = \frac{4}{(\sqrt{2})^m} = \frac{4}{(\sqrt{2})^{2^n}}. \end{aligned} \tag{1}$$

This means that the fraction of useful $\sigma$'s that Proposition 5.2 does *not* apply to is exponentially small in $m$ and doubly exponentially small in $n$.

We now summarize the above as follows.

**Theorem 5.4.** *For all but a negligible fraction of possible non-trivial syntactic definitions of privacy on $n$-bit databases, there exists a sanitizer such that every sensitive database is a part of $n$ counterexamples.*

*Proof.* By Propositions 5.2 and 5.3, (1), that $x \mapsto 4(\sqrt{2})^{-x}$ is a function that decreases super-polynomially, and the isomorphy between proper non-empty subsets of $\{0,1\}^n$ and non-trivial syntactical definitions of privacy $\sigma$. □

In the above, we constrained the adversary to only consider hypotheses pairs arising from single bit differences under a fixed encoding of $n$-bit data. Doing this results in a much weaker adversary than the adversary we described in Section 4. Theorem 5.4 tells us that almost all syntactical definitions of privacy allow at least one way of answering questions that exposes *every* sensitive database to *all* its possible vulnerabilities for this constrained and much weaker adversary.

# 6 Differential privacy and the single unknown bit

The way to avoid the above problem is to introduce uncertainty into the inference sketched above. In other words, given bit string $s$ and an unknown bit $b$, there should be uncertainty whether $f(sb) = f(s0)$ or $f(sb) = f(s1)$. This means that $f$ cannot be deterministic and consequently must be randomized. We can think of $f$ as a random algorithm that on input $d$ first chooses a probability density or mass $p_d$ and then returns a random variate according to this. Note that we can let the choice of density or mass be deterministic so that only the returned variate is chosen randomly.

Now, let $f$ be randomized and let $L_i(y) = P(f(si) = y) = p_{si}(y)$ for $i \in \{0, 1\}$ describe the likelihood of $b = i$ on receiving $y = f(sb)$, and let

$$\Lambda_i(y) = \frac{L_i(y)}{L_{1-i}(y)} = \frac{P(f(si) = y)}{P(f(s(1-i)) = y)}.$$

From a Bayesian perspective, $\Lambda_i(y)$ describes the degree to which we on seeing $y$ should update our a priori preference of $b = i$ over $b = 1-i$. Alternatively, from a hypothesis testing perspective, the likelihood ratio $\Lambda_i(y)$ is the test statistic used to determine whether to reject hypothesis $b = i$, and for simple hypotheses such as ours, the Neyman-Pearson lemma states that this test is the most powerful. Consequently, we can interpret $\Lambda(y) = \max_{i\in\{0,1\}} \Lambda_i(y)$ to represent the upper probabilistic bound on disclosure of $b$ from $y$. The closer $\Lambda(y)$ is to 1, the less we learn about $b$ from $y$. Differential privacy generalizes this bound to all databases and single record differences.

Let a record be an element from a set $V$, and let two databases $d_1, d_2 \in V^n$ for some positive integer $n$ be neighbors if they differ at most in a single record.

**Definition 6.1** (Differential Privacy [9])**.** A randomized algorithm $f$ taking input from $V^n$ and range $W$ is $\epsilon$-differentially private if

$$\frac{P(f(d_1) \in S)}{P(f(d_2) \in S)} \leq \exp(\epsilon)$$

for any neighboring databases $d_1, d_2 \in V^n$ and measurable $S \subseteq W$.

Worth noting is that in order to prove the above bound, the probabilities must be taken over what we can control, which is the randomness in the algorithm as opposed to the randomness in the data. Consequently, the above definition is independent of the data, and we cannot check whether a given value was produced in a differentially private manner. Furthermore, as suggested by the Bayesian view presented above, the differential privacy bound is valid independently of any a priori knowledge.

If we want a particular function $q$ computed from data $d$, it makes sense to choose sanitizer $f$ such that its output is concentrated around $q(d)$. If we do this, $f$ is a randomized version of the query response algorithm for $q$. Much research into differential privacy goes into designing $\epsilon$-differentially private versions of query response algorithms that maximize the concentration around $q(d)$ for some $q$ under the $\epsilon$ constraint. Consequently, the parameter $\epsilon$ represents a "knob" that trades off usefulness (concentration) against the ability to infer something about an individual, i.e., privacy.

Two further important properties that differential privacy has are: additive composition and post-processing invariance. We restate these more formally as (for more in-depth discussion and proofs see, e.g., [10]) follows.

**Theorem 6.1** (Composition of differential privacy [8])**.** *Let $f_1$ and $f_2$ be algorithms that are differentially private with $\epsilon_1$ and $\epsilon_2$, respectively. Then, the query $q(D_1, D_2) = (y, f_2(y, D_2))$ for $y = f_1(D_1)$ on any two databases $D_1$ and $D_2$ is $(\epsilon_1 + \epsilon_2)$-differentially private.*

The composition property means that the utility – privacy knob we have in the parameter $\epsilon$ can be used to adaptively budget for accumulated privacy "expenditure" incurred over time across different queries and databases. This is particularly important as Dinur and Nissim showed in 2003 that there is only a finite and even small number of questions we can answer about a database in a useful way before we start leaking potentially sensitive information [5]. A nice example of how synergy by collaboration is achieved is given by Sarwate et al. [19].

**Theorem 6.2** (Post processing for differential privacy [10])**.** *For any non-private randomized algorithm $g$ on databases, if $f$ is $\epsilon$-differentially private, then $h(x) = g(f(x))$ is $\epsilon$-differentially private.*

The post-processing property means that we can use differentially private results in any manner we wish without losing the privacy protection that differential privacy gives.

## 6.1 Revisiting Alice and Bob

Returning to the example involving Alice and Bob from Section 3, we can restrict Alice to receive a database count (i.e., number of rows in a database for which some predicate is true) as a variate from a Laplace distribution centered on the true count with standard deviation proportional to $\epsilon^{-1}$. Adding carefully chosen Laplace noise to query results is known as the Laplace Mechanism [9]. Due to the composition properties of differential privacy we can keep track of the overall inference likelihood change that Alice accrues even when she uses multiple queries. For Alice, the utility – privacy trade-off knob $\epsilon$ means that being relatively insensitive to individuals does not imply poor population statistics.

## 6.2 Revisiting disclosure control

We can use the post-processing property to close the circle back to Section 4 and Definition 4.1 of disclosure in terms of an inference machine $M_r$ for a property $r$.

**Proposition 6.3** (Disclosure control by differential privacy)**.** *If $f$ is $\epsilon$-differentially private, then for any neighboring databases $d, d' \in V^n$ and any $x$, $r$ and $M_r$,*

$$P[M_r(x, f(d)) = r(x)] \leq \exp(\epsilon) P[M_r(x, f(d')) = r(x)].$$

*Proof.* Fix $x, r$, and $M_r$, and let $g(y) = M_r(x, y)$. The theorem then follows directly from Theorem 6.2. □

In particular, Proposition 6.3 holds for any $x$ such that $x \in d$ but $x \notin d'$. This means that someone wanting to recruit for a study can say that "any disclosure about you will become at most $\exp(\epsilon)$ more likely on you entering the study as we are only releasing $\epsilon$-differentially private computations." Importantly, differential privacy simultaneously holds for all properties $r$, including identity. Consequently, we do not need to take potentially subjective choices regarding sensitivity of properties or attributes into account.

## 6.3 Limitations

While many types of questions about data can be answered well with differential privacy, there are questions that are hard to answer with reasonable accuracy if we require differential privacy. In general, this applies to queries $q$ that are very sensitive to single point substitutions. Examples of such include questions regarding connectivity in graphs; a single node deletion can change connectivity radically. Theoretical impossibility results also exist, for example regarding useful threshold queries on infinite domains [2]. What is not clear is whether these negative results are exclusive to differential privacy or are inherent to a larger notion of "strong" privacy.

# 7 Discussion

As we have argued above, anonymity and syntactic privacy will always leave doubt regarding protection of privacy. This situation is not helped by empirical risk analyses in support of syntactic approaches that implicitly only consider 1-to-1 relationships [11]. Defining privacy in terms of randomization independent of the data avoids the problems with anonymity and syntactic privacy, and allows answering questions regarding privacy risk quantitatively. However, a challenge with this lies in that such definitions are incompatible with current information exchange that depends on sharing data that has been anonymized according to some plausibly checkable criterion. Such exchange supports the massive data broker industry and is crucial to current secondary use of health information [24]. In this, abandoning a focus on anonymity and syntactic privacy represents a potentially expensive paradigm shift.

The ability to quantify privacy risk is also relevant for public policy formation. It is a requirement for making decisions based on rational risk – benefit analyses where we need to quantify both sides reliably. Such rational support for privacy policy might become even more important if trust in public management of population data erodes.

Worth noting when considering the above is that there is no necessary contradiction between strong, quantifiable privacy and utility, in fact it can enable use that is otherwise difficult [7]. That said, modern approaches to data and informational privacy such as differential privacy are not a technical panacea. It seems clear that protection of privacy will always require regulatory and contractual means. Nevertheless, we should strive to continually identify applications and areas where we can apply the strongest technical protections available.

# References

[1] Home Page of EU GDPR, 2017.

[2] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Differentially Private Release and Learning of Threshold Functions. *arXiv:1504.07553 [cs]*, April 2015.

[3] Gregory. J. Chaitin. Randomness and Mathematical Proof. *Scientific American*, 232:47–52, May 1975.

[4] Tore Dalenius. Towards a methodology for statistical disclosure control. *Statistik Tidskrift*, 15(429-444):2–1, 1977.

[5] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS '03: Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 202–210, New York, NY, USA, 2003. ACM.

[6] Cynthia Dwork. Differential Privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12, Berlin, Heidelberg, 2006. Springer-Verlag.

[7] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, August 2015.

[8] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *Advances in Cryptology (EUROCRYPT 2006)*, volume 4004, pages 486–503, Saint Petersburg, Russia, May 2006. Springer Verlag.

[9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the Conference on Theory of Cryptography*, 2006.

[10] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, August 2014.

[11] Khaled El Emam and Fida Kamal Dankar. Protecting Privacy Using k-Anonymity. *Journal of the American Medical Informatics Association*, 15(5):627–637, September 2008.

[12] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition Attacks and Auxiliary Information in Data Privacy. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 265–273, New York, NY, USA, 2008. ACM.

[13] Ruth Gavison. Privacy and the Limits of Law. *The Yale Law Journal*, 89:421–471, 1980.

[14] Adam Meyerson and Ryan Williams. On the Complexity of Optimal K-anonymity. In *Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '04, pages 223–228, New York, NY, USA, 2004. ACM.

[15] Barrington Moore. *Privacy: Studies in Social and Cultural History*. M.E. Sharpe, 1984.

[16] Paul Ohm. Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization. SSRN Scholarly Paper ID 1450006, Social Science Research Network, Rochester, NY, August 2009.

[17] Willem G. van Panhuis, Proma Paul, Claudia Emerson, John Grefenstette, Richard Wilder, Abraham J. Herbst, David Heymann, and Donald S. Burke. A systematic review of barriers to data sharing in public health. *BMC Public Health*, 14:1144, November 2014.

[18] Office for Civil Rights (OCR). Privacy, May 2008.

[19] Anand D. Sarwate, Sergey M. Plis, Jessica A. Turner, Mohammad Reza Arbabshirani, and Vince D. Calhoun. Sharing privacy-sensitive access to neuroimaging and genetics data: A review and preliminary validation. *Frontiers in Neuroinformatics*, 8, 2014.

[20] Cynthia Schairer, Sanjay R. Mehta, Staal A. Vinterbo, Martin Hoenigl, Michael Kalichman, and Susan Little. Perceptions of molecular epidemiology studies of HIV among stakeholders. *Journal of Public Health Research*, 6(3), December 2017.

[21] Daniel J. Solove. A Brief History of Information Privacy Law. SSRN Scholarly Paper ID 914271, Social Science Research Network, Rochester, NY, July 2006.

[22] Latanya Sweeney. K-anonymity: A Model for Protecting Privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, October 2002.

[23] Latanya Sweeney. Personal communication, December 2010.

[24] Adam Tanner. Strengthening Protection of Patient Medical Data, January 2017.

# Where is the web still insecure? Regional scans for HTTPS certificates

Anushah Hossain[1], Kristina Nelson[1], and Tjerand Silde[1,2]

[1] University of California Berkeley, Berkeley, California, USA
{anushah.h, krisn, tjerand.silde}@berkeley.edu
[2] Norwegian University of Science and Technology, Trondheim, Norway
tjerands@stud.ntnu.no

**Abstract**

To better understand web security as it is experienced around the world, we scan the top 500 most visited sites of internet users from nine countries of interest. We document HTTPS usage, the encryption algorithm, and certificate information, including issuing date and length of validity. Insights from this project will help benchmark web security prior to upcoming browser interventions[3], and identify regions that may benefit from capacity building to implement TLS in the future.

## 1 Introduction

HTTPS is an extension of the file transfer protocol HTTP that uses Transport Layer Security (TLS) to ensure that the connection between computers is secured against eavesdroppers and that information sent remains untampered. Sites have been rapidly moving to HTTPS from plain HTTP in the past two years, though most recent internet-wide scans find that approximately 41.5% of active websites are still insecure [6].

In this paper, we highlight two areas worth heeding in ongoing efforts to secure the web. We first consider regional variation in HTTPS usage amongst top sites browsed by international users. Who is mostly likely to encounter sites that are not secure? To answer this, we scan certificates of the top five hundred most visited sites from nine countries of interest. From the certificate data gathered, we are also able to assess the effectiveness of browser policy changes to incentivize implementation of HTTPS, which we discuss in the second half of the paper.

Our findings suggest that parts of the web remain insecure, and that the risk is not experienced evenly across the globe. We do find, however, that browser-led "nudges" may encourage uptake of HTTPS overall.

## 2 Background

Google and Mozilla both recently announced that they will mark websites explicitly as not secure within the browser address bar if they do not have a valid X.509 certificate. This follows a series of actions taken since 2014 to incentivize HTTPS usage:

1. August 2014: Google made HTTPS-status a ranking signal [4] for internet searches.
2. September 2016: Both Google [9] and Mozilla [7] announced that from January 2017, they will label HTTP pages with password or credit card form fields as "not secure," given their particularly sensitive nature.
3. February 2018: Google announced that from mid-July 2018, Chrome will mark all HTTP sites as not secure.

A November 2015 report by the National Institute of Standards and Technology (NIST) specifies the set of cryptographic algorithms and key sizes considered "secure" [8]. Acceptable hash-functions are: SHA-256, SHA-384, SHA-512 and SHA-3. RSA is secure with 2048 bit keys and Elliptic Curves (EC) with 224 bit keys. SHA-1 was broken in 2005 and should not be used [1]. RSA with 1024-bit keys is also considered breakable by an adversary with sufficient computational power [5]. We evaluate website security in accordance with these metrics in the following sections.

# 3 Methods

We select nine countries – Canada, China, Germany, Ghana, India, Iran, Norway, Russia and USA – that range in geography, income level, and political regime. For each country, we scrape the top five hundred most visited sites from the Alexa top sites service. Scans of these lists portray the security of the web as experienced by these countries internet users, but do not necessarily represent the security of locally-hosted sites.

To extract information on website certificates, we use the OpenSSL python library [10]. We see the X.509 certificate if it exists, and record information about the certificate issue and expiration dates, signing algorithm, encryption algorithm, key sizes and HTTP Strict Transport Security (HSTS) usage. HSTS is a policy mechanism where the browser refuses to set up a connection unless HTTPS is used. Determining which sites implement HSTS is made challenging by the variety of ways sites are able to deliver HSTS. Chrome and other major browser vendors have begun shipping a hard-coded preload list of HSTS websites, so that the browser treats them as HSTS without ever having to receive the header. As a result, the sites we identify as using HSTS may only be a fraction of those that actually implement strict transport security.

We collected the site listing data from Alexa on March 26, 2018, and extracted certificate information from the sites on April 14, 2018. We evaluated both the hash function and key length used to better characterize website security.

# 4 Results

## 4.1 Regional differences

Figure 1 and 2 summarize our results on site security as experienced by users from each country. We include a Global 500 column of the top sites overall, as ranked by Alexa top sites, as a point of comparison for the country results. We report HSTS percentages as a fraction of total HTTPS traffic, but note that these results are likely to be inaccurate as a result of the different ways by which HSTS is implemented.

Our results show that the United States (87%), Norway (85%), Canada (82%) and Germany (81%) have the highest percentages of top sites using HTTPS. China has the lowest fraction with only 54%. We find that some of the countries with the lowest HTTPS usage have relatively high HSTS usage within our sample. In Iran, for example, only 59% of the most visited countries use HTTPS, but 81% of those sites use HSTS. We hypothesize that this is due to differences in local and foreign websites. If the majority of HTTPS sites visited by internet users from Iran are sites belonging to global companies, for example, and those global sites tend to use HSTS, that would be reflected in a higher relative HSTS rate in our data for Iran. To investigate this in the future, we will need to filter websites by country of origin; we plan to begin testing this hypothesis by comparing overlap across country top 500 listings.

| | | Global 500 | Canada | China | Germany | Ghana | India | Iran | Norway | Russia | USA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| HTTPS | | 85% | 82% | 54% | 81% | 63% | 67% | 59% | 85% | 64% | 87% |
| HSTS (% of HTTPS sites) | | 91% | 67% | 74% | 62% | 69% | 69% | 81% | 64% | 73% | 63% |
| Signing Algorithm | ecdsa-with-SHA256 | 12% | 11% | 3% | 12% | 17% | 18% | 9% | 14% | 7% | 12% |
| | sha1WithRSAEncryption | 0% | 0% | 1% | 0% | 0% | 0% | 5% | 0% | 1% | 0% |
| | sha256WithRSAEncryption | 88% | 89% | 95% | 88% | 83% | 81% | 86% | 85% | 92% | 88% |
| Encryption Algorithm, Key Size | EC256 | 17% | 14% | 10% | 15% | 20% | 20% | 11% | 17% | 10% | 15% |
| | RSA2048 | 79% | 84% | 87% | 78% | 76% | 77% | 83% | 75% | 82% | 83% |
| | RSA4096 | 3% | 2% | 2% | 6% | 4% | 2% | 4% | 7% | 7% | 2% |
| Average Certificate Length (months) | | 19 | 20 | 23 | 20 | 18 | 18 | 36 | 21 | 20 | 19 |
| Total Site Count | | 500 | 500 | 500 | 500 | 500 | 500 | 500 | 500 | 500 | 500 |

Figure 1: Results per country.

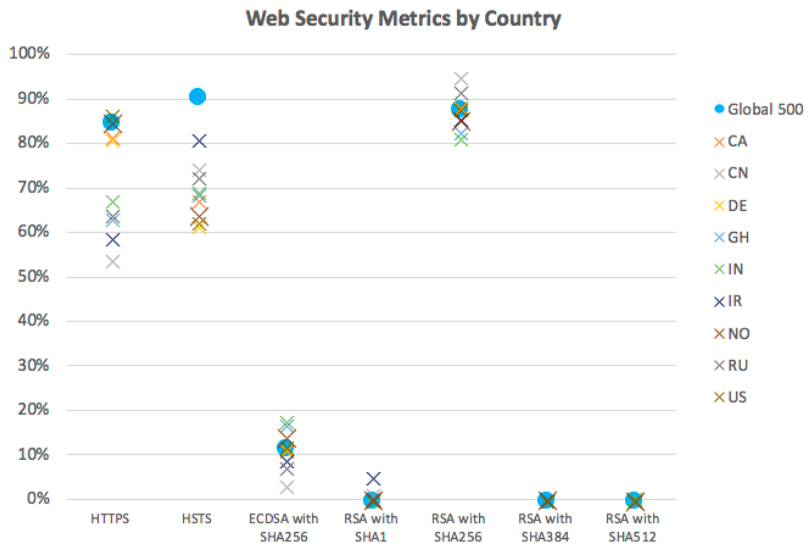



Figure 2: Security parameters.

We find that RSA with SHA256 is the most common signing algorithm used, followed by ECDSA with SHA256. Only a small fraction of the websites used encryption with insecure algorithms and key sizes. From our combined pool of top sites across countries, twenty-six unique websites still use SHA1 for signings and nine websites still use RSA with key size 1024 bits. Iran has the highest percentage of visited sites using the SHA1 hash function, possibly reflecting insecure local content. Neither RSA with SHA384 nor RSA with SHA512 are used in our sample sites.

Comparison of the Global 500 figures against those of each country show that the experiences of many internet users would be masked by aggregated figures. We see that the majority of countries browse fewer HTTPS sites than the global 500 would suggest, and far fewer HSTS sites.

### 4.2 Certificate Issuance

Here, we review the certificate data gathered from all of the unique sites scanned (3221 unique sites out of 4500 total scanned). We find that the certificates are typically issued for one, two, or three years, though we do see shorter periods of three or six months. Several certificates issued in China and Iran were valid for 30 to 100 years, but these was marked as insecure in major browsers.

The graphs in Figure 3 visualize the number of certificates that have been issued, how many are expiring soon and how long they are valid after the issue date. We mark the dates when Google and Mozilla released browser security announcements in the left-most figure. As Google and Mozilla begin rolling out strict warnings for insecure websites, we see that an increasing number of websites are issued new certificates.

The sharp increase in new certificates issued after the browser announcements suggests that many websites recently updated their certificates or were issued their first one. Future analysis will account for coinciding factors that may also have contributed to these trends, but our results thus far support the claim that Google and Mozilla's initiatives are successfully impacting the security culture among the top websites.

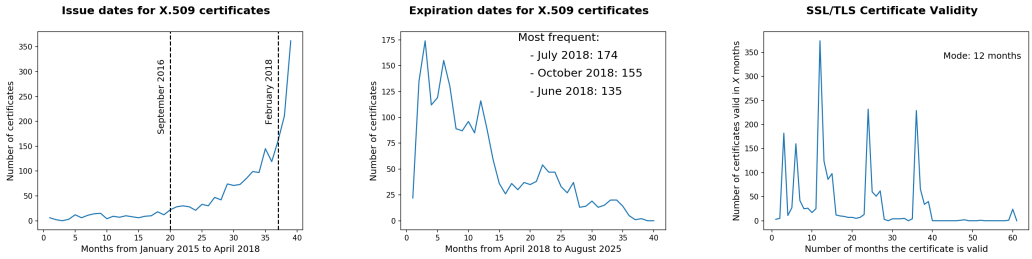



Figure 3: Certificate overview.

## 5 Limitations

Our work is necessarily limited by the sample of sites we examine. As Scheitle et al. [2] document, it is unclear how representative Alexa listings are of the entire web, as they are based on data collected from opt-in browser extensions. Though we considered alternate site listings provided by Quantcast and Majestic Million, we settled upon Alexa listings due to their stratification by country. Other listings typically require paid subscriptions.

We manually verified a subset of websites with valid certificates and websites with errors, and found both false positive and false negative results compared with the Python script. We plan to test whether a purely OpenSSL solution would give different results on the same top 500 lists. It is also possible to use other programming languages such as JavaScript. We hope these next steps can give us both more accurate data and also more insight into which algorithms are used for the symmetric key cryptography component of TLS, and the lengths of their security parameters.

Future work will attempt to distinguish between locally generated content and foreign content to better understand local capacity for implementing TLS, and assess the relationship between site popularity and security. We will also study HSTS more in detail to achieve more accurate result, in addition to check the availability of certificate revocation information and the certificate transparency for the different web pages of interest.

4

# 6 Conclusion

We scanned the top five hundred websites accessed by internet users from nine countries to paint a disaggregated portrait of web security. This contrasts with existing scans that evaluate the entire address space or specific sectors. Our results show significant regional variation and suggest that users from China, Ghana, Iran, and Russia are relatively more susceptible to eavesdropping or corrupted data when sending information over the internet.Future work will expand the sample of countries considered to assess broader regional patterns in security. These initial results suggest that while web security is improving, the benefits are not yet evenly distributed globally. Knowledge of where the web is insecure, as experienced by a country's users, can help policy makers and other stakeholders place targeted pressure on the sites in question to implement HTTPS and HSTS, or recommend stronger encryption algorithms.

Thus far, the majority of action incentivizing web security has come from private sector actors, as we've seen in the success of browser policies and cost-decreasing initiatives such as Let's Encrypt. As Google and Mozilla begin to mark websites without HTTPS as insecure, it will be interesting to see how these results evolve. We expect the percentage of websites using HTTPS to increase significantly in the coming months, given past responsiveness to browser policy.

# References

[1] CWI and Google. Shattered. shattered.io, 2017.

[2] Scheitle et al. Structure and stability of internet top lists. arxiv.org/pdf/1802.02651.pdf, 2018.

[3] Google. Say yes to https: Chrome secures the web, one site at a time. blog.google/topics/safety-security/say-yes-https-chrome-secures-web-one-site-time, 2011.

[4] Google. Https as a ranking signal. webmasters.googleblog.com/2014/08/https-as-ranking-signal.html, 2014.

[5] Matthew Green. How does the nsa break ssl? blog.cryptographyengineering.com/2013/12/03/how-does-nsa-break-ssl/, 2013.

[6] Qualys Inc. Ssl pulse: Monthly scan. www.ssllabs.com/ssl-pulse, 2018. [Accessed April 2018].

[7] Mozilla. Communicating the dangers of non-secure http. blog.mozilla.org/security/2017/01/20/communicating-the-dangers-of-non-secure-http, 2017.

[8] NIST. Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths. dx.doi.org/10.6028/NIST.SP.800-131Ar1, 2015.

[9] The Chromium Projects. Marking http as non-secure. www.chromium.org/Home/chromium-security/marking-http-as-non-secure, 2016.

[10] The pyOpenSSL developers. pyopenssl documentation. pyopenssl.org/en/stable, 2018.

# Fake Chatroom Profile Detection

Patrick Bours[1]*, Parisa Rezaee Borj[2], and Guoqiang Li[3]

[1] NTNU, patrick.bours@ntnu.no
[2] NTNU, parisa.rezaee@ntnu.no
[3] NTNU, guoqiang.li@ntnu.no

**Abstract**

People communicate more and more online in various manners, from posting status updates on Facebook, and sharing videos on YouTube, to direct chatting with friends while playing games and making new (international) friends in online chatrooms. The online society provides a million opportunities for those interested in engaging in harmless activities, but equally well it provides an environment for those with less honorable intentions. Online identities are hard to link directly to physical identities and anonymity and privacy are available to all, independent of the actual intentions of the person.

In our research, we focus on using biometric (keystroke dynamics) and textual (stylometry) features to determine both the correctness of the profile of chatter, as well as ongoing harassment activity. Biometric Keystroke Dynamics (KD) is generally used for authentication purposes to verify the identity of a user while he/she types the (fixed) password. Stylometry is used frequently for Authorship Attribution on long texts (or even books) to establish the identity of the author. In our research, we combine both factors but apply them on short texts that are sent in a chat. From this minimal amount of information, we first want to determine age group (adult vs. adolescent) and gender (female vs. male) of a chatter. This will prevent a 41-year-old male from pretending to be a 14-year-old girl in a chatroom.

Our initial focus in on chat data collected from students and staff of NTNU and we have used SVM to detect gender based on KD data only with an accuracy of approximately 80% per message. Even though this number might not be that high, we noticed for example that correct decision often had a high confidence level, while incorrect decisions were made with a lower confidence level. Even if we take the binary decision of the SVM, then simple majority voting will boost the probability of an error to about 10% with 5 chat messages and less than 1% in 13 chat messages.

In this presentation, we will describe a framework that can be used for creating a safer cyber society by detecting online grooming in chatrooms. We will focus on how this can make a safer online society.

Future work will be to include the stylometry features besides the KD features

*The author presented the paper at the NISK 2018 conference.

# Combining threat models with security economics

Per Håkon Meland*

Norwegian University of Science and Technology, per.hakon.meland@ntnu.no

**Abstract**

In order to defend against cybersecurity threats, we need invest in appropriate protective and reactive security measures, however, we also have to accept that *perfect* security will never be a reality in practice. The adversaries have a huge economic advantage and defenders cannot uphold a sustainable security budget in the long run.[1]

*Threat modelling* typically involves techniques where someone, e.g. a security expert or system owner, tries to think like an attacker in order to determine how systems can be attacked and exploited. In many disciplines, threats are estimated based on stochastic models developed from historical data, e.g. the expected next time to failure for a hardware device or extreme weather frequencies. However, cyberattacks is a relatively new phenomenon, and the attackers are strategic adversaries and not comparable to random natural events. The general unavailability and unreliability of historical data makes it difficult to estimate the likelihood of attacks, especially in areas with rapid technological advances. Therefore, we commonly start from assumptions about the adversary's capabilities, but just as important, we need to consider the financial motivations of possible adversaries.[2] This perspective is regarded as essential for understanding the state of cybersecurity today, as well as how to improve it moving forward.[3]

The goal with this study is to answer the following research question: *How can threat models in combination with economic incentives improve cyber risk quantifications?* When developing these models, there is a need to accept the general unavailability of reliable historical data, and instead build on data about the present to project the future. Identifying reliable data sources and models for opportunity cost for attackers and losses for defenders will be of benefit when estimating likelihood and severity for unwanted events. For example, it is possible to study the mechanisms and trends of the cybercrime market in order to improve our situational awareness about threats in the environment of our system.[4] Market prices contain some information about expectations and may serve as forward-looking indicators, in contrast to statistics calculated from historical data.[5] This is comparable to the arms market in the real world; if there is a high demand for aggressive weapons, then someone might be planning an attack. Also, if you are able to identify that the buyer of these goods happens to be a group or country with a grudge against you, then it is wise to install defence mechanisms that can handle the type of weapons that have been sold. In the cyber world, these dynamics works at a much higher speed, giving the defenders a preparation time of maybe hours or days.

*The author presented the paper at the NISK 2018 conference.

[1] Ross Anderson. "Why information security is hard-an economic perspective". In: *Computer security applications conference, ACSAC 2001.* IEEE. 2001, pp. 358–365.

[2] Tyler Moore and Ross Anderson. "Economics and Internet Security: A Survey of Recent Analytical, Empirical, and Behavioral Research". In: (2011).

[3] Tyler Moore. "The economics of cybersecurity: Principles and policy options". In: *International Journal of Critical Infrastructure Protection* 3.3-4 (2010), pp. 103–117.

[4] Shari Lawrence Pfleeger and Deanna D Caputo. "Leveraging behavioral science to mitigate cyber security risk". In: *Computers & security* 31.4 (2012), pp. 597–611.

[5] Ross Anderson et al. "Security economics and the internal market". In: *Study commissioned by ENISA* (2008).

# Debunking blockchain myths

Roman Vitenberg

Department of Informatics, University of Oslo, Norway
romanvi@ifi.uio.no

**Abstract**

The explosive popularity of the blockchain paradigm has brought upon an avalanche of startups, new strategies and departments at major industrial players, societal and scientific conferences, as well as government-level initiatives in Europe. While computer science has always been trendy and any emerging technology that bursts into popularity has always been accompanied with uncertainty, the situation with blockchain is unique. The uncertainty pertaining to Java, P2P, Web Services, cloud, IoT, and other technologies has been about standardization, competition between alternative solutions, functional extensions, and understanding of technological limitations. In the case of blockchain, however, it starts with the terminology, basic assumptions, and models.

The goal of this short paper is to reflect on the most common misconceptions found in non-scientific (and sometimes, also scientific) articles, and hopefully, contribute towards better understanding of the technology.

## 1 The rapidly evolving blockchain field

Since Bitcoin [12] inception and deployment in 2009, it has been creating waves of discussions and debates about its success. In the wake of Bitcoin came Ethereum [3] that aimed to facilitate programmatic mediation between cooperating parties by defining a feature-rich language for writing mediation software. When the blockchain wave hit the world, many dozens of competing industrial initiatives and proposals for blockchain software were put forward. A number of these proposals resulted in open source implementations, such as the Hyperledger [7] and Corda [2] projects promoted by IBM and the R3 consortium, respectively.

## 2 The architecture of Bitcoin

Bitcoin layered architecture is illustrated in Figure 1. The communication layer implements a specialized broadcast primitive in the Bitcoin P2P network. The data storage layer records all transactions in a distributed ledger implemented as a chain of blocks. The consensus layer realizes a consensus protocol through proof-of-work mining. The business logic layer implements a simple scripting language used for verifying transactions and redeeming their output. The discussion of blockchain myths in Section 3 mentions the requirements and design elements of the data storage, consensus, and business logic layers in Bitcoin and other blockchain systems.

## 3 Blockchain myths

**Myth 1: Blockchain is a replacement for database.** Blockchain differs from database in two fundamental ways. First, database is an organized collection of data representing the current system state. The main functionality of the database is to allow efficient data retrieval, fusion, and aggregation triggered by user queries. In contrast, most blockchain implementations represent a ledger in which a history of transactions (or, more generally, of changes to the system

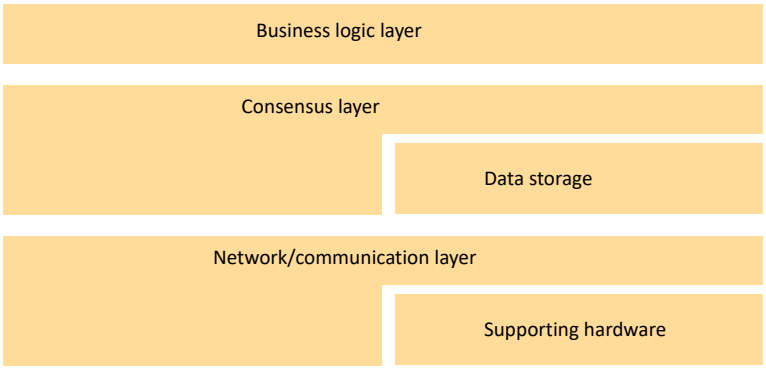



Figure 1: Layered architecture of Bitcoin

state) is recorded. For example, there is simply no concept of user balance in Bitcoin! While Ethereum keeps track of a contract state, it only provides limited means to retrieve and process state data, as explicitly defined by the contract. It does not support abstractions of a flexible query language, data view, schema, join, etc. Besides, Ethereum records a history of all changes to the state, which results in blockchain space being more expensive and the storage less efficient compared to the database. As a result, only specific data elements (such as short transactions or indices) are stored on blockchain. Most blockchain systems in application domains other than financial combine blockchain with offchain storage (databases or dedicated file systems). There are many implementations that support such a combination, e.g., StorJ [9], Filecoin [4], BigchainDB [1], etc.

Secondly, the trust model is radically different as observed in [8]. The database servers typically trust each other, even in federated databases, in the sense that they do not expect attacks from within the system. The main security focus is on making it difficult to compromise a server in the first place. To this extent, database systems defend against malicious clients by using firewalls, strict access control, and many other methods. The situation is fundamentally different in the blockchain environment: the interests of participating nodes are inherently misaligned so that they need to verify information received from each other and run a consensus to agree on changes to the data. While being able to agree on changes and progress in absence of a trusted administrator is a powerful abstraction, it bears a cost tag in terms of performance. If there are no misaligned interests between the participants and attacks from within the system are unlikely, there is little point in using blockchain technologies.

**Myth 2: Blockchain has a definition that is universally agreed upon.** Distributed Ledger Technologies (DLTs, in short) is a well-defined term: it refers to a system that records a ledger of transactions or a history of changes to the system state. The ledger is usually hard to temper with, which is a boon for security yet it also makes it hard to prune the history or compact the ledger.

In many contexts and for many purposes, people equate blockchain with DLTs. Note, however, that both narrower and broader meanings of "blockchain" are in use. Literally, blockchain means "a chain of blocks", which implies a specific data structure for the ledger implementation. A chain of blocks precludes any parallelism between the transactions, however, which has a negative impact on the performance. Some ledger implementations use more a complex data structure such as braids [11], which allow some degree of parallelism by retaining concurrently proposed competing blocks and merging them. Since the term of DLT does not imply

any specific data structure, it covers such a generalization. On the other hand, the term of "blockchain" becomes a misnomer in that case, a distinction that many people are unaware of. In absence of more refined terminology today, "blockchain" is used in the literature to refer to a chain of blocks or generalized DLTs.

To add to the confusion, some systems in this domain do not maintain a distributed ledger at all. For example, Corda [2] allows participating nodes to agree upon and maintain shared knowledge in a non-trusted environment typical for blockchain. However, each piece of information is only shared within a subset of nodes to which the information pertains. Yet, the term of "blockchain" is sometimes used to collectively refer to all systems in the domain including Corda.

**Myth 3: All blockchain technologies are similar to Bitcoin.** While most blockchain initiatives use a layered architecture similar to that of Bitcoin, with each layer serving a similar purpose, there are also significant differences in application requirements on the one hand and operational conditions and attack models on the other. In some cases, the functionality is significantly extended or even replaced. For example, the business layer implementation in Bitcoin is rather rudimentary compared to Ethereum and other later systems. The implementation of layers such as consensus in Hyperledger is radically different compared to Bitcoin. The hypothesis is that this diversity will continue to increase over time. Instead of a single homogeneous blockchain area, multiple technological subareas will be identified as the technology matures.

Perhaps the most basic distinction is that while Bitcoin is open and self-organizing (there are no validated identities, anyone can read the ledger and propose new transactions to include), other Blockchain implementation such as Hyperledger are managed, with authenticated identities, membership, and access control. Such implementations are commonly referred to as private blockchains.

**Myth 4: All blockchain technologies use mining and consume a lot of energy.** All blockchain technologies need to solve a consensus problem. The proof-of-work mining mechanism is Bitcoin solution for the consensus problem. It is notoriously known for consuming exorbitant amounts of energy [6]. Significantly more efficient mechanisms such as proof-of-capacity or proof-of-stake have been proposed. For example, proof-of-stake does not require any significant computation; the mining process is commonly referred to as "virtual" in this situation. However, consensus solutions in smaller-scale systems may not require mining at all. In fact, first consensus solutions appeared in the eighties, long before their applications in the area of cryptocurrency. The PBFT protocol [10] designed in 1999 has been adapted for blockchain needs and adopted as one of the principal consensus solutions in Hyperledger.

**Myth 5: All blockchain systems use a P2P network** It is true that most blockchain systems in existence use a P2P network. For self-organizing systems with many thousands of participating nodes, such as Bitcoin, that might be the most practical way to build a communication between the nodes and scale. For managed, private, member-only systems like Hyperledger or Corda, however, there might be little need to utilize a P2P network. They use it by-product rather than by-design: their architects took the Bitcoin design as a basis and then focused on the innovation related to the business logic, consensus, and storage solution design. Since the communication is not perceived as a bottleneck in blockchain, it does not receive a lot of attention in these rapidly developing systems. In a sense, it is possible to always use a P2P network inside any data center or across a small number of data centers, even though it might be an unnecessary element of the design that leads to a small negative impact on the

performance. It is envisioned, however, that in the future, some of the emerging blockchain systems will do away with P2P networks.

**Myth 6: Private blockchains are necessarily lacking transparency.** Perhaps most of the misunderstandings in non-technical literature are related to the concept of private blockchains. A lot of banks and other companies deploy blockchain solutions without disclosing any technical details, giving the word "private" a negative connotation. However, as explained above, private blockchain refers to a completely different concept; it implies managed rather than self-organizing deployment. It is called private because there is a members-only club. Perhaps "managed" blockchain would be a better term because the code can be transparent and even open source, as it is the case for Hyperledger. In fact, the ledger may also be publicly available for querying; only the right to propose modifications may be restricted to validated identities or members. This may be a suitable model for some blockchain applications such as a blockchain-based voting system.

While it can be argued that protected data query access diminishes transparency compared to Bitcoin, this is actually a desired property in some application domains. Imagine a blockchain system storing sensitive healthcare data which exposes the data to all and allows everyone to store a copy. This would be unacceptable even if the data are encrypted, not to mention incompliant with GDPR.

**Myth 7: Private blockchain are intended for the private market.** Another common misconception is that private blockchain solutions are only deployed by private companies and enterprises. Arguably, most blockchain deployments will be managed and will require validated identities. The managing entities might well belong to the public sector: universities, public healthcare clinics, and even governmental organizations as indicated by the publicized X-Road deployment in Estonia [5].

# References

[1] BigchainDB – The blockchain database. https://www.bigchaindb.com/.

[2] The corda project. https://www.corda.net/.

[3] The ethereum project. https://www.ethereum.org/.

[4] Filecoin – A blockchain-based storage network and cryptocurrency. https://filecoin.io/.

[5] Forbes: The tiny european country that became a global leader in digital government. https://e-estonia.com/tag/blockchain/.

[6] How much power does the bitcoin network use? https://www.thebalance.com/how-much-power-does-the-bitcoin-network-use-391280.

[7] The hyperledger project. https://www.hyperledger.org/.

[8] On distributed databases and distributed ledgers. https://www.corda.net/2016/11/distributed-databases-distributed-ledgers/.

[9] StorJ – Decentralized Cloud Storage. https://storj.io/.

[10] M. Castro and B. Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, Nov. 2002.

[11] B. McElrath. Braiding the blockchain. https://scalingbitcoin.org/hongkong2015/presentations/DAY2/2_breaking_the_chain_1_mcelrath.pdf.

[12] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, May 2009. http://www.bitcoin.org/bitcoin.pdf.

# Assessing face image quality with LSTMs

Tommy Thorsen, Pankaj Wasnik, Christoph Busch,
R. Raghavendra, and Kiran Raja

Norwegian University of Science and Technology, Norway

**Abstract**

Biometric authentication using fingerprints or face recognition is making its way into the mainstream, and there is an urgent need to make these authentication methods as secure and reliable as possible. One way to achieve better performance with a biometric authentication method, is to introduce a quality estimation step early in the pipeline, so that unsuitable, or low-quality samples can be rejected.

While existing work predominantly focuses on algorithms for detecting specific properties of the face images, we investigate whether machine learning techniques can provide a general way to estimate overall face image quality.

We train a selection of neural network types, and discover that a type of Recurrent Neural Network (RNN) called *Long Short-Term Memory* (LSTM) can reliably estimate face image quality, with better performance than the bespoke algorithms.

## 1 Introduction

In all biometric authentication systems, there is a chance of false positives or false negatives. Some genuine login attempts will be denied, and some erroneous attempts will succeed. The chance of such errors increase if the biometric samples are of low quality. Some users will attempt to enter low-quality samples on purpose, to cheat the system, and others will do so unintentionally. For a face recognition system, a sample may be of low quality if for instance the illumination is bad, or the camera is out of focus, or if the user intentionally conceals parts of their face. Regardless of the users' motivations, it is important that we are able to detect and reject low quality samples.

Determining the quality of a face image is commonly done by running it through a set of algorithms that detect and measure specific properties of the image. A number of such algorithms, can be found in the specification ISO/IEC TR 29794-5. Sharpness and contrast was originally proposed by Werner et. al. [14]. How to use facial symmetry calculations to detect lighting and pose problems is described by Gao et. al. [2]. Raghavendra et. al. [9] use Co-occurrence matrices to detect rotation and yaw of the head. Wasnik et. al. [13] proposes another measure called *edge density*, and use it to determine pose and lighting.

Common for all of the methods above, is that they are algorithms for detecting specific face image properties that have been found to correlate well with face image quality. It would be desirable to develop an approach that would leave out the guesswork as to which properties to measure and how to measure them. Machine learning techniques might provide the tools that we need. LSTMs and other RNNs are commonly used to analyze and make predictions on time series, but it has been shown that they are also suitable for image processing tasks such as handwriting recognition [4] and natural scene labeling [1]. In this paper, we show that the LSTM's ability to learn spatial dependencies and analyze parts of images based on surrounding context, makes them well suited for the task of biometric quality estimation.

### 1.1 Previous work

One of the earliest applications of biometric sample quality estimation was in the context of fingerprint recognition [10]. Ratha and Bolle's approach was to use wavelet compression to estimate fingerprint sample quality.

NIST's NFIQ is a well-known tool for estimating the quality of fingerprint images, but unfortunately the output of this tool is not as reliable as one could wish, and the small number of quality classes is quite limiting. Because of this, a lot of work has gone into the development of NFIQ 2.0, which outputs a score which complies to the international biometric sample quality standard ISO/IEC 29794-1:2016. This score is in the range [0-100], where 100 is the best possible quality score.

One paper that applies the ISO/IEC sample quality standard to face images is *Standardization of Face Image Sample Quality* [2], by Gao et. al. This paper looks at the challenge of estimating the score in a standardized and normalized way, and proposes some relevant measurements, such as facial symmetry.

The paper *Assessing Face Image Quality for Smartphone based Face Recognition System* [13], list an additional number of commonly measured properties, all of which are adopted from *ISO/IEC TR 29794-5*. These include *brightness*, *blur*, and *global contrast factor*.

Convolutional Neural Networks (CNNs), such as AlexNet, described by Krizhevsky et. al. [7], is well suited to processing images. Vizilter et. al. [12] use a CNN to do face image identification with good results. It seems likely that a CNN can also perform well at face image quality estimation.

LSTMs are defined by Hochreiter and Schmidhuber in their 1997 paper called *Long Short-Term Memory* [6]. An LSTM is a type of RNN, which can remember some data for a longer time. LSTMs contain something called *cell state* which can be written to according to the values of certain *gates* inside each LSTM cell. LSTMs typically learn faster, and perform better than regular RNNs.

## 2 Methods

### 2.1 Neural network architecture

We have implemented three different neural network architectures for comparison. As can be seen in figure 1, the bottom half of the models are the same for all architectures. This is to keep them as comparable as possible. The size of the models, as measured by the file-size of the fully trained graph, is also similar for all architectures.

#### 2.1.1 LSTM

This model consists of LSTMs interleaved with CNN layers and max pooling layers, finally passing through two fully connected layers. There are two layers with an LSTM and a CNN in each layer. The output sizes for the layers (and for all of the components within the layers) are 32 for layer one and 64 for layer two. The model has been trained for 30 epochs with a batch size of 256 and a learning rate of 0.001.

#### 2.1.2 Four-way LSTM

The construction of this model is the same as the regular LSTM model, except for one thing: Instead of using a single LSTM which analyzes the image data from the top-left towards the

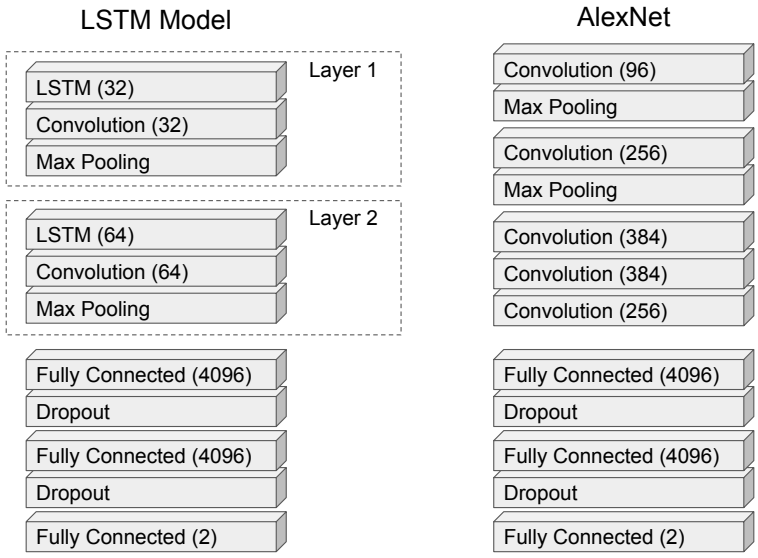



Figure 1: On the left is the LSTM based neural network model, which is used both for the simple LSTM architecture and the Four-way LSTM architecture. On the right is the AlexNet model.

bottom-right, we use four separate LSTMs that analyze the image in four different directions (see figure 2). This idea comes from the paper *Multi-dimensional Recurrent Neural Networks*[3], which makes use of four different 2D LSTMs to analyze images in all four directions. It has been shown that for one-dimensional data, running an RNN in both directions (a bidirectional recurrent neural network) can improve performance [11]. It seems likely that processing a two-dimensional image in all four directions might lead to a similar performance improvement.

To make the combined size of the four LSTMs in this model, the same as for the model with single LSTMs per layer, we divide the layer output size by four to get the individual LSTM output size. The first layer contains four LSTMs with output size 8, and the second layer contains four LSTMs without output size 16. The model has been trained for 30 epochs with a batch size of 256 and a learning rate of 0.001.

#### 2.1.3 AlexNet

This architecture is the AlexNet model as described by Krizhevsky et. al. [7]. This network represents a pure CNN architecture, and is a good reference for us to measure how our LSTM architectures compare to other neural network architectures. The model has been trained for 30 epochs with a batch size of 32 and a learning rate of 0.001.

### 2.2 Datasets

Our neural networks have been trained and tested on a dataset containing of face image from a variety of sources. Among them are: *AR Face database*, *FRGC database*, *NCKU Face database*, *Yale Face database* and the *CASIA Face V5 database*. The training dataset consists of roughly

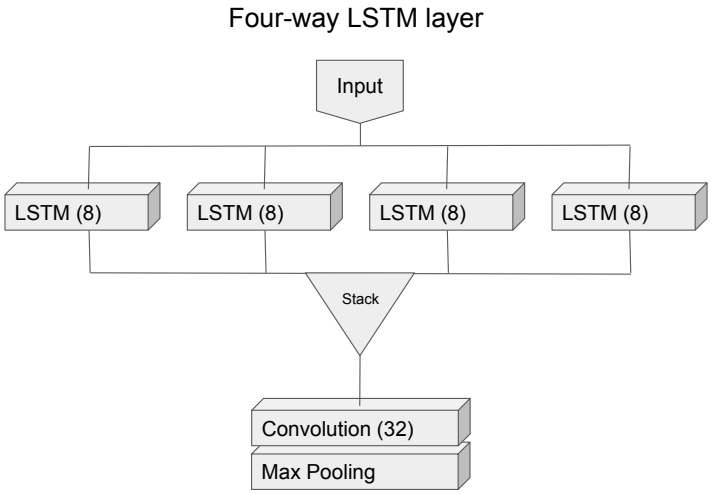



Figure 2: The four-way LSTM layer consists of four regular LSTMs which read the image in four different directions. The results from each LSTM are stacked to provide a single 32 unit output.

30000 face images, labeled as either `good` or `bad`. The neural networks have been trained for binary classification between these classes. The classification confidence (the estimated probability that a given image belongs to the `good` class) is used to derive an ISO compliant score in the range [0, 100].

The datasets used to test the performance of the trained networks contain face images captured with the front cameras of an iPhone 6 Plus and a Samsung Galaxy S7, making them quite realistic, at least in the context of face recognition for mobile phones. Each of these databases contain images of 101 different subjects, with 10 images of varying quality for each subject.

# 3 Results

To measure and visualize the performance of our face image quality assessors, we will use Error Reject Curves (ERCs), as recommended by Grother and Tabassi [5]. ERCs were originally designed to measure the performance of fingerprint image quality assessors, but are suitable for any kind of biometric system. In addition to our three neural network based systems, we also provide the results for a commercial solution.

To calculate an ERC, it is necessary to couple the quality assessor with a face recognition system. The curve shows how the performance of the face recognition system is affected by the quality assessment algorithm operating at different rejection rates. As such, the ERC is a good indicator of the predictive performance of a quality algorithm.

Figure 3 shows the ERCs for both test databases. The curves show that AlexNet does not perform as well as one would have thought, for this task. The regular LSTM performs really

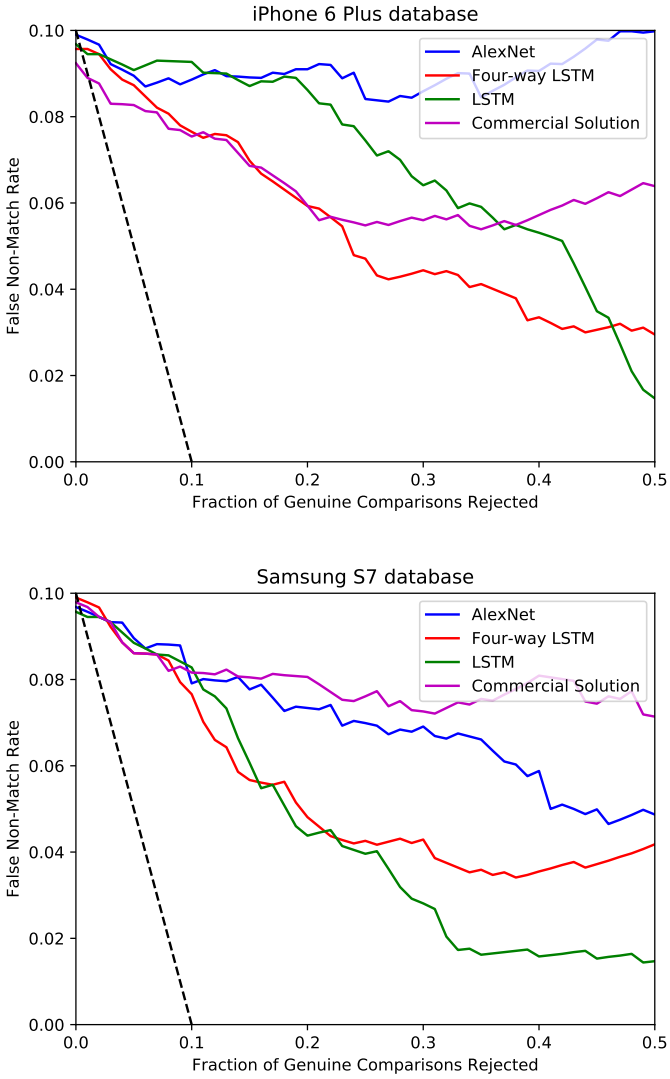



Figure 3: Top: ERCs for the iPhone 6 Plus database. Bottom: ERCs for the Samsung S7 database

well for the Samsung dataset, but for the iPhone dataset, the performance at low rejection rates is not so competitive. The difference between these datasets is mainly the due to the quality of the cameras of the two different smartphones. The Samsung frontal camera is better than the one on the iPhone, and produces higher quality images.

The Four-way LSTM model seems to give the most stable and predictable results. Especially at low rejection rates, it performs really well. The good performance is likely to be due to the four-way architecture's ability to consider each pixel given its surrounding context on all sides.

For a more quantitative analysis of the ERCs, we turn to Olsen et. al. [8] who propose the

| | iPhone 6 Plus Database | | Samsung S7 Database | |
|---|---|---|---|---|
| | $\eta^{erc}_{auc}$ | $\eta^{erc}_{pauc20}$ | $\eta^{erc}_{auc}$ | $\eta^{erc}_{pauc20}$ |
| LSTM | **0.0316** | 0.0132 | **0.0206** | 0.0102 |
| Four-way LSTM | 0.0328 | 0.0106 | 0.0333 | **0.0098** |
| AlexNet | 0.0810 | 0.0131 | 0.0470 | 0.0118 |
| Commercial Solution | 0.0495 | **0.0102** | 0.0739 | 0.0120 |

Table 1: AUC and PAUC for ERC plots for all quality assessors on both databases.

following metrics:

$$\eta^{erc}_{auc} = \int_0^1 ERC - area\ under\ theoretical\ best$$

$$\eta^{erc}_{pauc20} = \int_0^{0.2} ERC - area\ under\ theoretical\ best$$

The first metric is simply the integral of the ERC, giving us the area under the curve. We subtract the area under theoretical best, which corresponds to the area under the black dashed line in the ERCs. The second metric is the same as the first, except we only look at the first 20% of the ERC.

Table 1 shows these metrics. We can see that the regular LSTM performs best for the whole curve, while the Four-way LSTM gives better results for the first 20%.

# 4 Conclusion

We have shown that some types of deep neural networks can work well for estimating biometric sample quality for face images. The results indicate that the performance is as good as, if not better than that of traditional quality estimation methods, as implemented in a commercial face recognition product. We have also shown that RNNs (here represented by an LSTM) performs better at this task than a pure CNN of similar size.

When we consider the relative simplicity of development of a neural network quality assessor, compared to the development complexity of a set of quality measurement algorithms, we believe that this will be a very attractive approach for many prospective implementers.

# References

[1] Wonmin Byeon, Thomas M Breuel, Federico Raue, and Marcus Liwicki. Scene labeling with lstm recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3547–3555, 2015.

[2] Xiufeng Gao, Stan Z Li, Rong Liu, and Peiren Zhang. Standardization of face image sample quality. In *International Conference on Biometrics*, pages 242–251. Springer, 2007.

[3] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Multi-dimensional recurrent neural networks. In *ICANN (1)*, pages 549–558, 2007.

[4] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.

[5] Patrick Grother and Elham Tabassi. Performance of biometric quality measures. *IEEE transactions on pattern analysis and machine intelligence*, 29(4):531–543, 2007.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[8] Martin Aastrup Olsen, Vladimír Šmida, and Christoph Busch. Finger image quality assessment features–definitions and evaluation. *IET Biometrics*, 5(2):47–64, 2016.

[9] Ramachandra Raghavendra, Kiran B Raja, Bian Yang, and Christoph Busch. Automatic face quality assessment from video using gray level co-occurrence matrix: An empirical study on automatic border control system. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 438–443. IEEE, 2014.

[10] Nalini K Ratha and Ruud Bolle. Fingerprint image quality estimation. 1999.

[11] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[12] Yuri Vizilter, Vladimir Gorbatsevich, Andrey Vorotnikov, and Nikita Kostromov. Real-time face identification via cnn and boosted hashing forest. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 78–86, 2016.

[13] Pankaj Wasnik, Kiran B Raja, Raghavendra Ramachandra, and Christoph Busch. Assessing face image quality for smartphone based face recognition system. In *Biometrics and Forensics (IWBF), 2017 5th International Workshop on*, pages 1–6. IEEE, 2017.

[14] Martin Werner and Michael Brauckmann. Quality values for face recognition. In *NIST Biometric Quality Workshop*, volume 3, 2006.

# Baseline Evaluation of Smartphone based Finger-photo Verification System: A Preliminary Study of Technology Readiness

Pankaj Wasnik, Martin Stokkenes, Mihkal Dunfjeld, Raghvendra R., Kiran Raja, and Christoph Busch

Norwegian Biometrics Laboratory,
NTNU, Gjøvik, Norway
{pankaj.wasnik;martin.stokkenes;mihkaljd;raghavendra.ramachandra;
christoph.busch;kiran.raja}@ntnu.no

**Abstract**

Increasingly, smartphones come embedded with more and more advanced biometric recognition systems. The high quality embedded sensors capture the biometric data which helps to get higher biometric recognition accuracy and security. On the other hand, these sensors increase the cost in terms of battery usage, take up extra physical space also some hardware cost. This paper investigates the reliability and effectiveness of fingerphoto verification system developed using a smartphone. Consequently, biometric samples can be collected without the need for dedicated hardware, thus providing a cost-effective and reliable way to do biometric authentication. The focus of this paper is to study various baselines to evaluate the readiness of the technology. To this extent, we present a comparison between three non-commercial baseline systems with one commercial system. The commercial off-the-shelf (COTS) system shows the best performance of an equal error rate of 6.08%.

## 1 Introduction

Biometric characteristics have been used to verify and identify the individuals for a long time. With the advances in technology, substantial research has been going on smartphone biometrics since today's mobile phones contain critical data from users such as emails, personal and, banking data. For smartphones, for a long time, the only viable authentication methods are limited to pins and passwords. These methods are inconvenient as pins are often limited to a small number of digits, and severely restrict the keyspace, thus making the system secure only against low-effort adversaries while providing little security against advanced threats with considerable knowledge and resources. This gives a system where user-friendliness decreases as security increases and vice versa, effectively forcing people to choose weak passwords. As the amount of personal and sensitive information people store on their phones go up, security becomes increasingly important. Touch-based fingerprint recognition has been in use on smartphones for several years. However, there are some drawbacks to using traditional touch-based sensors. It requires a dedicated sensor on the device which increases the cost of the device; fingerprints can be deformed due to varying pressure, latent fingerprints can be extracted or add noise to subsequent images. There is also a hygienic concern for touching a sensor as large numbers of people may have come in contact with it.

Apart from the traditional biometric fingerprint recognition systems, there are also sensors based on optical scanning techniques using an image sensor to sample the data from the presented finger. With the evolution of cameras in modern smartphones, one can consider a similar approach, capturing images of a person's finger, potentially removing the need for a specialized

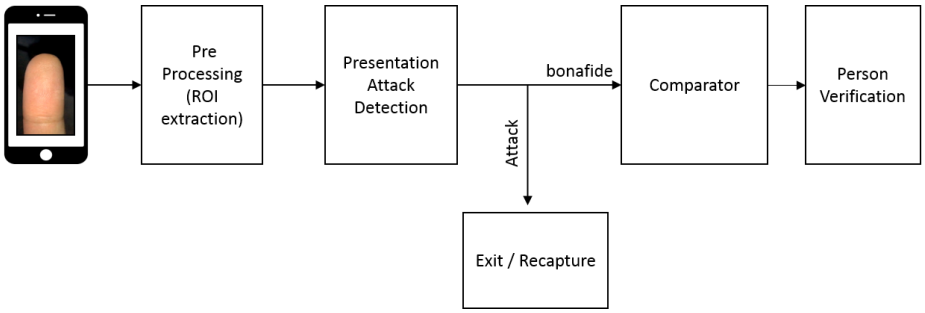



Figure 1: Block diagram of a typical fingerphoto verification system

sensor to be included in the device. However, this approach can increase the number of degrading factors such as illumination, orientation, focus, during the capture process. To compensate for such factors requires a method be robust towards these challenging factors.

## 1.1 Previous Work

There are previously limited works which explore biometric recognition based on fingerphotos captured using a smartphone camera or similar approaches [15, 4, 5, 12, 16, 18, 19, 9]. In [4, 12, 18] authors have explored the traditional minutia based approaches. They employed various minutia extraction algorithms to extract features from the fingerphotos for the final comparison. [19] proposed a contactless fingerphoto recognition system using smartphone cameras where they use SURF algorithm to extract non-conventional scale-invariant features. In [5] another approach is applied by Hiew et al. where a support vector machine (SVM) is trained and used in the verification process.

In recent years, researchers have been trying to establish a meaningful correlation between fingerprints and corresponding fingerphotos. One such method is proposed in [10] where interoperability between legacy fingerprint databases and fingerphotos is discussed. Their research shows that the performance of cross-matching fingerprint images with fingerphotos is still not that good enough which is partly due to skin deformations caused by pressing the finger against the sensor. In [7] authors explore the possibility of reconstructing 3D features from 2D images. There is also research in [8], on using 3D finger images rather than traditional 2D images. Common for all these approaches is the reliance on several pre-processing steps to segment and enhance the acquired finger samples.

## 1.2 Our Work

The above-mentioned earlier works present the feasibility of fingerphoto recognition systems. However, they do not perform well concerning biometric performance. One of the possible reason could be quality of the data captured using the smartphone cameras. In this paper, we aim to evaluate the feasibility and readiness of the fingerphoto verification systems on the state of the art smartphones, to testify whether it could be an excellent alternative to the traditional fingerprint recognition system. To this extent, we present the comparison between various fingerphoto recognition systems using non-commercial and commercial methods.

The rest of the paper is organized as: Section 2 describes and details the methodology used in this paper. Section 3 presents the database used. The experiments and results are discussed in Section 4, and Section 5 gives the concluding remarks.

# 2 Methodology

This section describes an overview of the fingerphoto verification system using commercial and non-commercial systems. Figure 1 shows the block diagram of the proposed verification framework. We would like to notify the reader that although the block diagram contains a Presentation Attack Detection (PAD) module, our work is limited to the extent of verification system only. However, in practice, every biometric recognition system should have a PAD employed in it. The COTS by Neurotechnology [14] is used as a baseline commercial system whereas three feature extraction techniques i.e., the local binary patterns (LBP), the histogram of oriented gradients (HOG) and the binarized statistical image features (BSIF) along with probabilistic collaborative representation classifier (ProCRC) are used as the non-commercial baseline systems.

## 2.1 ROI Extraction

The very first step in the verification pipeline is the extraction of the region of interest (ROI). For this purpose, an overlay mask is displayed on the mobile phone screen, when the user is capturing his or her sample an iOS application. The mask helps for two primary purposes: to place the finger at the right place and to extract the ROI. The region inside the rectangle is first processed to remove the background information, and later a suitable rectangular portion is cropped from the image. The cropped image is then used for feature extraction and enrollment process. Figure 2 shows the application screen along with the intermediate and final results of the ROI extraction process.

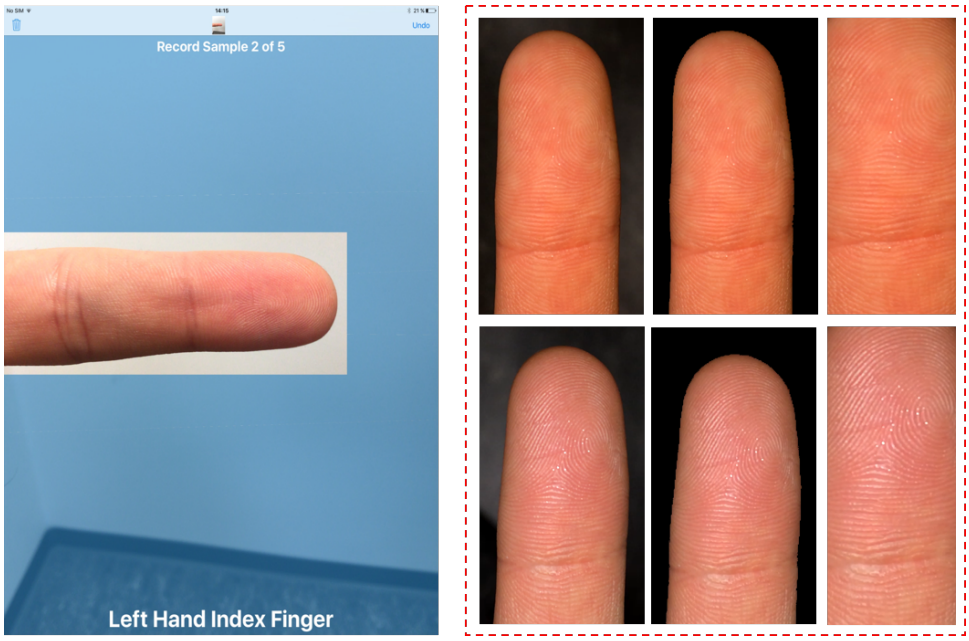



Figure 2: Data capture using the developed iOS application with the transparent blue mask. Fingers in red box show the intermediate and final results of ROI extraction process

## 2.2 Feature Extraction

The feature extraction is done using three non-commercial techniques i.e, LBP, HOG and BSIF. The detailed description of theses techniques is provided in the subsequent paragraphs of this section.

### 2.2.1 Local Binary Patterns

The LBP is a very well known texture descriptor used to extract the textural characteristics of an image. In typical LBP feature extraction, we first divide an image into cells containing $m$ x $m$ pixels and image will be binarized based on the gray value of its eight neighboring pixels. The pixel is assigned to the value of 0 if its gray value is higher than the neighboring pixels and vice-versa. Therefore, at a time any pixel is represented as a combination of these eight binary digits. These numbers are then further used to compute a histogram of the cell. The final feature vector is obtained by concatenation of the histograms of each cell together [1].

### 2.2.2 Histogram of Oriented Gradients

The HOG feature extractor first computes the intensity gradients of uniformly spaced, predefined normalized cells. It is mainly used in the shape detection and similar problems. The input image is first divided into the cells of equal sizes, and the 1-D histogram of gradients for each cell is then calculated to get the shape and appearance of the objects [3].

### 2.2.3 BSIF

The LBP image descriptor inspires the BSIF feature extraction. Similar to the LBP, here also the image is first divided into cells and a bit string of binarized pixels is represented in the histograms. The primary difference between these two methods is in BSIF image descriptor the filters are learned by image statistics while in LBP filters are predefined. The multiplication with the number of filters determines the length of the bit string. Further, the product of each filter multiplication determines the bit string and the bit is set to 1 if the product is $> 0$ and vice-versa [6].

## 2.3 Comparator

In general, given a feature vector as input to a trained comparator, it will classify whether a sample belongs to genuine or an imposter user. However, in case of commercial systems, we need not extract features and pass it to the comparator, simply one can give the captured image as an input to the system.

### 2.3.1 ProCRC

In this paper, we have employed a probabilistic collaborative representation classifier as our baseline non-commercial comparator. The ProCRC is a classifier that works by computing the probabilities of a sample belonging to each of the multiple classes and makes a decision based on the highest probability [2].

### 2.3.2 VeriFinger SDK from Neurotechnology

The VeriFinger SDK from Neurotechnology is used as a baseline COTS system [13]. In case of the VeriFinger SDK, 500 pixels per inch (ppi) resolution is recommended for an input image. Therefore, we have to convert the extracted ROIs to 500 ppi.

# 3 Database

A newly constructed database of 48 subjects is used to evaluate all four baseline systems. The database consists of two sessions one for enrollment and one for the probe. During session 1, we collected 10 images per subjects whereas in the second session we collected only five images. The images are captured using the rear camera of the iPhone 6S, and the extracted ROI is resized to 200x500 pixels. Further, the data from the first session are used for training the ProCRC classifier while the second session data are used for testing the classifier. In case of the COTS, data from the first session data are used for enrolment and the second session data are used for obtaining the verification scores. The input images for the COTS are converted to 500 ppi using ImageMagick Library [11].

# 4 Experiments and Results

The training of the ProCRC classifier is done by learning the feature vectors obtained from LBP, HOG and BSIF feature extractors. The ProCRC classifier tries to maximize the joint likelihood of a test sample belonging to each of the classes, and finally, the sample is classified into the class which has the maximum likelihood.

The training dataset consists of 48 classes in total. We have used 10 images of left index finger from the first session data for training the classifier while in case of the VeriFinger SDK we have used these images for enrollment of the subjects. In total, the training dataset consists of the 480 images. In the testing phase, we have used the data from the second session. These experiments resulted in 240 genuine and 11280 imposter comparisons. The accuracy of the systems is evaluated in terms of false match rate (FMR), false non-match rate (FNMR) and the equal error rate (EER). The verification results are presented using very well adopted evaluation criteria for any biometric system, i.e., the detection error trade-off (DET) and the receiver operating characteristic (ROC) curves.

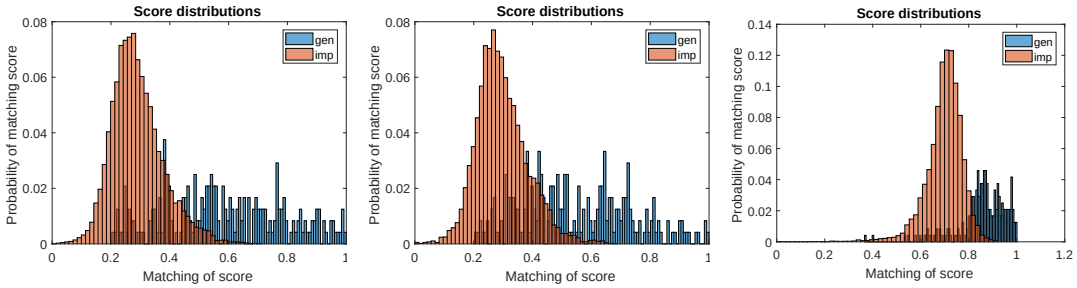
## 4.1 Evaluation of Non-Commercial System

The Figure 3 shows the genuine and impostor score distributions for the best settings for the non-commercial baseline systems (See Table 1). In an ideal situation, the genuine comparisons should be separated from the impostor comparisons, indicating the genuine comparisons yield a high similarity score while impostor comparisons yield a low similarity score. In practice, some degree of overlap between genuine and impostor distributions is always expected.

From the figure, it is evident that there is a high overlap in score distributions of all three feature extractors, resulting in high EER values and low performance. In particular, if we analyze the figures 3a and 3b carefully, we can observe that there is a significant overlap between the genuine and imposter scores of LBP and HOG. Though system based on HOG appears to have a higher number of genuine scores with a low similarly score, hence shows slightly higher an EER of 18.67% compared to LBP based system with an EER of 15.85%.

Similarly, we can see that the system based on BSIF also have a significant overlap; however, the number of misclassified scores are low, resulting in higher performance than LBP and HOG with an EER of 12.84%. The particulars of the EERs are tabulated in Table 1 with the best settings of the feature extractors.

We can draw the similar observations by plotting the DET curves. From the Figure 5a, we can see the DET curves for all three non-commercial systems appear to be farther from the origin. Furthermore, from the Figure 5b we can see the Genuine Match Rate (GMR) at FMR $= 10^{-2}$ for LBP, HOG, and BSIF is approximately 30%, which is a very low score.




(a) LBP distribution scores (b) HOG distribution scores (c) BSIF distribution scores

Figure 3: Genuine and imposter score distributions of baseline systems with best settings for LBP, HOG and BSIF

In order to improve the performance, we examined different possible combinations of cell size and filter size. The details are given in the Table 1. For the LBP and HOG, we tried to adjust the cell size with sizes of 4x4, 8x8, 16x16 and 32x32. Table 2 shows the EERs obtained for the different cell sizes. In case of the HOG based system, it is evident, more the number of features better is the performance. However, we cannot see similar behavior in case of the LBP based system. A cell size of 16x16 and 8x8 yielded the best performance in terms of the EER for the LBP and HOG respectively. The increasing or decreasing cell size further showed the degrading performance.

| **Feature Extraction** | **EER(%)** | **GMR(%)@ FMPR=0.001** | **Settings** |
|---|---|---|---|
| BSIF | 12.83 | 32.11 | FilterSize= 9x9 10 Bit |
| LBP | 15.85 | 34.01 | CellSize = 16x16 |
| HOG | 18.67 | 29.8 | CellSize = 8x8 |
| COTS | **6.05** | **83.8** | Not Applicable |

Table 1: EER and GMR@ FMR=0.01 values for all four baseline system

Furthermore, in case of the BSIF, we employed 40 texture filters ranging from the size of 5x5 to 15x15 with eight different bit sizes ranging from 5-bit to 12-bit. Table 3 shows the selected BSIF filters and the corresponding EER. The experiments yielded in the conclusion that decreasing the filter size below 7x7 and above 13x13 does not help in improving the performance. The best EER obtained is of 12.83% with the filter size of 9x9 and 10-bits. Overall, it is apparent that with the current level of performance for all three non-commercial systems appear to be non-deployable.

## 4.2 Evaluation of Commercial System

The Figure 4 gives the genuine and impostor score distributions for the commercial system. From the figure, we can see that the impostor score distribution is very dense and concentrated towards zero along with a less overlapped area with genuine distribution. However, it is challenging to set the threshold as both of the distributions are densely populated near to each other. Furthermore, from the figures 5a and 5b, we can see that the performance of the commercial system outperforms all of the non-commercial systems. With the 500 ppi image in-

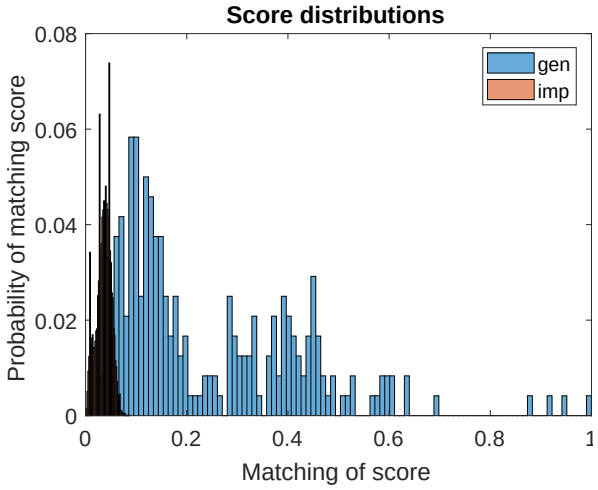
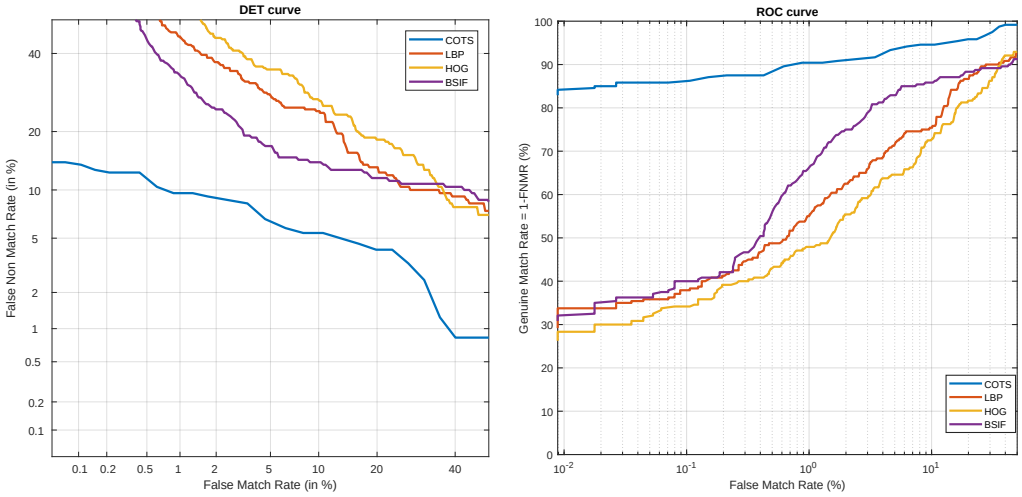



Figure 4: Distribution of scores for commercial system

put for enrollment and testing, we achieved the best performance in terms of EER of 6.05% as well as in terms of GMR at FMR = $10^{-2}$ we achieved the performance of 83.8%. However, it would be interesting to see the performance of the commercial system with respect to different ppi values. However, under the scope of this work, we have only carried out our experiments with the recommended input resolution. The performance of commercial




(a) DET curves (b) ROC curves

Figure 5: DET and ROC curves for all four baseline system

system indicates that the fingerphoto based verification systems are practically feasible and also deployable. However, if we do not intend to use any commercial system, then we must employ the advanced level of image preprocessing to extract the essential features to improve the performance.

| Image Cell Size | LBP | | HOG | |
|---|---|---|---|---|
| | EER(%) | Number of Features | EER(%) | Number of Features |
| 4x4 | 21.16 | 368756 | 18.72 | 218736 |
| 8x8 | 17.07 | 91450 | **18.67** | 52704 |
| 16x16 | **15.85** | 21948 | 20.39 | 11880 |
| 32x32 | 18.78 | 5316 | 22.02 | 2520 |

Table 2: Equal error rates for LBP and HOG with different cell sizes

| BSIF EER (%) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 5 bit | 6 bit | 7 bit | 8 bit | 9 bit | 10bit | 11bit | 12bit |
| **7x7** | 23.43 | **19.38** | **17.05** | 15.46 | **22.60** | 17.04 | 16.21 | 16.24 |
| **9x9** | **22.13** | 22.11 | 17.30 | **14.80** | 29.29 | **12.83** | **14.53** | **15.39** |
| **11x11** | 22.88 | 21.16 | 20.87 | 18.62 | 24.16 | 15.57 | 16.91 | 15.85 |
| **13x13** | 24.19 | 24.41 | 18.68 | 18.36 | 27.64 | 15.09 | 17.30 | 19.36 |
| **15x15** | 29.93 | 29.78 | 24.76 | 21.01 | 26.90 | 17.63 | 16.46 | 18.61 |

Table 3: Equal error rates for BSIF with different texture filters

# 5 Conclusion

Based on the experiments that have been conducted we can conclude that the performance of all baseline non-commercial systems is worse than commercial. We can say that none of the studied feature-extracting techniques can extract enough features for reliable comparison with the templates stored in the database. However, employing advanced pre-processing methods would aid in better feature extraction. Further, in comparison with commercial fingerprint recognition system where EER of 0.8% is achieved [17] we can say that all four baseline fingerphoto systems have the worse performance by at least one order of magnitude.

In conclusion, this paper presents the baseline evaluation of three non-commercial and one commercial system. Further, from results, we can say in case of the LBP and HOG we achieved best the EER of 15.85% for 16x16 cell size and 18.67% 8x8 cell size respectively. While in case of the BSIF the best performing texture filter was 9x9 10 bit. We also achieved the best EER of 6.05% with the commercial system, which proves the feasibility and readiness of the fingerphoto verification system. This motivates us to explore the various image processing, feature extraction and machine learning techniques to overcome the challenges of fingerphoto recognition.

# Acknowledgement

This work was carried out under the funding from the Research Council of Norway (Grant No. IKTPLUSS 248030/O70).

# References

[1] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041, 2006.

[2] S. Cai, L. Zhang, W. Zuo, and X. Feng. A probabilistic collaborative representation based approach for pattern classification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2950–2959, June 2016.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.

[4] Mohammad Omar Derawi, Bian Yang, and Christoph Busch. *Fingerprint Recognition with Embedded Cameras on Mobile Phones*, pages 136–147. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[5] Bee Yan Hiew, Andrew Beng Jin Teoh, and Ooi Shih Yin. A secure digital camera based fingerprint verification system. *Journal of Visual Communication and Image Representation*, 21(3):219 – 231, 2010.

[6] J. Kannala and E. Rahtu. Bsif: Binarized statistical image features. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1363–1366, Nov 2012.

[7] A. Kumar and C. Kwong. Towards contactless, low-cost and accurate 3d fingerprint identification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):681–696, March 2015.

[8] R. Donida Labati, A. Genovese, V. Piuri, and F. Scotti. Toward unconstrained fingerprint recognition: A fully touchless 3-d system based on two views on the move. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(2):202–219, Feb 2016.

[9] Guoqiang Li, Bian Yang, Martin Aastrup Olsen, and Christoph Busch. Quality assessment for fingerprints collected by smartphone cameras. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, pages 146–153. IEEE, 2013.

[10] C. Lin and A. Kumar. Matching contactless and contact-based conventional fingerprint images for biometrics identification. *IEEE Transactions on Image Processing*, 27(4):2008–2021, April 2018.

[11] ImageMagick Studio LLC. Imagemagick. https://www.imagemagick.org/script/index.php.

[12] R. Mueller and R. Sanchez-Reillo. An approach to biometric identity management using low cost equipment. In *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 1096–1100, Sept 2009.

[13] Neurotechnology. Neurotechnology verifinger. http://www.neurotechnology.com/verifinger.html.

[14] NEUROtechnology VeriLook 5.4. VeriLook 5.4 Face SDK . http://www.neurotechnology.com/face-biometrics.html.

[15] Ramachandra Raghavendra, Christoph Busch, and Bian Yang. Scaling-robust fingerprint verification with smartphone camera in real-life scenarios. In *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, pages 1–8. IEEE, 2013.

[16] A. Sankaran, A. Malhotra, A. Mittal, M. Vatsa, and R. Singh. On smartphone camera based fingerphoto authentication. In *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–7, Sept 2015.

[17] K. K. M. Shreyas, S. Rajeev, K. Panetta, and S. S. Agaian. Fingerprint authentication using geometric features. In *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–7, April 2017.

[18] C. Stein, C. Nickel, and C. Busch. Fingerphoto recognition with smartphone cameras. In *2012 BIOSIG - Proceedings of the International Conference of Biometrics Special Interest Group (BIOSIG)*, pages 1–12, Sept 2012.

[19] K. Tiwari and P. Gupta. A touch-less fingerphoto recognition system for mobile hand-held devices. In *2015 International Conference on Biometrics (ICB)*, pages 151–156, May 2015.

# Towards Fingerprint Presentation Attack Detection Based on Short Wave Infrared Imaging and Spectral Signatures

Marta Gomez-Barrero, Jascha Kolberg, and Christoph Busch

da/sec - Biometrics and Internet Security Research Group,
Hochschule Darmstadt, Germany
email: {marta.gomez-barrero, jascha.kolberg, christoph.busch}@h-da.de

**Abstract**

Biometric verification systems are currently being deployed in numerous large-scale and everyday applications. Among other vulnerabilities, presentation attacks directed to the sensor (e.g., using a face mask or a gummy finger) pose a severe security threat. To prevent such attacks, presentation attack detection (PAD) techniques have been proposed in the last decade. For the particular case of fingerprint recognition, most approaches are based on conventional optical or capacitive sensors, acquiring a single image, and which can thus detect only a limited number of materials used to fool the sensor.

In this paper we propose a PAD algorithm based on normalised spectral signatures extracted from Short Wave InfraRed (SWIR) images captured at four different wavelengths. It has been shown that the information contained in the selected SWIR wavelengths can help to discriminate skin from other materials. The extracted features are classified using a Support Vector Machine (SVM), thereby yielding a fast real-time performance which can be implemented on almost any application. The experimental evaluation shows that all but one material considered can be detected, including unknown materials (i.e., not used to train the classifier).

## 1 Introduction

Biometrics refers to automated recognition of individuals based on their biological (e.g., iris or fingerprint) or behavioural (e.g., signature or voice) characteristics [1]. Certainly, biometric recognition is very attractive and useful for the final user: forget about PINs and passwords, you are your own key [1]. In addition, they provide a stronger link between the subject and the action or event. All these facts have allowed a wide deployment of biometric systems in the last decade for different applications, ranging from border control to smartphone unlocking.

In spite of their numerous advantages, biometric systems are vulnerable to external attacks. Among the different possible points of attack summarised in [2], the biometric capture device is probably the most exposed one: in order to launch an attack on the capture device, no further knowledge about the inner functioning of the system is required. Such attacks are known in the literature as *presentation attacks* and defined within the ISO/IEC 30107 standard on biometric presentation attack detection [3] as the "*presentation to the biometric data capture subsystem with the goal of interfering with the operation of the biometric system*". In other words, an attacker can present the capture device with a *presentation attack instrument* (PAI), such as a gummy finger or a fingerprint overlay, in order to impersonate someone else (i.e., active impostor) or to avoid being recognised due to black-listing (i.e., identity concealer).

In order to prevent such attacks, *presentation attack detection* (PAD) methods have been proposed for several biometric characteristics, including iris [4, 5], fingerprint [6, 7] or face [8]. This term refers to any technique that is able to automatically distinguish between bona fide (i.e., real or live) presentations and access attempts carried out by means of PAIs [9] and

has attracted a considerable attention within the last decade. In fact, several international projects, like the European Tabula Rasa [10] and BEAT [11], or the more recent US Odin research program [12] deal with these security concerns. In addition, the LivDet – liveness detection competition series on iris and fingerprint [13] have been running since 2009.

As any other pattern recognition task, PAD techniques can benefit from data acquired with different sensors, thereby providing complementary sources of information [4, 7]. In particular, the use of multi-spectral technologies have been studied for face [14, 15] and fingerprint [16, 17] in the Near InfraRed (NIR) domain. In addition, Hengfoss *et al.* [18] analysed extensively the multi-spectral signatures of living against the cadaver fingers using spectroscopy techniques for wavelengths between 400 nm and 1630 nm. However, no PAIs were analysed in their work.

In contrast to the NIR wavelengths under 850 nm used in most of the aforementioned articles, it has been shown that human skin shows characteristic remission properties for multi-spectral SWIR wavelengths, which are independent of a person's age, gender or skin type [19]. Based on this principle, Steiner *et al.* described in [15, 20] a new skin detection approach based on the spectral signatures extracted from wavelengths ranging from 935 nm to 1550 nm. The authors trained a system based on a Support Vector Machine (SVM) to discriminate skin vs. non-skin (e.g., hair or make-up) pixels, thereby enabling a fast classification process. In their experiments on facial images partially covered with masks, hair or make-up, they achieved a pixelwise classification accuracy over 99.9%.

Inspired by the skin detection method presented in [15, 20], we propose a PAD method based on the spectral signatures of finger samples captured in the range 1200 nm – 1550 nm. To the best of our knowledge, this is the first fingerprint PAD method exploring SWIR imaging. In particular, pixels are classified as skin or non-skin, and a final decision (i.e., bona fide or presentation attack) for the complete sample is made based on the proportion of non-skin pixels detected. We have tested a wide variety of PAI species including both complete thick gummy fingers and more challenging overlays, fabricated with twelve different materials, and successfully detected all materials but one.

It should be highlighted that only six samples where used for training, thus being able to test the remaining six materials as *unknown attacks* (i.e., attacks not seen previously by the classifier, thereby representing a bigger challenge and a better representation of a real-world scenario). Even if large databases comprising thousands of bona fides are freely available, this is not the case for all PAIs. In particular, the LivDet competitions [13] offer large databases comprising bona fides and PAIs. However, only a few distinct PAIs are considered. Therefore, being able to train the system with a small number of samples, stemming from a reduced number of PAIs, is crucial to be able to detect unknown attacks. The pixel-wise classification of the proposed method allows the use of such a small training set.

The rest of the article is organised as follows. The SWIR sensor and PAD method proposed are described in Sect. 2. Then the experimental protocol and the results obtained are presented in Sect. 3. Final conclusions are drawn in Sect. 4.

## 2 Proposed Presentation Attack Detection Method

We describe in this section both the capture device and the PAD method proposed in detail. In particular, we define the normalised spectral signatures for each pixel and the classification algorithm to reach a final bona fide vs. presentation attack decision for the sample at hand.

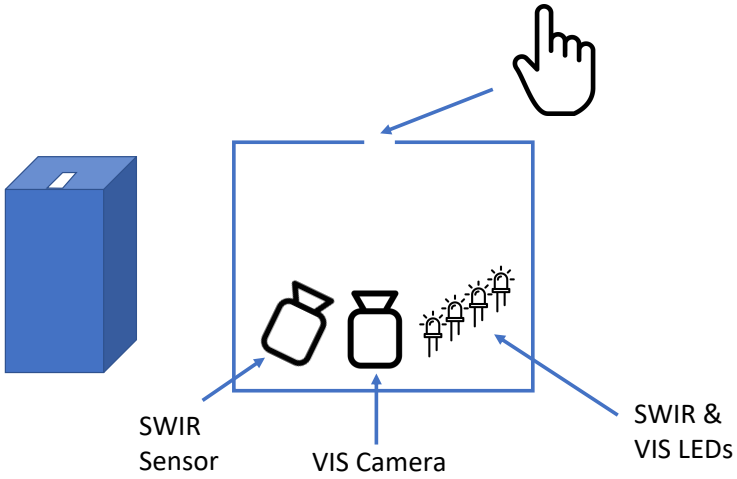
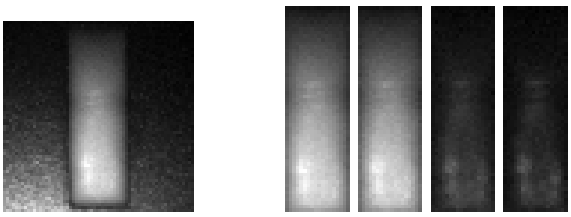



Figure 1: Finger sensor diagram. On the left, the complete box showing the open slot to place the finger for the acquisition. On the right, a diagram of the inner components: two different sensors for the SWIR images and the visible (VIS) light images, together with the corresponding LEDs.

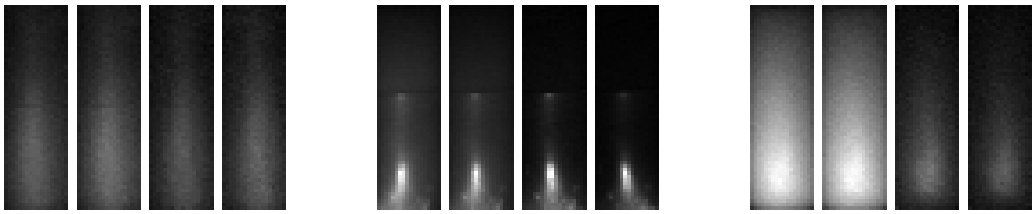


(a) (b) (c) (d) (e)

Figure 2: Bona fide samples: (a) complete image at 1200 nm, and cropped finger-slot regions, where each image was captured at (b) 1200 nm, (c) 1300 nm, (d) 1450 nm, and (e) 1550 nm, respectively.

## 2.1 Short Wave InfraRed (SWIR) Imaging

A diagram of the finger SWIR capturing device developed for the present work is included in Fig. 1. As it may be observed, the camera and lens are placed inside a closed box, which includes an open slot on the top. When the finger is placed there, all ambient light is blocked and therefore only the desired wavelengths are used for the acquisition. In particular, we have used a Hamamatsu InGaAs SWIR sensor array, which captures $64 \times 64$ px images, with a 25 mm fixed focal length lens optimised for wavelengths within $900 - 1700$ nm. In this article, we have considered the following SWIR wavelengths: 1200 nm, 1300 nm, 1450 nm, and 1550 nm, similar to the wavelengths considered in [15, 20] for the skin vs. non-skin facial classification. An example of the acquired images for a bona fide sample is shown in Fig. 2a for the 1200 nm wavelength. In addition, fingerprint verification can be carried out with contactless finger photos acquired in the visible spectrum with a 1.3 MP camera and a 35 mm VIS-NIR lens, which are placed next to the SWIR sensor within the closed box (see Fig. 1).



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

Figure 3: Cropped finger-slot regions corresponding to several presentation attack instruments: (a) to (d) silicone finger, (e) to (h) dragon skin overlay, and (i) to (l) playdoh finger. In all cases, each image was captured at 1200 nm, 1300 nm, 1450 nm, and 1550 nm, respectively.

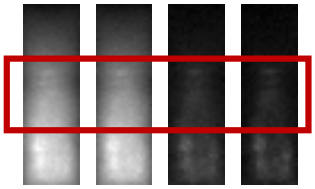


Figure 4: Selected ROI comprising the central $18 \times 20$ px from the images acquired at each wavelength.

In order to process the images, the central finger-slot region corresponding to the open slot where the finger is placed needs to be extracted from the background. Given that the finger is always placed over the open slot, and the camera does not move, a simple fixed size cropping can be applied. The corresponding bona fide samples for all SWIR wavelengths, with a size of $18 \times 58$ px, are depicted in Fig. 2b to 2e.

Finally, samples acquired from three PAIs fabricated with different materials are included in Fig. 3: (a) to (d) a silicone finger, (e) to (h) a dragon skin overlay, and (i) to (l) an orange playdoh finger. Some differences may be observed if we compare the images to those captured from a bona fide presentation in Fig. 2. Whereas for the bona fide, the images show a decrease in the intensity value for bigger wavelengths, this is not the case for the silicone and the dragon skin. Such trend will be hence exploited by the PAD method.

We may also see in both Fig. 2 and 3 that the upper and lower ends of the finger present a non-uniform illumination and reflections. For this reason, the PAD algorithm will only analyse the central part of the images, thus yielding a final ROI comprising $18 \times 20$ px (see Fig. 4).

## 2.2 Spectral Signatures

Similar to [15, 20], the spectral signature **ss** of a pixel with coordinates $(x, y)$ is defined as:

$$\mathbf{ss}\,(x, y) = (i_1, \ldots, i_N) \tag{1}$$

where $i_n$ represents the intensity value for the $n$th wavelength. In our particular case study, $N = 4$.

In order to account for illumination changes, and therefore achieve a signature independent of the absolute brightness of the image at hand, a normalised signature is computed. In addition, in order to capture the distinct trends accross different wavelengths shown in Fig. 2 for the

bona fides (i.e., skin pixels) and in Fig. 3 for the PAIs (i.e., non-skin pixels), a final normalised difference vector is computed as follows:

$$d\left[i_a, i_b\right] = \frac{i_a - i_b}{i_a + i_b} \tag{2}$$

with $a, b \leq N$, $a \neq b$ and $-1 \leq d\left[i_a, i_b\right] \leq 1$. In other words, the normalised differences between all possible wavelength combinations are computed. For our case study with $N = 4$, a total number of six differences are calculated:

$$\mathbf{d}\left(x, y\right) = \left(d\left[i_1, i_2\right], d\left[i_1, i_3\right], d\left[i_1, i_4\right], d\left[i_2, i_3\right], d\left[i_2, i_4\right], d\left[i_3, i_4\right]\right) \tag{3}$$

These normalised difference vectors $\mathbf{d}\left(x, y\right)$ will be used to classify skin vs. non-skin pixels.

### 2.3 Final Classification

The final bona fide vs. presentation attack decision for the sample at hand is made in a two step manner:

- For each pixel with coordinates $(x, y)$, the normalised spectral signature $\mathbf{d}\left(x, y\right)$ is computed and classified as skin vs. non-skin with a Support Vector Machine (SVM) classifier.
- Given that a single sample comprises a total number of $18 \times 20 = 360$ px per wavelength, the final score $s$ returned by the PAD method will be the proportion of non-skin pixels of the sample ROI in a range of 0 to 100, as per [21].

In other words, regarding the final score $s \in [0, 100]$ generated by the PAD algorithm, low values close to 0 will represent bona fide samples and high values close to 100 will denote presentation attacks.

It should be finally noted that, whereas other high performing state of the art methods need a high number of images for training, up to 1000, the present method, being based on a pixelwise classification, requires a very low number of images for training. In our experiments, only six images per class are utilised.

## 3 Experimental Evaluation

The PAD algorithm proposed in Sect. 2 is evaluated in this section. First, the database and evaluation metrics are described. We subsequently present and analyse the results obtained.

### 3.1 Experimental Setup

In the experiments, we have analysed the following twelve different PAIs:

- 3D printed fingerprint and 3D printed fingerprint coated with silver paint (denoted Ag) to mimic the conductive properties of the skin;
- fingers fabricated with blue and green wax, gelatine, playdoh, silly putty and silicone;
- overlays fabricated with dragon skin and urethane;
- fingerprints printed on regular matte paper and on transparency paper.

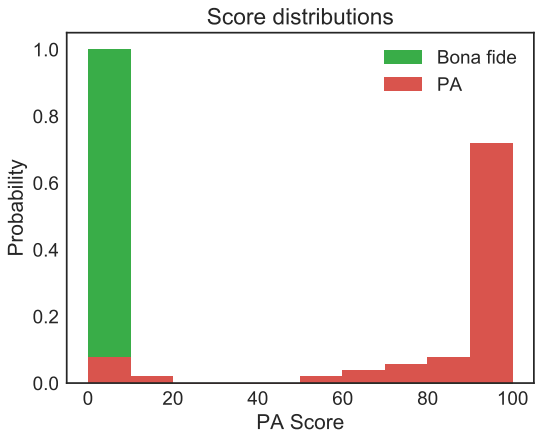



Figure 5: Normalised score distributions for bona fide (green) and PA (red) presentations. As it can be observed, only three scores yielded by PAs overlap with the scores stemming from bona fide presentations.

Regarding the bona fide samples, in order to maximise their variability, they have been captured from all fingers, from the little one to the thumb. For each bona fide and PAI, between two and three samples have been acquired.

Following common practices to achieve a balanced training of the classifier, the same number of samples (six) have been used for each class, namely: bona fides and PAs. For the latter, the following six different materials have been chosen: dragon skin overlay, urethane overlay, playdoh finger, printed fingerprint on matte paper, printed fingerprint on transparency paper and silly putty finger. Since in total twelve different materials have been analysed, the remaining six PAIs can be considered unknown attacks, thereby allowing us to test the robustness of the classifier in this more challenging scenario.

Finally, in compliance with the ISO/IEC IS 30107-3 on Biometric presentation attack detection - Part 3: Testing and Reporting [22], the following metrics are used to evaluate the performance of the PAD method:

- Attack Presentation Classification Error Rate (APCER): percentage of attack presentations wrongly classified as bona fide presentations.
- Bona Fide Presentation Classification Error Rate (BPCER): percentage of bona fide presentations wrongly classified as presentation attacks.

## 3.2 Results

The score distribution for the bona fide presentations (green) and the PA (red) are depicted in Fig. 5. As it may be observed, there is a very small overlap between both distributions: whereas all bona fide presentations achieve scores lower than 10 (i.e., less than 10% of their pixels are incorrectly classified as non-skin), only a single PAI (the playdoh finger), from which three samples have been acquired, achieves such a low score. Therefore, both classes are easily separable.

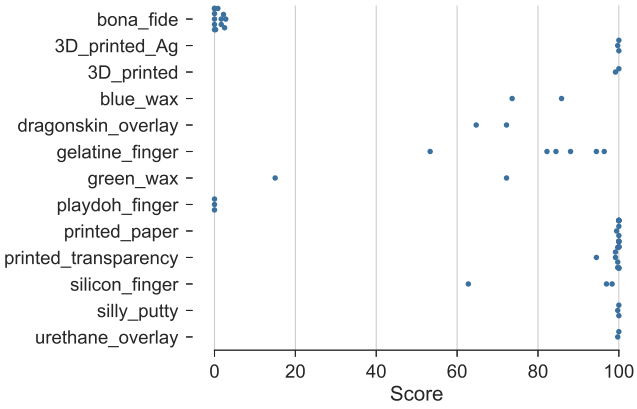
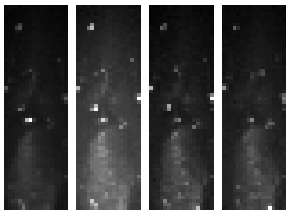



Figure 6: Scores yielded by each presenation, ordered by PAI species or bona fide along the y axis.



(a) (b) (c) (d)

Figure 7: Example of a noisy image for the green wax finger, captured at (a) 1200 nm, (b) 1300 nm, (c) 1450 nm, and (d) 1550 nm.

In order to further analyse the performance of the PAD method, all scores are presented in Fig. 6, classified along the y axis according to their origin: either bona fide or a particular PAI. For most of the bona fide samples, the score is 0 (i.e., all pixels are classified as skin), and all scores lie under 3 (i.e., less than 3% of the pixels are incorrectly classified).

Regarding the different PAIs considered, it may be observed that all playdoh finger samples achieve also a score of 0. This is due to the similarity in the responses to the selected wavelengths with respect to the bona fides, as it may be seen in Figs. 2b to 2e (bona fide) and Figs. 3i to 3j (playdoh PAI). On the other hand, all the remaining PAIs can be discriminated from bona fides with a threshold below 10. Moreover, most of the remaining PAIs yield scores over 80 (i.e., over 80% of the pixels are correctly classified as non-skin). Among the exceptions one of the green wax finger samples shows a remarkably lower score (15 vs. 75 – 85 for the remaining samples). This is due to a noisy acquisition, as shown in Fig. 7. In spite of that fact, the PA is still correctly detected.

In summary, the overall performance of the PAD algorithm for decision threshold on 5 (i.e., at least 5% of the pixels in the ROI are classified as non-skin) yields an APCER = 5.7% for BPCER = 0%, thereby granting a highly secure (low APCER) and convenient (low BPCER) system.

It should be finally highlighted that all unknown PAIs (i.e., both 3D printed fingerprints, blue and green wax fingers, gelatine finger and silicone finger) are correctly detected as PAs. That means the classifier is robust to all previously unseen attacks considered, thereby proving the soundness of the approach. Such success is due to the fact that most materials, excluding the orange playdoh tested, exhibit a similar behaviour to each other for the selected SWIR bands, and at the same time different from the bona fides. We may thus conclude that the proposed sensor and algorithm are able to detect unknown attacks.

# 4 Conclusions

We have presented a novel fingerprint presentation attack detection approach based on spectral signatures extracted from SWIR images. Contrary to other high performing PAD algorithms, since classification is performed in a pixelwise fashion, very few samples are required for training (six bona fides and six PAIs in our experimental evaluation). This allows for a robust training in spite of the lack of freely available databases comprising a wide variety of PAIs.

The experiments have shown that all bona fide samples are correctly classified as such, thus yielding a highly convenient system. On the other hand, all materials except for playdoh are also detected as PAs. Even unknown attacks are correctly detected, thereby showing the promising performance of the proposed method. This is due to the similarities exhibited by most PAIs at the selected SWIR bands, and their difference with respect to the bona fides behaviour.

In addition, given the reduced number of pixels of the final ROI considered (360 px, see Fig. 4), the low dimensionality of the feature vector (6) and the use of an SVM classifier, samples are classified in a fast and efficient manner. We can thus conclude that the proposed PAD method can be utilised for real-time applications.

As future work lines, we will acquire a bigger database, comprising more PAIs and more bona fide samples, in order to further test the performance of the algorithm to both known and unknown attacks. We will also explore other feature extraction techniques to deal with the undetected PAIs (playdoh finger) including both other algorithms trained on the same SWIR data and other acquisition technologies.

# Acknowledgements

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA) under contract number 2017-17020200005. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

# References

[1] A. K. Jain, Technology: Biometric recognition, Nature 449 (207) 38–49.

[2] N. Ratha, J. Connell, R. Bolle, Enhancing security and privacy in biometrics-based authentication systems, IBM Systems Journal 40 (3) (2001) 614–634.

[3] ISO/IEC JTC1 SC37 Biometrics, ISO/IEC 30107-1. Information Technology - Biometric presentation attack detection - Part 1: Framework, International Organization for Standardization (2016).

[4] J. Galbally, M. Gomez-Barrero, Presentation attack detection in iris recognition, in: C. Busch, C. Rathgeb (Eds.), Iris and Periocular Biometrics, IET, 2017.

[5] J. Galbally, M. Gomez-Barrero, A review of iris anti-spoofing, in: Proc. Int. Workshop on Biometrics and Forensics (IWBF), 2016.

[6] E. Marasco, A. Ross, A survey on antispoofing schemes for fingerprint recognition systems, ACM Computing Surveys (CSUR) 47 (2) (2015) 28.

[7] C. Sousedik, C. Busch, Presentation attack detection methods for fingerprint recognition systems: a survey, IET Biometrics 3 (4) (2014) 219–233.

[8] J. Galbally, S. Marcel, J. Fierrez, Biometric antispoofing methods: A survey in face recognition, IEEE Access 2 (2014) 1530–1552.

[9] S. Marcel, M. S. Nixon, S. Z. Li (Eds.), Handbook of Biometric Anti-Spoofing, Springer, 2014.

[10] TABULA RASA, Trusted biometrics under spoofing attacks (2010).
URL http://www.tabularasa-euproject.org/

[11] BEAT, Biometrics evaluation and testing (2012).
URL http://www.beat-eu.org/

[12] ODNI, IARPA, IARPA-BAA-16-04 (thor) (2016).
URL https://www.iarpa.gov/index.php/research-programs/odin/odin-baa

[13] Livdet - liveness detection competitions (2009–2017).
URL http://livdet.org/

[14] Y. Wang, X. Hao, Y. Hou, C. Guo, A new multispectral method for face liveness detection, in: Proc. Asian Conf. on Pattern Recognition (ACPR), 2013, pp. 922–926.

[15] H. Steiner, S. Sporrer, A. Kolb, N. Jung, Design of an active multispectral SWIR camera system for skin detection and face verification, Journal of Sensors 2016.

[16] R. K. Rowe, K. A. Nixon, P. W. Butler, Multispectral Fingerprint Image Acquisition, Springer London, 2008, pp. 3–23.

[17] S. Chang, K. Larin, Y. Mao, W. Almuhtadi, C. Flueraru, Fingerprint spoof detection by NIR optical analysis, in: State of the Art in Biometrics, InTech, 2011, pp. 57–84.

[18] C. Hengfoss, A. Kulcke, G. Mull, C. Edler, K. Püschel, E. Jopp, Dynamic liveness and forgeries detection of the finger surface on the basis of spectroscopy in the 400–1650 nm region, Forensic science international 212 (1-3) (2011) 61–68.

[19] J. A. Jacquez, J. Huss, W. McKeehan, J. M. Dimitroff, H. F. Kuppenheim, Spectral reflectance of human skin in the region 0.7–2.6 $\mu$, Journal of Applied Physiology 8 (3) (1955) 297–299.

[20] H. Steiner, A. Kolb, N. Jung, Reliable face anti-spoofing using multispectral SWIR imaging, in: Proc. Int. Conf. on Biometrics (ICB), 2016, pp. 1–8.

[21] ISO/IEC JTC1 SC37 Biometrics, ISO/IEC DIS 30107-2. Information Technology - Biometric presentation attack detection - Part 2: Data formats, International Organization for Standardization (2017).

[22] ISO/IEC JTC1 SC37 Biometrics, ISO/IEC IS 30107-3. Information Technology - Biometric presentation attack detection - Part 3: Testing and Reporting, International Organization for Standardization (2017).

# Fighting Ransomware with Guided Undo

Matthias Held and Marcel Waldvogel

University of Konstanz, Konstanz, Germany
{matthias.held, marcel.waldvogel}@uni-konstanz.de

**Abstract**

Ransomware attacks are rare, yet catastrophic. On closer inspection, they differ from other malware infections: Given appropriate preparation, they do not need to be detected and prevented, but could be recovered later. However, current ransomware protection follows the beaten path of anti-malware copying their fallacies. We show how the move to personal cloud storage allows for a paradigm shift in ransomware protection: exceptional attack isolation, perfect elimination of false positive alerts, and simplified recovery.

In this paper, we analyze the necessary operations for ransomware, extend existing ransomware taxonomy, and verify them against real-world malware samples. We analyze the costs and benefits of moving ransomware detection to versioned personal cloud storage. Our content, meta data, and behavior analysis paired with a 'guilt by association' capability greatly improve the false positive rate, but the guided undo make this rate all but inconsequential. Even though the user now carries a new burden, it comes with clear responsibilities and benefits, while being freed from questionable duties, resulting in a win-win situation for user experience and detection quality.

## 1 Introduction

Ransomware attacks are most effective when they can strike when no backups of the data exist, and the victim notices the encryption of his personal data after modifying files just a few hours ago. Tempting the victims to pay ransom in the hope of getting relief, and thus financing organized crime or losing all the personal data generating an economic loss for organizations and companies. It is therefore imperative for users and society to reduce the impact of ransomware to a rare event, causing no more than a brief annoyance. Current ransomware detection is integrated with or modeled after other malware protection: The key to defense is software running on the victim's machine, trying to identify malware when it is downloaded or first activated. Identifying the purpose of software from its bytestream is impossible [24], and once the software is running, anti-malware is not more powerful or bug-free than the operating system, whose duty includes protection from malicious software. This model cannot eliminate the problems of malware circumventing the detection software, thus leading to limitations which reduce the identification and recovery quality [23, 22].

Traditional ransomware detection analyzes program code and file operations, among other indicators, to detect potential malware activity. Either of them can result in false positives, triggering the user to react, often with panic. The anti-malware developer's goal therefore is to minimize the number of false positives, resulting in false negatives. If malware goes undetected, the data, even on the backup disk, when connected, may be encrypted or destroyed; an essentially unsolvable dilemma.

A more suitable solution would be an undo operation to the state of the files before the attack, with a backup strategy that cannot be circumvented. The ongoing move to personal cloud storage, however, opens the possibility of a paradigm shift in ransomware detection. Having an up-to-date copy of all important data on an unrelated system where no ordinary user can install or run additional software provides for exceptional attack isolation, perfect elimination

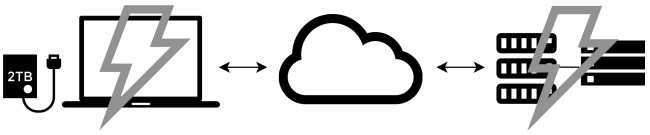



Figure 1: Possible victim machines

of false positive alerts, and simplified recovery. These benefits from attack isolation come at a cost: the weaker coupling to the events on the victim's machine reduces information about ongoing file operations as part of the malware, such as the sequence of actual file operations, and details including file offsets.

The missing real-time information has some impact on *content-based* and *metadata-based* indicators, which may analyze short-term (on the order of seconds) event sequences; however, analyzing long-term *behavior* remains unaffected. We shift the classification towards this indicator, supported by the file versioning to observe file operations without data loss. This shift improves the separation between processes of benign software which work with high entropy files and ransomware attacks.

The independence gained by the weak coupling between victim machine and cloud storage more than offsets the analysis limitations, as the detection software and archived file versions cannot be manipulated by the ransomware.

We propose the following data security model to protect against many incidents, including ransomware attacks (Figure 1): The user's machine (left) does a traditional backup to a locally-attached storage (far left). It also performs a real-time synchronization to a personal cloud storage, such as provided by a personal machine of the Raspberry Pi class or a cloud provider. Important is that the cloud storage be versioned, so that previous, unaffected, states of the data may be restored. As long as at most one of the two systems (user machine, personal cloud) is infected, data can be recovered. In the unlikely event that the cloud storage is the one being infected, synchronization of destroyed data may happen back to the user machine. This is guaranteed not to affect the backup, as it is outside of the synchronization scope.

As a result, unlike most other security incidents, ransomware does not necessarily need to be detected on first sight. Relaxing the requirements by using a delayed detection and recovery approach assisted by the personal cloud storage, allows us to remove the need of having a 100% detection rate without any false positives, something infeasible with real-time detection.

Offering such a model should also bring a usability improvement for the user by simplifying the recovery process by using the proposed indicators to analyse and classify the files and offering an color guidance for the user with an easy-to-use one-click recovery.

This paper makes the following contributions:

1. We define the general problems fighting ransomware using generic indicators.

2. We formulate a taxonomy of requirements for operation classes as well as indicator categories and verify it with real-world malware samples.

3. We define a set of *content-based*, *metadata-based* and *behavior-based* indicators which are a set of already suggested indicators and indicators extended or improved by us.

4. Based on these indicators, we implement a ransomware detection in an app on top of *Nextcloud* cloud storage, which assists making the right selection when recovering from an attack.

## 2 Problems fighting ransomware

Ransomware attacks are rare events. However, is it necessary to analyze every file operation to detect them. Once we use an automatic recovery, we need to have a detection rate without any false positives since a – still so small – false positive rate would lead to many false recoveries due to many file operations performed on a computer system [18, 1].

Perfect protection can be achieved with a system that will not lose any data during a ransomware attack and will always recover the data that was destroyed by ransomware and not removed by the user himself. Educating the user through an easy-to-use recovery, guiding the user with informations gathered by analysing and classifying file operation sequences, will lead to such a mechanism where the user is in control. Since ransomware attacks are so rare, it is easier for the user to take a spare moment to recover if an attack happens instead of taking many moments struggling with false positives.

This change can be compared with the usability step as in the 80s where the move from file-based manual versioning to the simple undo method was established. This reduced the effort for the user drastically. Furthermore, it opened up the file versioning for the group of unsophisticated users by simplifying the recovery. This point is important in companies, universities or organizations which offer a "bring your own device (BYOD)" integration, where our usability concept can reduce the effort of the IT-support by allowing the users to perform the recovery by themselves [5].

This approach should not just be seen as another step in the arms race between defenders and attackers. Using backups on an autonom system is only existing solution which can be merely circumvented with tremendous effort: Automated simultaneous attacks of two related autonom systems with the goal to disable the backup system and encrypt the data. Extending this solution with our easy-to-use recovery improves it drastically by educating the user and allowing him to quickly restore all changes. Bypassing this recovery by exploiting the insight in our metrics would lead to additional effort, we might assume that this is comparable with the popular and well-known *Nigerian scam*. Where the attacker as *Nigerian royalty* asks for money by sending a poorly worded email. Cormac Herley in his paper *Why do Nigerian Scammers Say They are from Nigeria?* showed that not only defenders have false positives also the attackers suffer from false positive. This is a result of the trade off between profit and investment based on the effort performed. Cormac Herley stated that the typos in the emails are a method to reduce the false positives on the attacker side by removing the users who would answer these emails and still not pay the money from the pool of victims – generating unrewarded effort for the attackers – on the first sight leaving only the users who are naively enough to pay the money with or without the typos [7]. Transferring this to our problem allows us to assume that additional effort leads to a worse investment-profit-balance which does not result in an improvement of the ransomware but a more specific targeting of users who do not protect them self.

The generic real-time ransomware detection approaches proposed in different papers offer a real-time detection together with an automatic recovery proclaiming a nearly perfect detection rate with very few false positives [2, 3, 11, 9]. The rarity of a ransomware attack and automatic recovery make theses good detection rates problematic: Assuming we have a ransomware detection software, which monitors every of the 100,000 file operations a day on the file system with a false positive rate of 0.1%. This would result in 100 falsely triggered recoveries leading to 100 unnecessary user interactions reverting these false recoveries. Additionally, the efforts for reducing the false positives lead to a very concrete set of indicators allowing the ransomware authors to build the malware against these indicators avoiding the detection [1].

For blocking decisions, we therefore should include every possible generic property. It is unavoidable to integrate a file versioning mechanism to be able to consider more file operations. This can also be done in real-time and with automatic recovery [3] but is limited by the base rate fallacy.

This cannot be considered reliable due to ransomware circumventing local backup software and detection software by disabling them or overwriting the Master Boot Record to perform the encryption in a state of the pc where only the malware is running [23, 22].

# 3 Related work

Existing ransomware indicators were used in different attempts to detect malicious software, while having several limitations like false positives in form of benign software, which fulfilled the same indicators e.g. high entropy files created by compression tools [17] or bypassing the monitoring by avoiding the indicators thresholds or hiding the malicious process [3, 8]. Long-term observations of ransomware showed their proceeding and possible indicators to prevent zero-day attacks of ransomware [10]. This already constructed a wide-spread set of indicators for detecting ransomware. Possible improvements for the indicators were evaluated: Such as the analysis of the entropy of files to improve it by being able to distinguish between encryption and compression, which are both high entropy files, without leading to a satisfying result [4].

Additionally, the long-term observation *Symantec* published several reports with information and statistics of ransomware over the last several years showing the spreading, behavior, attack methods and how to prevent attacks [19, 21, 20, 16].

Also other researchers published in-depth analyses of ransomware offering informations to many different families [15, 12].

Independent from such indicators, there are techniques like controlling the access to the command-and-control servers to prevent the execution of the ransomware [2] and the detection and escrowing of the encryption key to later decrypt the files without having to pay the money [11].

These indicators were already used to build techniques for detecting and preventing ransomware in real-time [8, 9, 11, 17]. In addition, there is a concept based on sequences and a backup strategy but it acts on the local system without the need of interaction of the user [3]. Limitations of this approach are mainly the bypassing of the monitor by tampering the Kernel or multiprocess malware.

Additionally, the base rate fallacy is a huge problem due to the rarity of ransomware attacks with just a few true positives and potentially many false positives due to plenty non-ransomware-specific file operations leading to a burden for the user [18, 1].

Our work is unlike the real-time approaches and can be described as *delayed detection and recovery*. It presents a method using a personal cloud storage by providing easy-to-use recovery from ransomware attacks including taking the users into control of the recovery. The need for such a concept was also indicated by a support page in the *Dropbox* help center [5]. They tackle this problem by requesting the user to recover every single file by hand or sending a list of encrypted files to the *Dropbox* support.

This approach is based on the personal cloud storage *Nextcloud* and utilizes the integrated file versioning and trash bin methods [13, 14]. The integrated methods use additional logical file storages or directories to backup modified files with database links adding the necessary informations of the file version.

# 4 Background: Classes

Ransomware activity is not a static set of operations, which is performed in the same way in every ransomware family [10, 15, 12, 16]. The reasons for this are on one hand the race between security researchers and ransomware programmers to detect and hide the activities by changing the operations, and on the other hand the huge code base of existing ransomware to copy from. This leads to several implementations and many different families. Based on our analysis of ransomware operations, we also believe that some of the unusual, inefficient behavior we have seen is due to the high division of labor or just due to oversight or stupidity of the ransomware developer.

The process of constructing an efficient and working way to detect ransomware includes formulating classes for the different types of ransomware. These will then be used to define indicators.

The classes in this section are based on the behavior of existing ransomware families and possible – not yet existing – implementations. For a better overview, the ransomware classes are separated into the following subclasses: **File destruction**, **operation sequence**, **file type funnelling** and **Entropy funnelling**. In addition, the **operation sequence** classes are described for the local file system and the cloud storage.

A ransomware belongs to each of these subclasses and fulfils only one of the properties of the according subclass.

## 4.1 File destruction

The main behavior is the encryption of user files, therefore the ransomware must read the data, write the data and remove the original data afterwards.

This behavior is implemented in various ways by different ransomware variants. The diverse implementations can be classified in two classes:

**Class A** Replaces the data of a file *in-place*, thus it first reads the data from the file then encrypts it, writes the encrypted data back to the file, and closes it. Afterwards, the file is optionally renamed.

**Class B** Creates a new file and moves the data to the new file by reading the content from the original file, encrypting it and writing the encrypted data to the new file, deleting the original file after closing the files.

Existing literature classifies ransomware in three classes, where **Class A** and **Class B** remain the same [17]. They additionally introduced a class, which is an extension of **Class A**, moves the files in an extra folder before encrypting the files *in-place* and then moving them back. We treat this class as a subclass of **Class A**, because the main indicator for this class is the *in-place* encryption and the movement of the files is an unnecessary extension, which does not change the fundamental characteristics.

## 4.2 Operation sequence

The file destruction can be performed in different operation sequences where every sequence is unlike the others due to the order of operations. The ordering can change with every system hence we divide the operation sequence classes into two different scenarios: The operations performed on a local file system and the operations performed during a synchronization process with a cloud storage.

The differentiation between those two scenarios have the simple reason, that the synchronization is performed by a client software, which reorders the file operations that happen during a synchronization interval leading to different operation orders.

Thus the classes for the local file system are defined as following:

**Batch:** Batch WRITE followed by batch DELETE/RENAME-with-overwrite.
**Interleaved:** Interleaved WRITE – DELETE/RENAME-with-overwrite.
**In-place-overwrite:** In-place-overwrite

The classes for the file operations visible during the synchronization are defined like this:

**Batch:** Batch WRITE followed by batch DELETE/RENAME.
**Inversed batch:** Batch DELETE/RENAME followed by batch WRITE.
**Interleaved:** Interleaved WRITE – DELETE/RENAME.
**Chaos:** Chaos WRITE – DELETE/RENAME.

Class **Inversed batch** represents a regroup of class **Batch** and class **Chaos** depicts a regroup and arbitrary split of class **Interleaved**. As mentioned in the previous section the operations can be preceded by first moving the files to a special directory.

### 4.3 File type funnelling

Other options include whether the file extensions of encrypted files are random strings, a ransomware-specific extension, or the file type is unchanged from the original file. Similar things may also happen to the file name [6].

File type funnelling describes the process of reading many files with known file extensions and writing files, which can be assigned to the following classes:

1. All file extensions are unknown and the same.
2. All file extensions are unknown and every extension is distinct.
3. All file extensions are unknown and mixed.
4. All file extensions are known, but files are corrupted.

We define "known" in this enumeration as: File extensions which are known well or file extensions with well-known file types. Additionally, class 3 and 4 are possible but so far non-existing implementations of file type funnelling.

### 4.4 Entropy funnelling

All ransomware families encrypt the data of the user files to extort the user. Hence the entropy of the files is increased – except if all the original files were compressed or also encrypted – thus all current ransomware families show the property of entropy funnelling, where the entropy level of all files is changed to become nearly the same.

In theory, there are options to reduce the entropy of the files written or have them stay nearly the same. Such malware has not been described nor seen in the wild.

## 5 Background: Indicators

The discussed indicators in this section capture the destructive properties of ransomware regarding a single file or a sequence of files and not modifications of the operation system. The

described indicators were chosen to be general enough to work for all ransomware variants and try to ignore specific identifiers like specific strings or network traffic to the command-and-control server.

We divide these into three groups of *content-based*, *metadata-based* and *behavior-based* properties, where the *content-based* indicators capture the properties of the data, *metadata-based* indicators express the properties of the file and the *behavior-based* ones describe the properties of the sequence. They are described in more detail in [6].

Most of these indicators are well discussed in [8, 17, 3]. They also showed that the single indicator is not able to determine between benign software and ransomware, but several of those indicators together are able to do so.

The goal of the following chapter is to improve the significance of the Shannon Entropy and to append yet not discussed indicators, which are described in operation timing and operation quantities together with size quantities.

## 5.1 Content-based

### 5.1.1 Shannon entropy

The Shannon entropy is a measurement of information in a file. Files with a high entropy are compressed and encrypted files, where the information level is reduced to gain a specific property of compression or encryption. Thus a ransomware attack should change the entropy of files to a higher average due to the encryption of files.

The definition of the Shannon entropy for a byte array of 256 bytes is as follows:

$$e = \sum_{i=0}^{255} P_{B_i} log_2 \frac{1}{P_{B_i}}$$

for $P_{B_i} = \frac{F_i}{totalbytes}$ and $F_i$, the number of instances of byte value $i$ in the array. This sum expresses a value between 0 and 8, where 8 is the perfectly even distribution of the byte values in the array [17].

This is an indicator for malware files, as mentioned encrypted files have a nearly 8 distribution. Compression also tends to a nearly 8 distribution. Therefore, we analysed high entropy files with the goal to find a method to distinguish between encrypted and compressed files.

### 5.1.2 Improved Shannon entropy

To reduce the false positives of compressed files we use the standard deviation of the entropy of the data to distinguish between compressed and encrypted files. This method is based on the differences in the variance of high entropy blocks in the file data which are depicted in the box plots 3 an 2 of file encrypted with *AES* and files compressed with *Deflate*. The box plots are asymptotic because with larger data blocks the entropy of the blocks approaches the expected value and therefore reduces the standard deviation and larger blocks compensate areas with low entropy better than smaller blocks.

We use the property that the entropy of data blocks of encrypted files are very dense distributed, where in contrast the entropy of data blocks of compressed files is not very dense distributed.

This property is depicted by the standard deviation if we cut the data of the file in 256 bytes blocks and calculate the entropy of every block and the standard deviation of the entropy. The comparison of the standard deviation can be examined in figure 4. We choose a threshold of

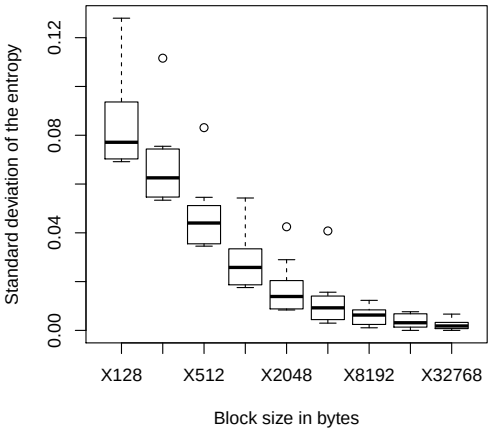



Figure 2: Boxplot of the standard deviation of files compressed with *Deflate*.

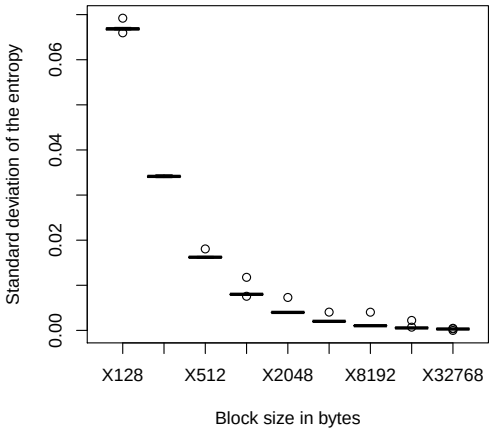



Figure 3: Boxplot of the standard deviation of files encrypted with *AES*.

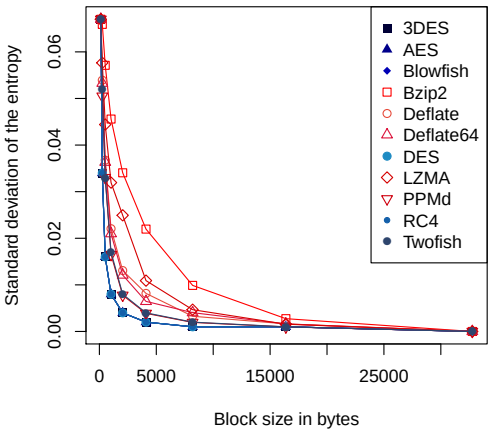



Figure 4: Comparison between the compressed files in red and encrypted files in blue.

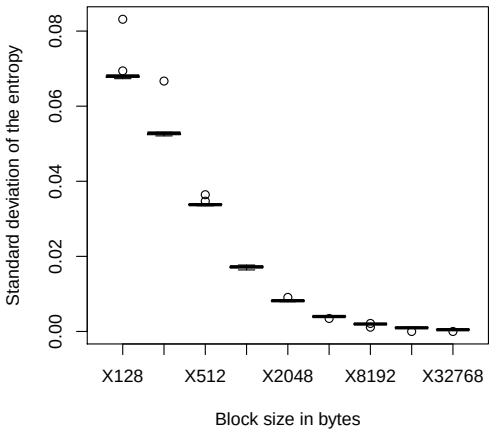



Figure 5: Boxplot of the standard deviation of files compressed with *LZMA*.

0.06 for a data block size of 256 bytes for differentiating between encrypted and compressed files. The classification of this method does have a false positive rate of 10% but improves the indicator of the entropy of the data a lot due to the usage of a guilt by association presumption.

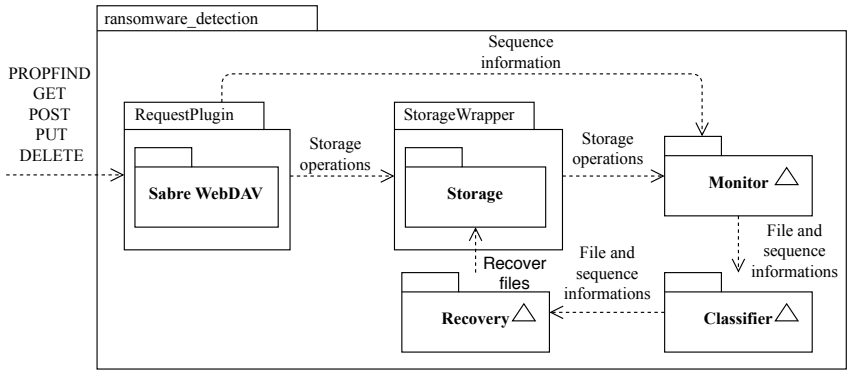



Figure 6: Architecture and integration of the ransomware detection application, which separates the concerns of monitoring, classifying and recovering files and file sequences.

The false positive rate results from compression algorithms that have a very dense distribution with less outliers e.g. *LZMA* (see figure 5). In contradiction, all evaluated encryption algorithms generated a dense distribution hence we have less false negatives. Files classified as encrypted are more suspicious then other high entropy files thus we increase the suspicious classification of the file.

This means we have only a few false negatives and some false positives: If we classify a sequence we can use the 'guilt by association' conclusion to assure our suspected case.

# 6 Application implementation

The ransomware detection implementation is based on the personal cloud storage *Nextcloud*. The reason for this storage are the good file versioning and trash bin methods, which are integrated into this cloud storage and are used here to recover from a detected ransomware attack.

The functionality and limitations of the underlying file versioning and trash bin methods of *Nextcloud* will be described in the limitations.

To describe the integration into the *NextCloud* server, we describe the monitoring of file operations in the first subsection, followed by the integration of the sequence analysis. The last subsection describes the recovery method which uses the integrated file versioning and trash bin of *Nextcloud*.

## 6.1 Monitoring

To monitor all file operations, the data storage of *Nextcloud* is wrapped, piping all file operations through our analyser, gaining the needed operations by analysing the operations with indicators defined in section 5.

## 6.2 Sequence analysis

The sequence analysis puts file operations together to a sequence by looking for time intervals where nothing happens. In the context of *Nextcloud* this is equivalent with synchronization

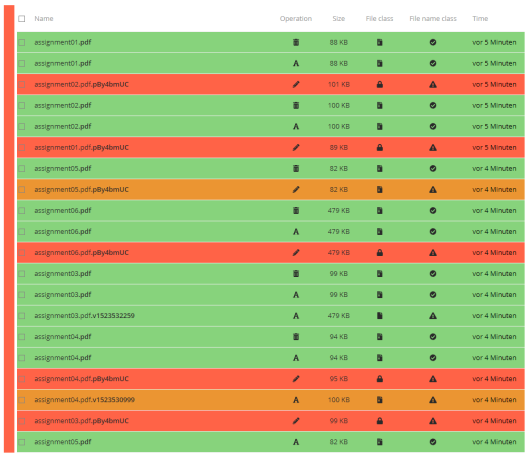



Figure 7: Colouring of a sequence of file operations created by the ransomware *GoldenEye*.

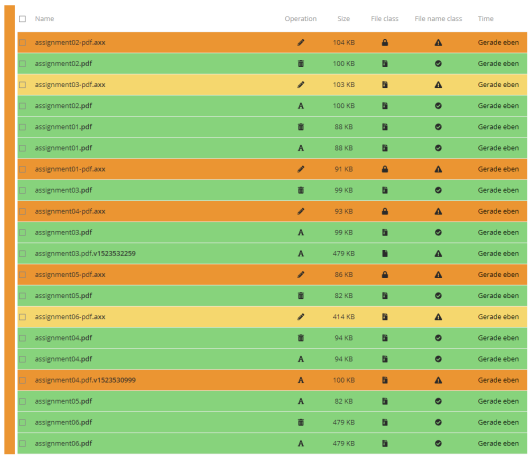



Figure 8: Colouring of a sequence of file operations created by the file encryption tool *AxCrypt*.

requests with no changes. Thus we are detecting all synchronization requests of a client without file storage access and if there is a succession of six synchronization requests without changes, we start a new sequence of file operations. The succession of six synchronization requests conforms three minutes where no changes are synchronized. This time interval is relevant for ransomware families (e.g. *WannaCry*) which work by batch writing all files and afterwards batch deleting all files and would lose the correct classification if the sequence of file operations would be split. During our evaluation, *WannaCry* delayed the file deletion by 10 to 90 seconds depending on the host system, therefore we defined a reasonably long enough time interval which reflects a break of file operations. However, if this threshold fails the user, in the situation of a ransomware attack, should nonetheless be able to find the split sequence easily by looking for the files which he knows were encrypted and recover both with an additional operation.

### 6.3 Classification

The results of the analysis performed during the monitoring are used to classify file operations and sequences of them. These results are used in recovery interface to give the user a color guide, which leads to the following depiction of file operation sequences (see figure 7, 8, 9 and 10).

### 6.4 Recovery

To recover from ransomware attacks the selected sequence of file operations will be reverted; that means new created files will be deleted and deleted or renamed files will be restored. This is done by using the integrated file versioning methods of *Nextcloud*.

## 7 Experiments

In this section the application is reviewed in different aspects to evaluate the functionality and effectiveness. Therefore the classification of the sequences and the effectiveness of the indicators

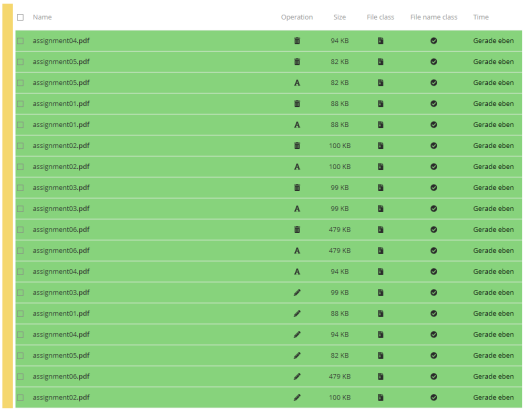



Figure 9: Colouring of a harmless sequence of file operations created by manual batch DELETE/RENAME followed by batch WRITE.


Figure 10: Colouring of a harmless sequence of file operations created by writing many files.

| Family | File suspicion score | Quantities | File type funnelling | Entropy funnelling | Sequence suspicion score |
|---|---|---|---|---|---|
| BTCWare | 0.88 | 2.00 | 2.00 | 2.00 | 6.88 |
| Cerber | 0.75 | 2.00 | 2.00 | 2.00 | 6.75 |
| Evasive | 0.83 | 2.00 | 2.00 | 2.00 | 6.83 |
| Evader | 0.50 | 2.00 | 2.00 | 2.00 | 6.50 |
| GlobeImposter | 0.63 | 2.00 | 2.00 | 2.00 | 6.63 |
| WannaCry | 0.81 | 2.00 | 2.00 | 2.00 | 6.81 |
| Mamba | 0.79 | 2.00 | 2.00 | 2.00 | 6.79 |
| GoldenEye | 0.93 | 2.00 | 2.00 | 2.00 | 6.93 |
| **Median** | **0.63** | **2.00** | **2.00** | **2.00** | **6.63** |

Table 1: A list of ransomware families and their *behavior-based* indicator values with the sequence suspicion score.

are evaluated.

## 7.1 Classification

The sequence classification was tested with a dataset of multiple ransomware families which were submitted to *VirusTotal* in the last months.

The table shows the weighted sum of the *content-based* and *metadata-based* indicators notated as file suspicion score, followed by the *behavior-based* indicators. The sequence suspicion score is the sum of all indicators and the file suspicion score, for more details see [6].

For every of those samples the classification reached a critical sequence suspicion score leading to a highly suspicious sequence.

*AxCrypt* also reached a higher suspicion level compared to the other benign processes. The reason for this is that this is file encryption software which has the same behavior as ransomware.

## 7.2 Indicator effectiveness

Comparing the indicators and their scores between the benign software and the ransomware we notice that the quantities and the file type funnelling clearly separate them. Additionally, the

| Program | File suspicion score | Quantities | File type funnelling | Entropy funnelling | Sequence suspicion score |
|---|---|---|---|---|---|
| Batch image resize | 0.25 | 0.00 | 0.00 | 0.00 | 0.25 |
| Application update process | 0.25 | 0.00 | 0.00 | 0.00 | 0.25 |
| AESCrypt | 0.72 | 0.00 | 2.00 | 2.00 | 4.72 |
| AxCrypt | 0.68 | 1.00 | 2.00 | 2.00 | 5.68 |
| WinRAR compress | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| WinRAR decompress | 0.38 | 0.00 | 0.00 | 0.00 | 0.38 |
| Zip compress | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| Zip decompress | 0.38 | 0.00 | 0.00 | 0.00 | 0.38 |
| Git | 0.25 | 0.00 | 0.00 | 0.00 | 0.25 |
| **Median** | **0.25** | **0.00** | **0.00** | **0.00** | **0.25** |

Table 2: A list of benign application and their *behavior-based* indicator values additionally with the sequence suspicion score.

file suspicion level of benign software is lower but it is only useful in the union with the other indicators due to the classification of the file name, file extension and file class which can be easily faked.

In contrast the other two *behavior-based* indicators are less burdened with false positives, since it is much harder to hide those operations from the monitor, and capture multiple properties which are significant for ransomware.

# 8 Usability

To evaluate the usability of our approach we performed an online survey without any limitations for the participations. The reason for this is that our approach tries to be usable for everybody.

We had 30 participants between the age of 20 and 46 years with 57,1% being male and 42,9% being female.

The goal was to evaluate the usability of our approach compared to restoring all files file-by-file with the use of the trash bin of a personal cloud storage and in addition the support of color guidance for detection ransomware attacks.

The survey was structured as following: Firstly, we collected personal information and the foreknowledge followed by the preference about single file recovery or sequence recovery by asking simple multiple choice questions after describing the scenario to the participants. Thirdly, we asked the participants to decide, which of the given three sequences depicts a ransomware attack. This was done for three sets of three sequences each with and without color guidance. For each set of sequences without color guidance the colors were removed and was presented to the user before presenting the sets with color guidance. To assist the participants spotting the ransomware attacks, two behavior characteristics were given before displaying the sets of sequences. Each set was presented alone and the decision was collected directly afterwards. The sequences were displayed as screenshot similar to figures 7, 8, 9 and 10. Afterwards, the feedback and criticism of the color guidance and the whole approach was collected with a text field and we also asked – for participants who preferred single file recovery over sequence recovery – if they changed their mind about.

50% did not know something about ransomware before taking the survey but two did not answer this question. Asked if the participants would prefer to recover all files file-by-file or as a sequence of files – 96,7% preferred the sequence recovery. Reasons for this were the easy usage and the quicker approach but it was also demanded to be able to select single files. The one person preferring single file recovery over sequence recovery changed their mind after learning about the sequence recovery.

The participants correctly recovered 52% of the ransomware attacks without guidance com-

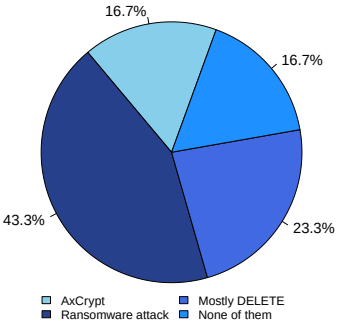



Figure 11: Unguided sequences classified as ransomware attack for sequence set 1.

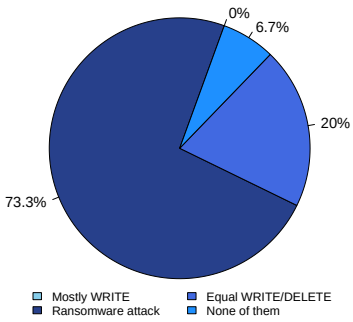



Figure 12: Unguided sequences classified as ransomware attack for sequence set 2.

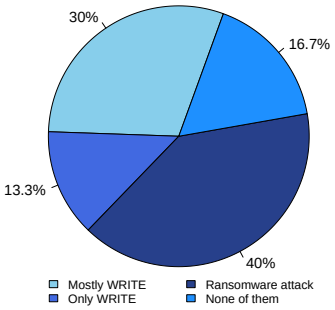



Figure 13: Unguided sequences classified as ransomware attack for sequence set 3.

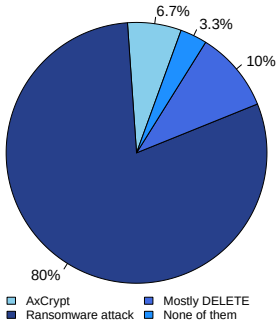



Figure 14: Guided sequences classified as ransomware attack for sequence set 1.

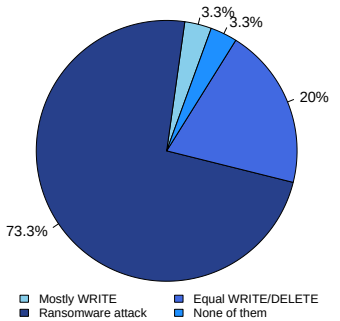



Figure 15: Guided sequences classified as ransomware attack for sequence set 2.

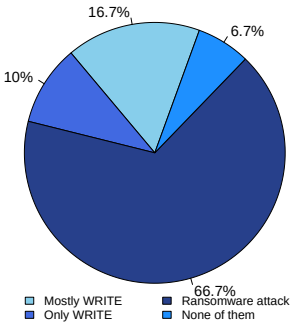



Figure 16: Guided sequences classified as ransomware attack for sequence set 3.

pared to 73% correct recovered ransomware attacks with guidance. Thus we have an increase by 21% with the help of the color guidance. This is also confirmed by 73% of the participants who said that the color guidance helped them spotting the ransomware attacks quicker and easier due to the clear color choice and the direct comparison. One person was confused by the guidance and the approach as whole. Two participants were not able to spot the ransomware attack neither without guidance nor with the guidance and did not perceive the guidance as simplification.

Without the guidance the ransomware attack must fulfil one of the both given behavior characteristics very clearly to be classified as ransomware attack by the participants. If this is not given or a sequence looks similar, the percentage of the correctly as ransomware classified sequence by the participants is only half (see figures 11, 12 and 13).

In contrast the sequences with guidance were classified correctly in over 66% of all questions (see figures 14, 15 and 16).

The foreknowledge does not help the user when it comes to spotting the ransomware attacks. We assume the reason for this is that many reports about ransomware do explain the basic behavior but the most readers cannot transfer their knowledge into a concrete process on a computer.

Conclusively, the recovery interface with the file sequences and the color coding are easily adopted and well understandable thus offer a very good usability. The color coding reduces the uncertainty of the user which seem to exist without it. The participants liked the quick recovery of file sequences making the extra amount of work – in contrast to automatic recovery approaches – not problematic.

More details of the survey are given in [6].

# 9 Limitations

Although we extract the false positives by handing over the decision to the educated user guided by the information gathered by our analysis and classification methods, we can separate the limitations in the non-technical user responsibilities and the technical ones.

The non-technical user responsibilities are to be creative, adaptive and sceptical to non-suspicious sequences which can be created with bypassing our technical approaches [18].

Independent from the non-technical user responsibilities, there are some technical limitations, which can be used to evade our detection mechanisms and the according countermeasures.

## 9.1 Attacking the file classification score

The file classification score consists of three parts: The file name classification, the file extension classification, including the file corruption, and the entropy of the file data.

For every part simulating a normal file takes effect and allows the attacker to keep the suspicion level below the threshold. The simulation of a normal file is simple for the file name and file extension but more extensive when it comes to the data entropy. This could be done by adding garbage data to it. This would lead to additional effort, for adding and removing the garbage data, and would also increase the file size. Hence we may gain additional detection patterns and encrypted files with the same reduced entropy or additional header informations about the entropy reduction if done randomly for every file.

All three bypassing methods are possible and would reduce the file suspicion level drastically but would be detected in the sequence classification score because the attacker has to encrypt many files on the system – optimally all files on the system – to launch an successful ransomware attack.

## 9.2 Attacking the sequence classification score

Attacking the sequence classification score is much more complex then attacking the file classification score since it is not based on *content-based* or *metadata-based* indicators but rather on *behavior-based* indicators. The behavior of ransomware is unique and is hardly changed.

We use four *behavior-based* detection techniques to classify a sequence of file operations: operation quantities and size quantities can be circumvented by delaying operations long enough that the monitor creates a new sequence or by writing additional files. However, delaying

operations are contradictory to the needs of ransomware and increase the risk of being detected in the meantime. This risk is targeted by our approach by defining a long enough time interval between sequences. Although some ransomware families keep the file extension from the victims file, keeping the header informations of the file to bypass the corruption detection is special and would make the encryption and decryption process more complex. Writing additional files or increasing the file size by adding garbage data leads to additional effort because it must be added systematically that it can be removed before decrypting the file. The file type funnelling can be bypassed by using the same file extension and the same header informations as the victim file. To bypass the entropy funnelling the attacker must reduce the entropy of the file. The problems with this were already explained in the last subsection.

All these attacks are possible but are only effective as an union. Implementing only one or two would lead to a lower suspicion but the sequence could still be easily spotted as ransomware attack by the user.

### 9.3 File versioning

File versioning and trash bin implementations in personal cloud storages comprise some advantages in contrast to a classical backup strategy: A cloud storage separates the ransomware host from the file storage helping to guard the detection mechanism and the files. Additionally, we have double existence of the data thus a disruption of the synchronization – be it because of ransomware or by the user – allows us to always recover the files also if it is a selective disruption. Furthermore, people who locally backup regularly can be a victim of backup encryption by ransomware, a disk failure or a loss of the backup [23, 22].

Together with these features there are some negative aspects in terms of storage space, speed and complexity. *Nextcloud* uses a simple file versioning and trash bin functionality – by copying the file to a special directory before performing any changes to it – which can use up to 50% of the user's availably free space but uses clean up methods to reduce the space used. Furthermore, for every synchronized file modification a constant number of additional operations have to be performed to create a file version and backup the file accordingly in the correct storage. This usage of multiple backup storages and file management methods increases the complexity of storing and synchronizing files of the system [13, 14].

Compared to the gain of file versioning and the trash bin, the effort is reasonably small.

### 9.4 Local changes

Changes applied on the local system which are then later encrypted by a ransomware attack can be a problem depending on when the synchronization process takes place: If the synchronization takes place before the attack no data will be lost. If the synchronization takes place after the encryption the changes will be lost. The time interval between the modification of the data and the synchronization is small and so will be the changes. Thus we can assume that the user is currently working on the data and will immediately notice that something is wrong with the files. Therefore the user will be able to recover all the data and manual reapply the changes.

## 10 Conclusions

In this paper, we defined general problems fighting ransomware with generic properties. This included the base rate fallacy and the problem of missing significant generic indicators regarding a single file for blocking them in real-time.

We also described ransomware classes based on the behavior of the file destruction, the operations sequence, the file type funnelling and the entropy funnelling. Following, we formulated ransomware indicators and categorised them into three classes: *Content-based*, *metadata-based* and *behavior-based* ones.

We observed that we are able to separate encrypted from compressed files on the basis of the standard deviation of the entropy once we split the file in data blocks, allowing us to increase the validity of the entropy indicator.

Justified through the base rate fallacy of automatic real-time detection and recovery of ransomware attacks, we propose a delayed detection and recovery based on a personal cloud storage with file versioning permitting us to remove false positives and increase the usability by taking the user into responsibility. This approach is supported by the guidance of the classification of the file operation sequence based on the ransomware indicators.

This method is supported by the idea that we have no need to recover from ransomware attacks in real-time as long as we do not lose data in the mean-time, we can increase the quality of the classification and the usability of the recovery for the user.

The usage of a strategy with file versioning opens up the use of *behavior-based* indicators to classify the whole sequence file operations according to a ransomware attack. This strategy reflects an improvement regarding the already proposed classification approaches.

Additionally, the implementation of the ransomware detection on the basis of a personal cloud storage removes the threat of the monitor being attacked and also takes care that there is always a backup of the files.

## 11 Acknowledgements

The authors would like to thank VirusTotal for providing malware samples.

## References

[1] BBC UK. A scanner to detect terrorists. [online], 2009. http://news.bbc.co.uk/2/hi/uk_news/magazine/8153539.stm (accessed March 28th, 2018).

[2] Krzysztof Cabaj and Wojciech Mazurczyk. Using Software-Defined Networking for Ransomware Mitigation: The Case of CryptoWall. *Netwrk. Mag. of Global Internetwkg.*, 30(6):14–20, November 2016.

[3] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. ShieldFS: A Self-healing, Ransomware-aware Filesystem. In *Proceedings of the 32Nd Annual Conference on Computer Security Applications*, ACSAC '16, pages 336–347, New York, NY, USA, 2016. ACM.

[4] Craig from /dev/ttyS0. Differentiate Encryption From Compression Using Math. [online], 2013. http://www.devttys0.com/2013/06/differentiate-encryption-from-compression-using-math/ (accessed March 28th, 2018).

[5] Dropbox Inc. What to do if your files were corrupted or renamed by ransomware. [online], 2017. https://www.dropbox.com/help/security/ransomware-recovery (accessed March 28th, 2018).

[6] Matthias Held. Detecting ransomware. Master's thesis, University of Konstanz, Konstanz, Germany, 2018.

[7] Cormac Herley. Why do nigerian scammers say they are from nigeria? *WEIS*, June 2012.

[8] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 757–772, Austin, TX, 2016. USENIX Association.

[9] Amin Kharraz and Engin Kirda. Redemption: Real-Time Protection Against Ransomware at End-Hosts. In *RAID*, 2017.

[10] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks. In *Proceedings of the 12th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment - Volume 9148*, DIMVA 2015, pages 3–24, New York, NY, USA, 2015. Springer-Verlag New York, Inc.

[11] Eugene Kolodenker, William Koch, Gianluca Stringhini, and Manuel Egele. PayBreak: Defense Against Cryptographic Ransomware. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 599–611, New York, NY, USA, 2017. ACM.

[12] Vadim Kotov and Mantej Singh Rajpal. Understanding Crypto-Ransomware: In-Depth Analysis of the Most Popular Malware Families. Technical report, Tech. rep. Bromium, 2014.

[13] Nextcloud GmbH. Nextcloud 13 user manual. [online], 2018. https://docs.nextcloud.com/server/13/user_manual/files/version_control.html (accessed June 4th, 2018).

[14] Nextcloud GmbH. Nextcloud 13 user manual. [online], 2018. https://docs.nextcloud.com/server/13/user_manual/files/deleted_file_management.html (accessed June 4th, 2018).

[15] Segun I Popoola, Ujioghosa B Iyekekpolo, Samuel O Ojewande, Faith O Sweetwilliams, and AA Atayero. Ransomware: Current Trend, Challenges, and Research Directions. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, pages 169–174, 2017.

[16] Kevin Savage, Peter Coogan, and Hon Lau. The evolution of ransomware. *Symantec, Mountain View*, 2015.

[17] N. Scaife, H. Carter, P. Traynor, and K. R. B. Butler. CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 303–312, June 2016.

[18] Bruce Schneier. *Beyond Fear: Thinking Sensibly About Security in an Uncertain World*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[19] Symantec Corperation. Internet Security Threat Report: Ransomware and Businesses 2016. [online], 2016. https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/ISTR2016_Ransomware_and_Businesses.pdf (accessed March 28th, 2018).

[20] Symantec Corperation. Internet Security Threat Report: Volume 22. [online], 2016. https://www.symantec.com/content/dam/symantec/docs/reports/istr-22-2017-en.pdf (accessed March 28th, 2018).

[21] Symantec Corperation. Internet Security Threat Report: Ransomware 2017. [online], 2017. https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-ransomware-2017-en.pdf (accessed March 28th, 2018).

[22] Symantec Security Response. BadRabbit: New strain of ransomware hits Russia and Ukraine. [online], 2017. https://www.symantec.com/connect/blogs/badrabbit-new-strain-ransomware-hits-russia-and-ukraine (accessed March 28th, 2018).

[23] Symantec Security Response. Petya ransomware outbreak: Heres what you need to know. [online], 2017. https://www.symantec.com/blogs/threat-intelligence/petya-ransomware-wiper (accessed March 28th, 2018).

[24] Alan Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230265, 1937.

# Source Code Patterns of Cross Site Scripting in PHP Open Source Projects

Felix Schuckert[12], Max Hildner[1], Basel Katt[2], and Hanno Langweg[12]

[1] HTWG Konstanz,
Department of Computer Science,
Konstanz, Baden-Württemberg, Germany
felix.schuckert@htwg-konstanz.de
hanno.langweg@htwg-konstanz.de
maxhildner@fastmail.com
[2] Norwegian University of Science and Technology,
Department of Information Security and Communication Technology,
Gjøvik, Norway
basel.katt@ntnu.no

**Abstract**

To get a better understanding of Cross Site Scripting vulnerabilities, we investigated 50 randomly selected CVE reports which are related to open source projects. The vulnerable and patched source code was manually reviewed to find out what kind of source code patterns were used. Source code pattern categories were found for sources, concatenations, sinks, html context and fixes. Our resulting categories are compared to categories from CWE. A source code sample which might have led developers to believe that the data was already sanitized is described in detail. For the different html context categories, the necessary Cross Site Scripting prevention mechanisms are described.

## 1 Introduction

Cross site scripting (XSS) is on the fourth place in Common Weakness Enumeration (CWE) top 25 2011 [3] and on the seventh place in Open Wep Application Security Project (OWASP) top 10 2017 [4]. Accordingly, Cross Site Scripting is still a common issue in web security. To discover the reason why the same vulnerabilities are still occurring, we investigated the vulnerable and patched source code from open source projects. Similar methods, functions and operations are grouped together and are called source code patterns. Our work shows which source code patterns occur in real life projects, to provide a data set that can be compared to existing vulnerability data sets like SAMATE SARD [8]. The questions to be answered are: How do source code patterns from real projects look like? What main protection mechanisms are required to prevent Cross Site Scripting attacks? Real source code samples from open source projects are investigated to get answers to these questions.

This paper begins with an overview of related work in section 2. Section 3 explains how we got the source code sample and how the manual review process looks like. The next section explains what Cross Site Scripting categories from CWE [3] exist and how they fit into source code pattern categories. In section 5 our taxonomy resulting from the manual review process is explained. The sample from CVE-2012-5163 described in section 6 is a sample where developers might thought that data is already sanitized. In section 7 the results are discussed. The final section provides some discussion about how our result can be used in the future.

## 2 Related Work

For SQL injection and Buffer Overflow vulnerabilities, we created the source code patterns with the same method that was used by [23] and [22]. Research projects about general classification of source

code patterns exist. Lerthathairat and Prompoon [15] did research about the classification of source code into bad code, clean code and ambiguous code. They use metrics in source code like comments, the size of the function, et cetera. These metrics were analysed by using Fuzzy logic to determine which category the source code belongs to. Bad and ambiguous code were improved by refactoring. A more security related work is from Hui et al. [14], they use a software security taxonomy for software security tests. They created a security defects taxonomy based on top 10 software security errors from authoritative vulnerabilities enumerations. Their taxonomy is categorized into *induced causes*, *modification methods* and *reverse use methods*. They advise that their taxonomy should be used as security test cases. Stivalet and Fong [24] present a tool that allows to create short code examples including software vulnerabilities. They split up the source code parts into *input*, *filtering* and *sink*. Permutations of these parts will create different examples. All of the examples can be found in the Testsuite 103 within the Software Assurance Reference Dataset [8].

The research also requires data sources, which are used to classify the source code patterns. Massacci and Nguyen [17] researched different data sources for vulnerabilities, e.g. Common Vulnerabilities and Exposures (CVE), National Vulnerability Database (NVD), et cetera. They looked into which data sources were used by other research projects. They also used Firefox as a database for their analysis. Wu et al. [25] use semantic templates created from the existing CWE database [3]. They should help to understand security vulnerabilities. The authors did an empirical study to prove that these semantic templates have a positive impact on the time to completion on finding the vulnerability.

Louw and Venkatakrishnan [16] present a tool Blueprint can be used to defend against Cross Site Scripting attacks. Different Cross Site Scripting variants are explained and how they can be exploited. Another framework from Gupta and Gupta [13] can be used to protect against XSS attacks specialized on HTML 5 web pages. Another approach from Maurya [18] [19] shows how to prevent Cross Site Scripting vulnerabilities on the server side. In their work different security levels require different sanitization approaches. Nadji et al. [20] present different XSS attack scenarios and also suggest on how to prevent these attacks using a combination of server and client protection mechanisms. Another work from Gundy et al. [12] uses a randomized approach to prevent against XSS attacks. As long as the attackers cannot predict the randomization that approach should deny any XSS exploits. In contrast we suggest using the correct standard prevention methods based on the html context to prevent XSS attacks.

# 3 Method

For our research we decided to review vulnerabilities that were found in open source projects. We focus on vulnerabilities that are tracked in the *CVE* [1] database. CVE reports between 2010 and 2016 (seven years) are used as data samples to ensure that developers had sufficient time to patch the vulnerabilities. To get the vulnerable and patched source code related to CVE reports the results from the source code crawler from [22] were used. It checks CVE entries for related GitHub [2] patches and downloads the vulnerable and patched source code. Table 1 shows how many source code samples for different vulnerability types and programming languages are found. We chose PHP as reviewed programming language because it provides the most samples (122) related to Cross Site Scripting. Out of these source code samples 50 CVE reports are randomly selected for a manual review. 50 samples means 1% of all CVE entries related to XSS and 40% of all CVE entries related to XSS and PHP.

Cross Site Scripting vulnerabilities are commonly split into three parts (sources, sanitization and sinks). The sources are methods where data is provided which can be manipulated by a user. Sinks are methods where such data can be harmful. In a XSS vulnerability it will result in scripts that will be executed on the victim's browser. Sanitization methods transform user provided data such that it will not be harmful, if it reaches sinks.

The manual reviews were done as follows. For each CVE report the note entry was looked up. It usually describes which input fields or variables can be used to exploit the vulnerability. Just within the scope these variables/fields are the tainted data sources for the Cross Site Scripting vulnerabilities. If no variable/field is mentioned the source code is manually backtracked from the patched source code

| Categories from CVE details | CVEs | Github | PHP | Java | C/C++ | js | Python | Ruby |
|---|---|---|---|---|---|---|---|---|
| Denial Of Service | 10,929 | 737 | 7 | 3 | 664 | 4 | 8 | 6 |
| Execute Code | 10,647 | 217 | 70 | 6 | 83 | 8 | 12 | 17 |
| Overflow | 6,594 | 302 | 3 | 1 | 280 | 0 | 5 | 1 |
| Obtain Information | 5,471 | 233 | 32 | 6 | 153 | 5 | 10 | 11 |
| Cross Site Scripting | 4,878 | 219 | 122 | 5 | 2 | 46 | 14 | 11 |
| Memory corruption | 3,419 | 52 | 0 | 0 | 49 | 0 | 0 | 0 |
| Bypass a restriction or similar | 2,609 | 96 | 24 | 4 | 51 | 1 | 6 | 2 |
| Gain privileges | 2,207 | 118 | 4 | 0 | 100 | 0 | 0 | 1 |
| SQL Injection | 1,732 | 71 | 53 | 3 | 2 | 2 | 0 | 5 |
| Directory traversal | 1,059 | 35 | 14 | 1 | 7 | 3 | 2 | 5 |
| CSRF | 1,046 | 33 | 25 | 2 | 0 | 6 | 1 | 3 |
| File Inclusion | 100 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| Http response splitting | 74 | 4 | 0 | 0 | 0 | 0 | 0 | 0 |
| Summary | 50,765 | 2,120 | 354 | 31 | 1,391 | 75 | 58 | 62 |

Table 1: Software vulnerabilities in open source software grouped by vulnerability type and programming language.

to find the sources. By doing a data flow analysis from the sources relevant source code patterns are tracked. For example, it will be noted if a sanitization method from a framework is used. For sources, insufficient sanitization, PHP sinks, HTML context and fixes a taxonomy is created based on the review results.

# 4 CWE

Common Weakness Enumeration (CWE) [3] provides categories for software vulnerabilities. CWE-79 is the basic enumeration for Cross Site Scripting. It is distinguished between reflected XSS, stored XSS and dom-based XSS. In a developer perspective these are different sources and sinks depending on when the sanitization should happen. For example, if only sanitized data should be stored in the database, the category can be seen as sink category. On the contrary it can be seen as a source category, if sanitization should only occur before the data is presented on the web page. The CWEs 81 to 87 are different variants of the base CWE-79. These can be seen as different categories for failed sanitization methods and different HTML contexts. The table 2 shows source code pattern categories of the different CWE enumerations.

# 5 Taxonomy of Source Code Patterns

In this section the different taxonomies resulting from the review process are described in detail. The correlations to more or less corresponding CWE categories are mentioned.

## 5.1 Sources

Figure 1 shows three categories for the sources. There are 51 sources because CVE-2014-9270 is a stored and reflected Cross Site Scripting vulnerability. Accordingly, for each type a different source was found (database source and user input source).

| | | |
|---|---|---|
| CWE-81 | Improper Neutralization of Script in an Error Message Web Page | HTML context |
| CWE-82 | Improper Neutralization of Script in Attributes of IMG Tags in a Web Page | HTML context |
| CWE-83 | Improper Neutralization of Script in Attributes in a Web Page | HTML context |
| CWE-84 | Improper Neutralization of Encoded URI Schemes in a Web Page | HTML context |
| CWE-85 | Doubled Character XSS Manipulations | Failed sanitization |
| CWE-86 | Improper Neutralization of Invalid Characters in Identifiers in Web Pages | Failed sanitization |
| CWE-87 | Improper Neutralization of Alternate XSS Syntax | Failed sanitization |

Table 2: CWE Cross Site Scripting variants mapped to categories.

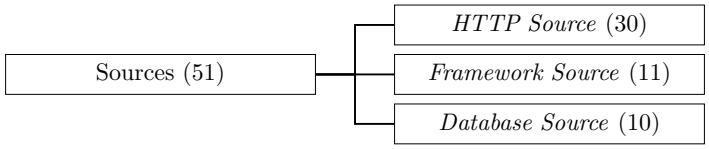



Figure 1: Taxonomy of sources based on the data set.

#### *HTTP Source*

Sources which fall in the *HTTP Source* category are basic HTTP methods provided by PHP. No wrapper was used which might trick developers into thinking that the data might already be sanitized. Sources from this category are related to reflected Cross Site Scripting vulnerabilities. In our data set we found *$_GET*, *$_REQUEST*, *$_POST*, *$_SERVER* and *$_COOKIE*. These are reserved variables from PHP [7]. *$_FILES*, *$_HTTP_RAW_POST_DATA* would also fall into this category, but these were not found in our data set.

#### *Framework Source*

PHP is commonly used with frameworks which provide features to create web pages more conveniently. Methods that wrap around methods from the *HTTP Source* category or methods provided from frameworks to get user provided data fall into this category (e.g. *gpc_get_string()*, *getParam()*). All our samples were internally using sources from the *HTTP Source* category. Accordingly, vulnerabilities with sources from this category are also related to reflected Cross Site Scripting. The extra category was created because such methods might already sanitized the input.

#### *Database Source*

Stored Cross Site Scripting stores the malicious input in the database. That input will be later on presented without any further sanitization on a web page. For our research the methods where user data is returned from the database is seen as a source. For example, in our data set we found functions like *db_fetch_row()* and *serendipity_db_query()*.

## 5.2 Insufficient Sanitization

There is a difference between sanitization methods and escaping methods. Sanitization methods are removing suspicious character which is commonly used to mitigate SQL injection vulnerabilities. In

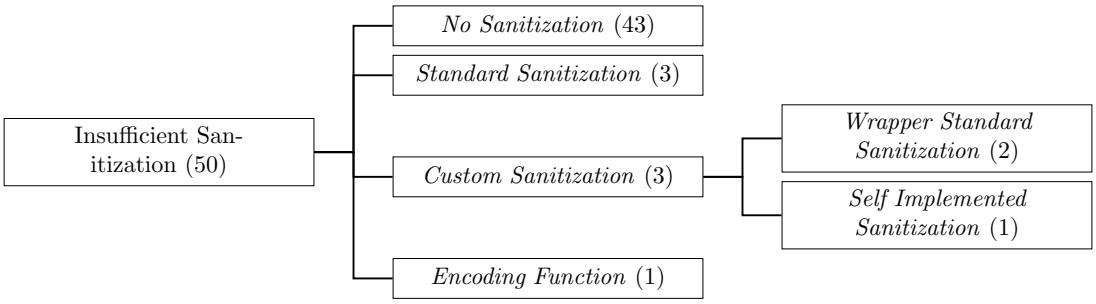



Figure 2: Taxonomy of insufficient sanitization methods based on the data set.

contrast escaping methods are just escaping suspicious character that cannot be used to inject any code. The resulting data from both methods will not be harmful anymore. Escaping methods are more commonly used to protect against Cross Site Scripting attacks because it will not remove characters that an user wants to be displayed. We group both methods together and call them sanitization methods which can either be sanitization methods or escaping methods. Most of our vulnerable samples did not use any sanitization methods. Nevertheless, some did use well known sanitization methods, but Cross Site Scripting vulnerabilities still occurred. Figure 2 shows an overview of the categories for insufficient sanitization methods.

### *No Sanitization*

As the name already indicates, no sanitization method was used. Samples where plain user input will reach a sink fall into this category.

### *Standard Sanitization*

Developers actually used official sanitization methods like *htmlspecialchars()*, but a Cross Site Scripting vulnerability still existed. Why was that sanitization insufficient? Two of the three found samples, the sanitization was insufficient because the HTML sink was from the category *JavaScript Context*. If sinks are from that category, the sanitization has to be more specialized because the context is already in Javascript. How to prevent XSS attacks in such a context is described later on. The last sample did have a special condition where the sanitization method is not used. That sample is described in detail in section 6.

### *Custom Sanitization*

Custom sanitization methods were also found in the data set. Two samples did use wrapper methods which internally use methods from the *Standard Sanitization* category. Accordingly, they should be protected against XSS attacks, but the sinks are again from the *JavaScript Context* category.

One sample did implement a sanitization method using *str_replace()* method. The implementation was insufficient and the patch did fix the vulnerability by using a sanitization method from the category *Standard Sanitization*.

### *Encoding Function*

One sample did use an encoding function. These functions are not supposed to be used as a sanitization method. Nevertheless, as seen in the fixes categories, some developers patched the vulnerabilities by using encoding functions. This just prevents Cross Site Scripting vulnerability as long the context is from the *Plain HTML Context* category and the correct HTML encoding function is used.

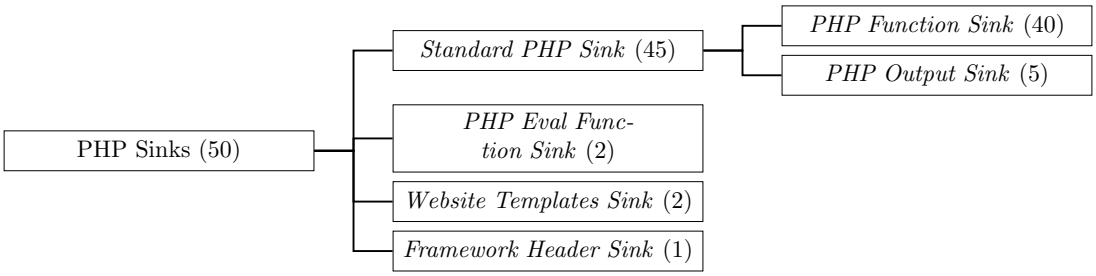



Figure 3: Taxonomy of PHP sinks based on the data set.

## 5.3 PHP Sinks

The sink categories are split into PHP and HTML sinks. An overview of the PHP sink categories is shown in figure 3. This sinks taxonomy is useful for developers because these are the sinks where user input without sanitization will be harmful.

#### *Standard PHP Sink*

This category covers standard output to the web page. Most of our data samples did have sinks in the *Standard PHP Sink* category. It is split into the first category *PHP Function Sink* where methods are used to output the data. Common methods are *echo()* and *print()*. The other category *PHP Output Sink* covers simple output in PHP files, where the *<?...?>* element is used.

#### *PHP Eval Function Sink*

In our data set two samples did have an *eval()* function as sink. These CVE reports were marked as Cross Site Scripting vulnerability. Actually these sinks also open up Direct Dynamic Code Evaluation vulnerabilities (CWE-95). Nevertheless, it can also result in a Cross Site Scripting vulnerability.

#### *Website Templates Sink*

Some PHP frameworks provide template files, which can be used to write PHP similar code with some extra features. In our data set two samples are using *tlp* files from the Smarty framework [9]. If such a framework template is used it falls into this category.

### 5.3.1 *Framework Header Sink*

One sample did have the sink in a header parameter. The PHP framework used in that sample provides a function to set a header parameter. Accordingly, this category was created for sinks which allow to modify the HTTP header.

## 5.4 HTML Context

The PHP sinks categories are more focused on what functions are used to print the data. HTML context rather focus on where the data is presented in the web page. This taxonomy is more similar to the categories provided by CWE. Figure 4 shows the categories found in our data set.

#### *Plain HTML Context*

Outputs which get into a context of this category are in a simple plain HTML part. No special condition like being in a Javascript context or being an attribute. Standard sanitization methods are well suited

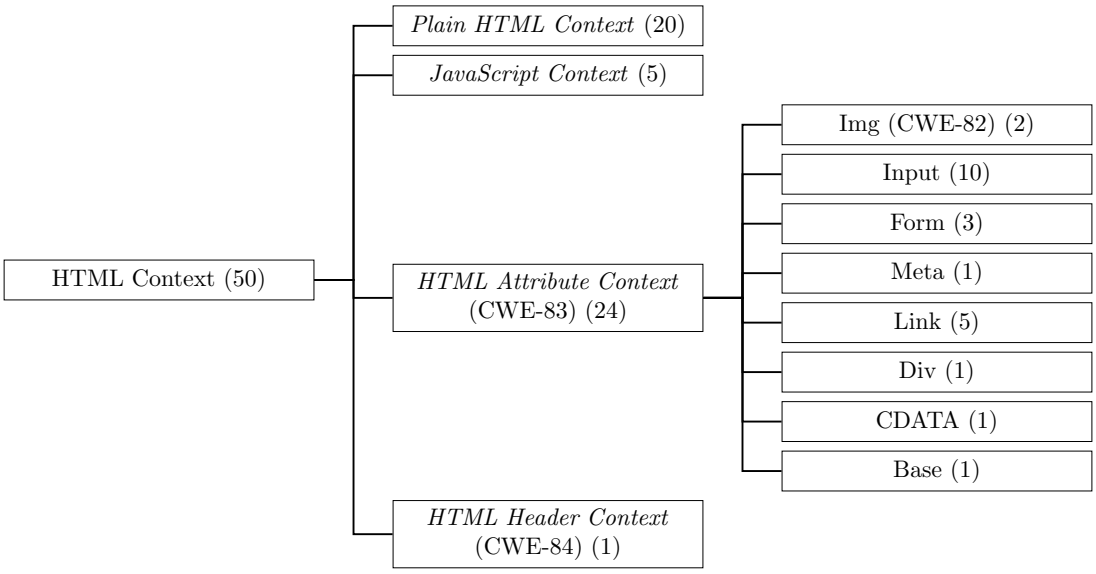



Figure 4: Taxonomy of HTML sinks based on the data set.

for such a context. Also the use of HTML encoding functions is enough as long the output is inside the HTML body [6].

### *JavaScript Context*

Sinks where the output will be in a Javascript context fall into this category. Accordingly, sanitization must be more specialized. Looking into the OWASP XSS prevention sheet [6] explain that simple encoding functions are not enough. The output also should be quoted and sanitized to prevent any Cross Site Scripting attacks [6]. It is important to know that inputs must be data only. Otherwise, the prevention of Cross Site Scripting will be very difficult and requires further restrictions. In our data set only data was used inside a script tag. Another pitfall in this context is the method *htmlspecialchars()* because it does not remove simple quotes without setting the *ENT_QUOTES* parameter. If developers use simple quotes as escaping and do not set the parameter, XSS attacks are still possible.

### *HTML Attribute Context*

This is the same category as CWE-83. The output is inside a HTML attribute. The CWE-82 is specialized version of being an image attribute. In our data set only two samples actually were in a image attribute context. The input has to be sanitized and quoted like in the *JavaScript Context* category.

### *HTML Header Context*

One sample did have a sink from the category *Framework Header Sink*. Accordingly, the context is a HTTP header and that sample falls into this context category. To prevent any Cross Site Scripting attacks in a header, sanitization methods from *Standard Sanitization* are required. Encoding function will not be sufficient.

| HTML Context | Encoding | Sanitization | Sanitization & Escaping |
|---|---|---|---|
| *Plain HTML Context* | prevent | prevent | prevent |
| *HTML Attribute Context* | insufficient | insufficient | prevent |
| *HTML Header Context* | insufficient | prevent | prevent |
| *JavaScript Context* | insufficient | insufficient | prevent |

Table 3: CWE Cross Site Scripting variants mapped to categories.

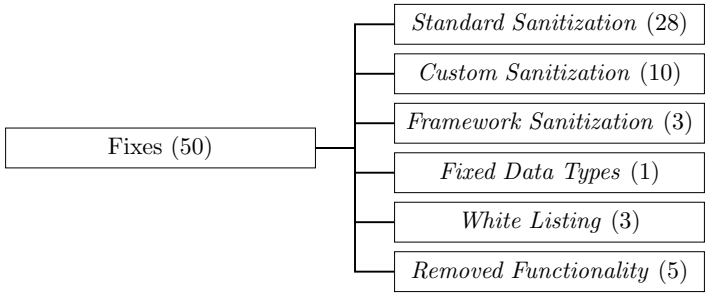



Figure 5: Taxonomy of HTML sinks based on the data set.

### 5.4.1 Sanitization in different contexts

As already mentioned, different HTML context require different sanitization methods. The table 3 provides an overview of what sanitization methods are sufficient enough to prevent any Cross Site Scripting attacks. Accordingly, it would be helpful to sanitize and escape any user input to be protected against XSS attacks in all HTML contexts.

## 5.5 Fixes

An important part to prevent Cross Site Scripting is correct sanitization of user input. Most reviewed projects had no sanitization before the patch related to the CVE report was developed. The patches contain different fixes to sanitize inputs. Figure 5 shows the taxonomy created for the fixes which were used in our data set.

#### *Standard Sanitization*

Fixes of this category used the standard functions provided by PHP. No combination of multiple sanitization methods or a sanitization method from a framework is used. In our sample common methods were *htmlentities()*, *htmlspecialchars()*, *urlencode()*, etc.

#### *Custom Sanitization*

Some patches did fix the vulnerability by using a custom developed sanitization method. Few used a combination of different sanitization methods from the category *Standard Sanitization* as sanitization method. Also some samples did fix the vulnerability by using a sanitization method which did use replace methods like *preg_replace()*.

#### *Framework Sanitization*

Three samples were using sanitization methods provided by a framework to fix the vulnerability. Fixes that use a framework sanitization method falls into this category. For example, the *esc_html()* function

from WordPress [10] fall into this category.

#### *Fixed Data Types*

One sample did use a fixed data type to prevent any Cross Site Scripting vulnerabilities. An ID value was evaluated as an integer value. Accordingly, a simple and elegant way of fixing the vulnerability.

#### *White Listing*

White listing is a common way to prevent any injection attacks. Just fixed values are allowed and all other inputs will be ignored. Three of the samples used white listing to fix the vulnerabilities.

#### *Removed Functionality*

Another category to fix a vulnerability is to remove that output. Even some sample did remove some functionality which contained the vulnerability. At least it fixed the vulnerability.

## 6 Special case: CVE-2012-5163

To get a better understanding of the manual review process this section provides an example of the CVE-2012-5163 report in the open source project Osclass [5]. Looking into the report notes reveals that the id parameter in *enable_category* can be used to inject arbitrary code. Accordingly, the source is already known. Looking into the vulnerable source code reveals that the function *Params::getParam("id")* is used which falls in the ***Framework Source*** source category. The code snippet 6 shows the related source code. As seen on line 9 and 14 standard sanitization methods are used. Consequently, it falls in the insufficient sanitization category ***Standard Sanitization***. Nevertheless, because *$htmlencode* is on default set to false, the sanitization method is not used. The function will return the value without any sanitization for the *enable_category* id. As this sample shows, a developer might think that the *getParam()* function does already sanitize the inputs but in specific conditions it does not.

The code snippet 7 shows a shortened version of the *doModel()* method, which prints the not sanitized user input. Line 4 and 6 shows a little part of the data flow that was tracked by the manual review. The final PHP sink is found on line 7, where the output will be encoded by the function *json_encode* and the sink is the *echo* function. Accordingly, as PHP sink was found from the ***Standard PHP Sink*** category. The *doModel()* functions creates a web page where the found echo result in a plain HTML context. Therefore it falls into the ***Plain HTML Context*** category. The fix was very simple by encapsulating the related *getParam()* call with a *strip_tags()* functions. Thus the fix category is ***Standard Sanitization*** because it is a function provided by PHP. The encoding function is supposed to be used for a JSON context. Accordingly, this does not prevent any XSS attacks in a HTML context.

## 7 Discussion

The results did show that Cross Site Scripting vulnerabilities have a high rate of vulnerabilities where no sanitization is used. The vulnerable source code for CVE-2013-0807, CVE-2014-8793 and CVE-2013-4880 reports have the vulnerabilities including source and sink in one line of code. These results show that Cross Site Scripting is not as present in developer's minds as it should be. For Cross Site Scripting the different HTML contexts are relevant because as described in section 5.4.1 they require different sanitization methods. Accordingly, the prevention mechanisms and context categories have to fit to prevent any attacks.

The CWE sub categories for Cross Site Scripting are not very detailed. As our results show many different categories exist. Especially the context category *JavaScript Context* should exist because it requires more specialized sanitization to prevent any XSS attacks. The method *htmlspecialchars()*

```
static function getParam($param, $htmlencode = false)
{
    if ($param == "") return '' ;
    if (!isset($_REQUEST[$param])) return '' ;

    $value = $_REQUEST[$param];
    if (!is_array($value)) {
        if ($htmlencode) {
            return htmlspecialchars(stripslashes($value), ENT_QUOTES);
        }
    }

    if(get_magic_quotes_gpc()) {
        $value = strip_slashes_extended($value);
    }

    return ($value);
}
```

Figure 6: Params::getParams() function with insufficient sanitization from CVE-2012-5163.

```
// root category
if( $aCategory['fk_i_parent_id'] == '' ) {
...
$aUpdated[] = array('id' => $id) ;
...
$result['affectedIds'] = $aUpdated ;
echo json_encode($result) ;
break ;
}
...
break ;
```

Figure 7: A shortened version of the sink source code part from CVE-2012-5163.

should probably also escape simple quotes as default because developers might not read the documentation carefully enough to know that an additional parameter is required to escape simple quotes. In a *JavaScript Context*context it opens up unnecessary XSS vulnerabilities.

# 8 Conclusion and Future Work

We analysed the source code of 50 GitHub projects which are correlated to CVE reports mentioning Cross Site Scripting and available source code patches. The results show different taxonomies for important source code patterns. Relations to the existing CWE categories are created. Our taxonomy is more focused on the developers point of view. In combination with our previous work [22], [23], three taxonomies for different vulnerabilities categories are created. These taxonomies allow further research using the taxonomy and the source code samples as a dataset. These categories can be used to get a better understanding where Cross Site Scripting vulnerabilities can occur. Especially, the HTML context taxonomy has a big influence on what prevention mechanisms should be used.

The sample set was very small; more samples should be analysed for the source code patterns and

compared to our results. Another interesting aspect would be to research source code patterns of XSS samples in the programming language JavaScript. These patterns could also be compared to our results. This taxonomy could be used to improve the teaching of software security skills for developers. The knowledge of source code patterns can be used to create exercises. An interesting point will be to create these exercises automatically similar to previous research projects [21] [11]. Existing source code from projects can be used and transformed to create these source code patterns. Another interesting point will be using these categories to test static code analysis tools. These can be investigated whether they detect all permutations. It will be interesting to see what combinations of the categories are difficult to be detected from static code analysis tools.

# References

[1] CVE Common Vulnerabilities and Exposures. https://cve.mitre.org/.

[2] GitHub. https://github.com/.

[3] CWE - Common Weakness Enumeration. http://cwe.mitre.org/, 2015.

[4] OWASP Top Ten Project. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project, 2015.

[5] Osclass - open source classified. https://osclass.org/, 2018.

[6] XSS (Cross Site Scripting) Prevention Cheat Sheet. https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet, 2018.

[7] PHP - reserved variables. http://php.net/manual/en/reserved.variables.php, 2018.

[8] SAMATE - SARD, 2018. URL https://samate.nist.gov/SARD.

[9] Smarty PHP Template Engine. https://www.smarty.net/, 2018.

[10] WordPress. https://wordpress.org/, 2018.

[11] B. Dolan-Gavitt, P. Hulin, E. Kirda, T. Leek, A. Mambretti, W. Robertson, F. Ulrich, and R. Whelan. LAVA: Large-Scale Automated Vulnerability Addition. *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*, pages 110–121, 2016. doi: 10.1109/SP.2016.15.

[12] M. V. Gundy, M. V. Gundy, H. Chen, and H. Chen. Noncespaces: using randomization to enforce information ow tracking and thwart cross-site scripting attacks. *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, pages 1–18, 2009.

[13] S. Gupta and B. B. Gupta. Js-san: defense mechanism for html5-based web applications against javascript code injection vulnerabilities. *Security and Communication Networks*, 9(11):1477–1495, 2016.

[14] Z. Hui, S. Huang, B. Hu, and Z. Ren. A taxonomy of software security defects for SST. *Proceedings - 2010 International Conference on Intelligent Computing and Integrated Systems, ICISS2010*, pages 99–103, 2010. doi: 10.1109/ICISS.2010.5656736.

[15] P. Lerthathairat and N. Prompoon. An approach for source code classification to enhance maintainability. *Proceedings of the 2011 8th International Joint Conference on Computer Science and Software Engineering, JCSSE 2011*, pages 319–324, 2011. doi: 10.1109/JCSSE.2011.5930141.

[16] M. T. Louw and V. N. Venkatakrishnan. Blueprint: Robust prevention of cross-site scripting attacks for existing browsers. pages 331–346, May 2009. ISSN 1081-6011. doi: 10.1109/SP.2009.33.

[17] F. Massacci and V. H. Nguyen. Which is the right source for vulnerability studies?: An empirical analysis on Mozilla Firefox. *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, pages 4:1—-4:8, 2010. doi: http://doi.acm.org/10.1145/1853919.1853925.

[18] S. Maurya. Positive security model based server-side solution for prevention of cross-site scripting attacks. *12th IEEE International Conference Electronics, Energy, Environment, Communication, Computer, Control: (E3-C3), INDICON 2015*, pages 1–5, 2016. doi: 10.1109/INDICON.2015.7443473.

[19] S. Maurya and A. Singhrova. Cross Site Scripting Vulnerabilities and Defences: A Review. *International Journal of Computer Techonology and Applications*, 6(June):478 – 482, 2015.

[20] Y. Nadji, P. Saxena, and D. Song. Document structure integrity: A robust basis for cross-site scripting defense. *Proceedings of the Network and Distributed System Security Symposium*, pages 1–42, 2009.

[21] F. Schuckert. PT: Generating Security Vulnerabilities in Source Code. In M. Meier, D. Reinhardt, and S. Wendzel, editors, *Sicherheit 2016 - Sicherheit, Schutz und Zuverlässigkeit*, pages 177–182, Bonn, 2016. Gesellschaft für Informatik e.V.

[22] F. Schuckert, B. Katt, and H. Langweg. Source Code Patterns of SQL Injection Vulnerabilities. *International Conference on Availability, Reliability and Security*, 2017. doi: 10.1145/3098954.3103173.

[23] F. Schuckert, M. Hildner, B. Katt, and H. Langweg. Source Code Patterns of Buffer Overflow Vulnerabilities in Firefox. *Proceedings of Sicherheit 2018*, pages 107–118, 2018. doi: 10.18420/sicherheit2018_08.

[24] B. Stivalet and E. Fong. Large Scale Generation of Complex and Faulty PHP Test Cases. *Proceedings - 2016 IEEE International Conference on Software Testing, Verification and Validation, ICST 2016*, pages 409–415, 2016.

[25] Y. Wu, H. Siy, and R. Gandhi. Empirical results on the study of software vulnerabilities. *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, pages 964–967, 2011.

# Comparing Open Source Search Engine Functionality, Efficiency and Effectiveness with Respect to Digital Forensic Search

Joachim Hansen, Kyle Porter, Andrii Shalaginov, and Katrin Franke

NTNU Digital Forensic Group
Department of Information Security and Communication Technology
Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology
joachim90jh@gmail.com,{kyle.porter,andrii.shalaginov,katrin.franke}@ntnu.no

**Abstract**

Keyword search is one of the key components of the Cyber Crime Investigations. It has a direct influence on precision and relevance of the data found on seized data carriers. However, many of the digital forensics tools developers do not reveal the actual underlying algorithms or source code of their search engines. Therefore, there is a challenge to verify their accuracy and efficiency. On the other hand, open-source search engines are an alternative to using proprietary keyword search tools, where they have extensive functionality and perform well on large-scale datasets. The goal of this paper is to explore the applicability of such search engines in forensics search. The contribution of the paper is two-folded. First, a thorough literature review and comparison of the supported functionality documented by open-source search engines and open-source digital forensic tools was performed. In addition, a survey of existing publicly-available digital forensics datasets was conducted. Second, out of reviewed search engines, Solr and Elasticsearch were selected and compared by their functionality, efficiency in searching and indexing, and effectiveness of search results with respect to digital forensic search using relevant datasets. Our findings should assist those in the digital forensic community when choosing the appropriate open source search engines for keyword search in large-scale datasets.

## 1 Introduction

The current Big Data digital landscape has provided digital forensic investigations with massive amounts of structured and unstructured data to analyze and search for relevant evidence, and the quantity of data to analyze only continues to grow [32]. Such circumstances lead to important considerations such as how forensic practitioners should search through their collected data for investigations in a reasonable amount of time. Another challenge is how to best handle the storage requirements of the data. The use of relational databases to process the data has been already found to be inappropriate for digital investigations years ago. The reason for this is that the majority of the data retrieved is unstructured (for example, human written text) and, therefore, require other approaches that relational storage [31].

Information retrieval systems, such as search engines, are often used to help locate enterprise data where these enterprises oftentimes have to manage large volumes of heterogeneous data structures and formats [15]. For the digital forensic practitioner, the processing of the data must be reliable, forensically sound, and ideally the search engines and forensic tools supporting these ends should utilize algorithms with low memory and time complexities. On the other hand, we have to take into consideration so-called Forensic Search that may put specific restrictions of the keyword search process. These include lack of standardized approach to data preprocessing

and formatting (for subsequent search needs), and the heterogeneous nature of the data (varied file types). In addition, the digital forensics practitioner may experience new data formats that also put demand on quality of the search results, which additionally have to follow Digital Forensics Process guidelines.

The problem with proprietary digital forensic tools is that developers do not reveal the underlying algorithms or source code. Therefore, we examine open-source search engines as an alternative method to using proprietary keyword search tools, where they have extensive functionality and perform well on large-scale datasets. There is a little chance to perform equivalent analysis of proprietary software due to license and source code availability limitations. In particular, the goal of this paper is to evaluate the performance of selected open source search engines and search functionality on forensic data.

Our first contribution is the comparison of the supported functionality documented by open-source search engines and open-source tools. The different search features of several popular open-source search engines and forensic tools are accounted for in an easy to read checklist. Additionally, we conducted a survey on existing publicly-available digital forensic datasets, six of which were used for testing. Our second contribution is an experimental comparison of *Solr* and *Elasticsearch*. This benchmark experiment tests how well they perform at indexing, searching and memory consumption during searching. Our results indicate *Elasticsearch* was generally better then *Solr* at index creation time, minimizing index size and response time for the first run of search terms. *Solr* outperformed *Elasticsearch* on second run of search terms. The difference between the search engines with respect to memory performance during searching was negligible. Information regarding which open-source search engines are available, their search features, and their searching and indexing capabilities is valuable for forensic practitioners when choosing to utilize an open source-search engine for performing keyword search in large-scale datasets.

The paper has the following structure. Section 2 presents the documented functionality of open-source search engines and existing publicly available forensic-related datasets. Experiment approach motivation, data selection and computing environment are given in the Section 3. Analysis of efficiency and effectiveness of the selected open-source search engines on given digital forensics data follow in the Section 4. Finally, Section 5 discusses implications of the study and concludes the paper.

# 2 Open Source Digital Forensic Tools, Search Engines, and Datasets

In this section we describe the results of a survey of the search functionality of open-source search engines and open-source forensics tools. In addition, an overview of the publicly-available digital forensics-related datasets is given.

## 2.1 Functionality of Open Source Digital Forensic Tools and Search Engines

Our choices of open source search engines to analyze was based on popularity of use (Google Trends) and mentions within the scientific literature. To narrow down our selection of open source forensic tools and search engines we selected them based on their degree of documentation and tool category. The inspection process was performed as a combination of targeted manual inspection, keyword searching of technical documentation and source code comments of the software being inspected. By targeted manual inspection, we mean looking at portions of

the documentation more likely to include relevant information. Some documentation pages might be very old or indicate that the documented feature is experimental, and as such were excluded from our review. One issue with the inspection process is that inferences often had to be made on out of context images and text that describes the search capabilities. Moreover, the description was often quite short. Ideally, the software capabilities would be confirmed by practical tests, but perhaps this can be done as future work. Ultimately, Sleuth Kit - Autopsy, Volatility, Mozilla InvestiGator, and Hachoir were the forensic tools selected based on their degree of documentation and catalogue of tools. Elasticsearch, Solr, and Sphinx were the search engines selected based on popularity. Table 1 shows a summary of the open source forensic tool and search engine functionality analysis. A checkmark indicates that the given program has the capability and an empty cell means that it does not. This Table is a result of extensive literature review aimed at understanding of all possible search functionalities existing in digital forensics tools.

While we do not define every capability of each forensic tool and search engine, we do go over the one we utilized for our experiments. *Full text search* is suitable for finding relevant documents in a large set of unstructured data [14, 16]. A document in full text search is considered a list of searchable terms (e.g. words and numbers) [16]. The terms are usually indexed in order to have faster subsequent searches.

## 2.2 Publicly available digital forensics-related databases

To support this study, an overview of the publicly-available datasets related to digital forensic was performed. In contrast to the overview of all possible Digital Forensics datasets performed in 2017 [9], this work focuses only on publicly-available, which means that anybody can fetch them and repeat the experiments. It is important to understand that there is no way of getting real-world data from crime investigation, however, there are plenty of datasets created by researchers for data analysis purposes. The systematic literature review included following the iterations:

1. Search digital libraries, scan scientific articles for names, direct links or sources related to the datasets below, and use this information on the Google search engine to identify individual datasets or repositories of datasets.
2. Document search phrases that resulted in identifying new datasets.
3. Repeat step 1 and 2 with other resources like github, Kaggle and figshare to locate more datasets.

Table 2 is a summary of review process to identify datasets candidates to be part of the experiment set in the next section. We identified 83 datasets that are publicly available and attributed to Digital Forensic. This number excludes biometric datasets such as images of fingerprints, hand signature, gait, voice recognition and iris. However, the review will include authorship attribution corpus. To our knowledge, there has not been performed any comprehensive enumeration of forensics-related datasets.

Table 1: Comparison of search capabilities and functionality
**Source:** [2, 6, 10, 19, 20, 23, 25, 27]

| Capability | Sleuthkit | Volatility | Mozilla Invest-Gator | Hachoir | Elasticsearch | Solr | Sphinx |
|---|---|---|---|---|---|---|---|
| Regular expression | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Decide/Insensitive case | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | |
| Concurrent search | ✓ | | | | ✓ | | ✓ |
| Automate search, with respect to keywordlist | ✓ | | | | | | |
| Import keywords | ✓ | | | | | | |
| Export keywords | ✓ | | | | | | |
| Periodical search | ✓ | | | | | | |
| Substring matching | ✓ | | | | ✓ | ✓ | ✓ |
| Export search results | ✓ | | | | ✓ | ✓ | |
| Match highlighting | ✓ | | | | ✓ | ✓ | ✓ |
| UTF-8 Encoding support | ✓ | | ✓ | ✓ | ? | ✓ | ✓ |
| UTF-16 Encoding support | ✓ | | | ✓ | ? | | |
| ISO-8859-1 Encoding support | | | | ✓ | ? | | |
| Deduplication support | ✓ | | | | | ✓ | |
| Approximate hash based matching | ✓ | | | | | | |
| Orphan/deleted file search | ✓ | | | | | | |
| RAM search | ✓ | ✓ | ✓ | | | | |
| Matching memory structures (pre-made) | | ✓ | | | | | |
| Hash database lookups | ✓ | | | | | | |
| Wildcard | | ✓ | | | ✓ | ✓ | ✓ |
| Binary search | ✓ | ✓ | | | | | |
| HTML renderer for search results | | ✓ | | | | | |
| Support for masking sensitive fields | | | ✓ | | | | |
| Exact hash matching | | | ✓ | | | | |
| System provided keyword suggestions | | | | | | ✓ | |
| AND, OR, NOT, GROUP boolean operators | | | | | ✓ | ✓ | ✓ |
| + boolean operator (term must exist) | | | | | | ✓ | |
| File search filter | | | ✓ | | | | |
| Retrieval of documents not matching filters | | | ✓ | | | | |
| Set max search hits | | | ✓ | | ✓ | ✓ | |
| Stripping senstive metadata | | | ✓ | | | | |
| Increase search priority of important indexes | | | | | ✓ | | |
| Terminate search after a given elapsed time | | | | | ✓ | ✓ | |
| Sorting search results | | | | | ✓ | ✓ | ✓ |
| Customized message/ post-search action | | | | | ✓ | ✓ | |
| Aggregated summary of search results | | | | | ✓ | | |
| Narrow search results with post filter | | | | | ✓ | ✓ | |
| Set relevancy weight for field | | | | | ✓ | ✓ | |
| MoreLikeThisQuery | | | | | ✓ | ✓ | |
| Search result clustering | | | | | | ✓ | ✓ |
| Minimum matching criteria | | | | | | ✓ | |
| Fixed relevancy score | | | | | | ✓ | |
| Field collapsing | | | | | ✓ | ✓ | |
| Support for TF-IDF | | | | | ✓ | ✓ | ✓ |
| Language detection on index time | | | | | | ✓ | |
| Fuzzy matching | | | | | ✓ | ✓ | ✓ |
| Faceted search | | | | | ✓ | ✓ | |
| Phonetic search | | | | | ✓ | | |
| Geospatial search | | | | | ✓ | ✓ | |
| Streamed search | | | | | | ✓ | |

| Name | Catagory |
|---|---|
| BAC | Authorship |
| CTFAC | Authorship |
| PCSN | Authorship |
| TBGC | Authorship |
| Personae | Authorship |
| PAN-Enron | Authorship |
| PAN/CLEF12 | Authorship |
| PAN13 | Authorship |
| PAN14 | Authorship |
| PAN15 | Authorship |
| RCTAC | Authorship |
| MUD03 | Authorship |
| netre-sec | Collection of misc |
| MTA13-17 | Collection of misc |
| pcapr | Collection of misc |
| PCAPsDB | Collection of misc |
| CAIDA | Collection of misc |
| csmining | Collection of misc |
| AZSecure-data | Collection of misc |
| Digital Corpora | Collection of misc |
| DFCF review | Collection of misc |

| Name | Catagory |
|---|---|
| GIfiles | Email |
| USHCE | Email |
| 419 fraud dataset | Email |
| MLE200 | Email |
| Enrondata | Email |
| RAISE | Files |
| SherLock | Files |
| AndroZoo | Files |
| CTD15 | Financial fraud |
| UCSD-FICO-09 | Financial fraud |
| CMS | Financial fraud |
| PaySim | Financial fraud |
| BankSim | Financial fraud |
| MICC | Forgery |
| Brian Carrier | Forensic Images |
| RDC | Forensic Images |
| CFReDS | Forensic Images |
| VirusShare | Malware |
| BIG15 | Malware |

| Name | Catagory |
|---|---|
| Drebin | Malware |
| DroidWare | Malware |
| MILCOM16 | Malware |
| Kharon | Malware |
| Mudflow | Malware |
| ISOT2010 | Malware |
| ECML/PKDD07 | Malware |
| CSIC10 | Malware |
| BlogPcap | Malware |
| Malwarerec | Malware |
| CTU-13 | Malware |
| ISCX | Malware |
| ISCXAB | Malware |
| DAROA98/99 | Network |
| DARPA2000 | Network |
| MAWILab | Network |
| KDD Cup99 | Network |
| UNSW-NB15 | Network |
| NSA-CDX | Network |
| ADFA | Network |

| Name | Catagory |
|---|---|
| Kyoto data | Network |
| crawdad | Network |
| ICS-pcap | Network |
| Common Crawl | Network |
| NSL-KDD | Network |
| ISCXTNT | Network |
| ISCXVNV | Network |
| ISCXIDS | Network |
| YPFC | Password |
| VincentPassword | Password |
| MBT08 | RAM |
| NUSSC | SMS |
| WebbSC11 | SPAM |
| DITSSC | SPAM |
| TREC05-07 | SPAM |
| Hewlett spam | SPAM |
| WEBSPAMUK07 | SPAM |
| microblogPCU | SPAM |
| TREC11 | SPAM |
| SPAM/HAM | SPAM |
| phishtank | Phising |
| millersmiles | Phising |
| PWDS15 | Phising |

Table 2: Dataset names and category

# 3 Experiment Methodology

The focus of practical evaluation in this study is to provide understanding of how well the open-source search engines perform on digital forensics-related data and whether their effectiveness can be considered acceptable for criminal investigations.

## 3.1 Experimental Approach

To measure the difference in effectiveness and efficiency for both search engines, the following experiments were conducted:

- Experiments with fulltext searching
- Experiments performed on a set of search engines (*Solr* and *Elasticsearch*).
- Set of keywords based on domain knowledge of datasets and a search for strings that are not present in the dataset
- Searching within an index (i.e. not searching across all indexes or multiple indexes at the same time).
- Search time
- Cache temperature
- Memory measurement during search
- Search Accuracy - count of clear cut misses
- Out of box configurations (default values)

There are two main limitations of the experiment. The first being that the experiments are performed on only one virtual machine. This environment does not allow testing for how well the search engines perform at distributed search. The second issue is that only the default configurations was tested (out-of-the-box setup) with *Solr* [11] and *Elasticsearch* [7].

## 3.2 Data selection

When working with and evaluating digital forensics tools it is difficult to obtain a real-world dataset due to sensitivity of the information, which might be related to criminal cases. Addition-

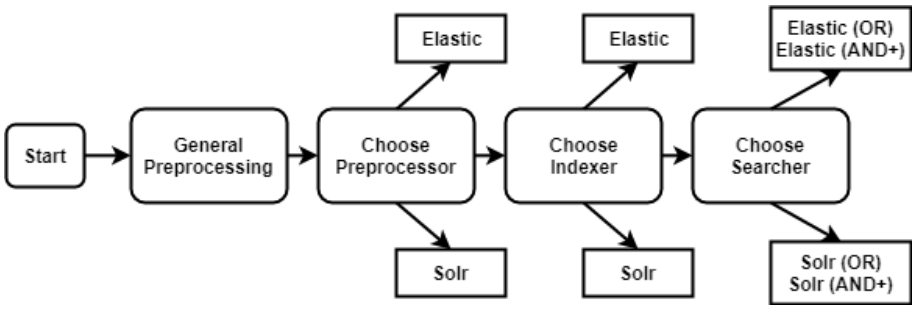
ally, digital forensics experts may experience a variety of data formats, even when considering primarily text datasets. One of the criteria was also to look for large-scale datasets to have a better performance testing, therefore, the datasets were selected based on dataset size and forensic category. The list is shown below.

1. **Fraud:** Enron email dataset [4].
2. **Network:** Snort IDS log file [26].
3. **Email:** Hillary Clinton emails [13].
4. **Malware:** VirusTotal and PE32 reports [21].
5. **Spam:** DITSSC [3].
6. **SMS:** NUSSC [5]

### 3.3 Dataset Pre-processing, Indexing and Search Queries

A short description of the experimental design including the preprocessing, indexing and searching steps in the experiment are given here. As shown in Figure 1, the first step is running the datasets though a general preprocessing step.

Figure 1: Flowchart experimental design




The data pre-processings step includes following sub-steps:

- Remove *JSON* structure, *XML* structure and/or dataset specific structure from all the datasets.
- Removes characters that cannot be processed by the *JSON* parser in *Solr* or *Elasticsearch*.
- Removes junk (e.g. terms that is not interesting, such as many repeated characters)
- Removes unnecessary whitespace
- Removes empty lines
- Separate dataset entries on their own lines.
- Split the dataset into **bulk files** of 1000 lines each (each line is considered as a document for importing).

Then after general pre-processing the dataset, bulk files have to go through another two pre-processing steps, one for *Elasticsearch* and one for *Solr* as shown in the Figure 2a and Figure 2b respectively.

A single content field entry is 1 *JSON* document to be indexed. We call the output from these steps bulk *Elastic* and bulk *Solr* respectively. The next 2 steps are *Elastic* and *Solr* specific bulk indexer. These indexers take the *Elastic* and *Solr* bulk files as input and indexes all the bulk files/the entire dataset. During these steps, the following measurements are taken:

```
{ "index":{} }
{"content":"bulk file line n" }
{ "index":{} }
{"content":"bulk file line n+1" }
```

(a) JSON format for ElasticSearch

```
[
{"content":"bulk file line n" }
{"content":"bulk file line n+1" }
]
```

(b) JSON format for Solr

- The *Linux command time* is used to capture total elapsed time of indexer.
- *QTime* in *Solr* and *took* in *Elasticsearch* are used as response time.
- A size command in *Elasticsearch* are used to get the size of the index. Additionally, the filesystem in our environment is inspected to find the index size in *Solr*.

The last two steps are searching with *Solr* and *Elasticsearch*. The Solr(OR)/Elastic(OR) and Solr(AND+)/Elastic(AND+) are similar. The (OR) search queries are using the Boolean OR operator between multiple search terms. The (AND+) search queries requires that all search terms in the search string are present in the matching string, and that the terms are in the right order for them being a match. At these steps, the following measurements are taken: Clear cut misses, Memory stats are captured with the *Linux top* command, *QTime* and *took* response time.

### 3.4 Computing Environment

To perform benchmarking of effectiveness and efficiency of the selected search engines, the following setup was used: Virtual Machine (6 cores, 40GB RAM and 2TB storage) with Ubuntu 16.04.3 LTS, *Openjdk 1.8.0_131*, *Elasticsearch 6.0.1* and *Solr 7.1.0* and *Solr Cloud*.

## 4 Results & Analysis

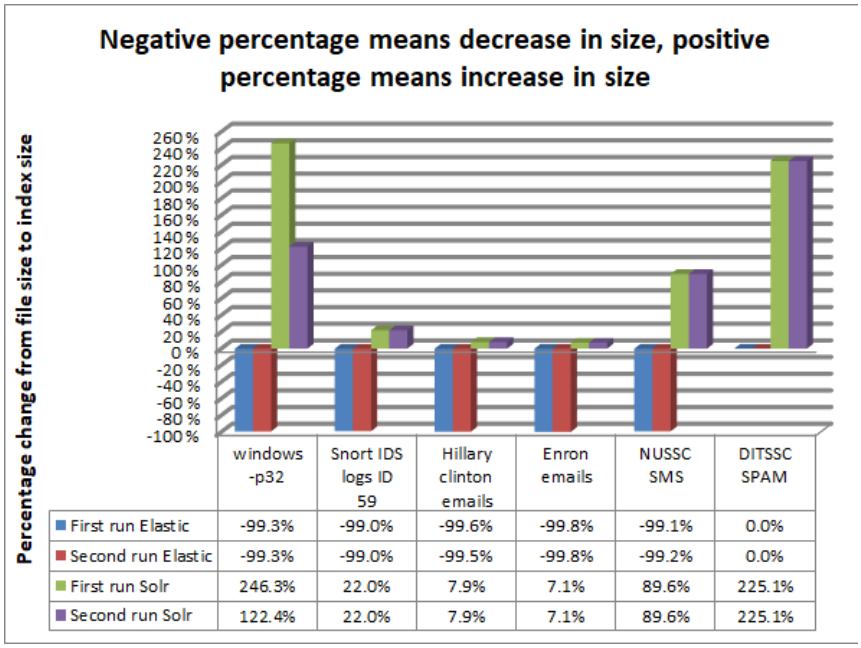
This section is devoted to quantitative comparison of the efficiency in indexing and searching of *Solr* and *Elasticsearch* with subsequent analysis of the obtained results.

### 4.1 Indexer performance

The benchmark of the indexer considers the change in size starting from the collection of documents to be indexed to the resulting index size, as well as the real time, response time and delta (difference between the times) when measuring index creation time. Real time (in milliseconds) is the total elapsed time by the indexer process measured by the Linux time command. Response time is the time it takes from the indexer getting the index request until completion of the indexer, given in milliseconds by QTime (*Solr*) and took (*Elasticsearch*). Delta is the the remaining time when subtracting the response time from real time (i.e. delta = real time - response time). We measure for delta since the response time and real time may not be the same. This is because the real time includes I/O overhead (request creation, context switching, writing to console, etc.). For each benchmark test we run the indexer twice on each data set, denoted first run and second run respectively. The Figure 3 shows the percentage size change from "to be indexed" data-set to "indexed data-set" for *Solr* and *Elasticsearch*.

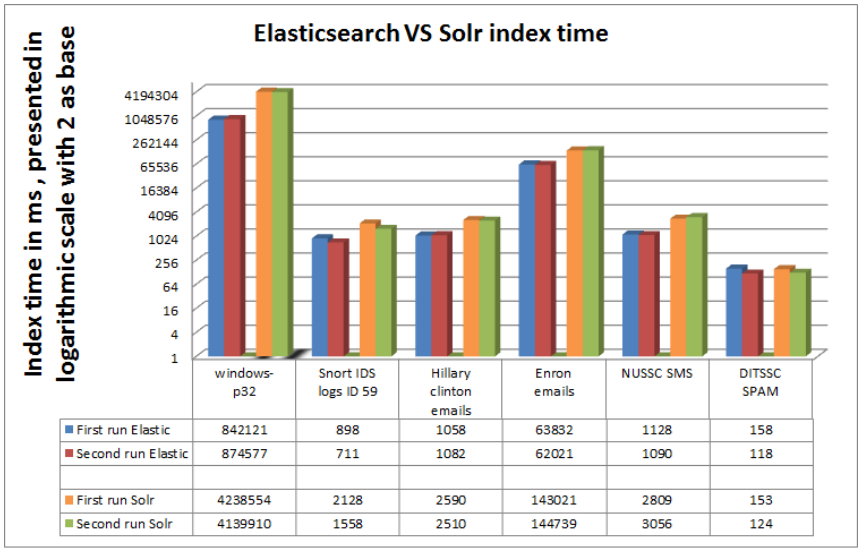
Figure 4 shows a comparison of the response time for *Solr* and *Elasticsearch* indexer for each individual dataset. In Figure 6 shows the real elapsed time for indexing and how much time was spent on I/O for *Solr* and *Elasticsearch*.

Figure 3: The change in index size




| | windows -p32 | Snort IDS logs ID 59 | Hillary clinton emails | Enron emails | NUSSC SMS | DITSSC SPAM |
|---|---|---|---|---|---|---|
| First run Elastic | -99.3% | -99.0% | -99.6% | -99.8% | -99.1% | 0.0% |
| Second run Elastic | -99.3% | -99.0% | -99.5% | -99.8% | -99.2% | 0.0% |
| First run Solr | 246.3% | 22.0% | 7.9% | 7.1% | 89.6% | 225.1% |
| Second run Solr | 122.4% | 22.0% | 7.9% | 7.1% | 89.6% | 225.1% |

Figure 4: Index time Took and QTime




| | windows-p32 | Snort IDS logs ID 59 | Hillary clinton emails | Enron emails | NUSSC SMS | DITSSC SPAM |
|---|---|---|---|---|---|---|
| First run Elastic | 842121 | 898 | 1058 | 63832 | 1128 | 158 |
| Second run Elastic | 874577 | 711 | 1082 | 62021 | 1090 | 118 |
| | | | | | | |
| First run Solr | 4238554 | 2128 | 2590 | 143021 | 2809 | 153 |
| Second run Solr | 4139910 | 1558 | 2510 | 144739 | 3056 | 124 |

From our indexing experiments, we find that Elasticsearch is generally much faster than *Solr* at indexing (index response time) for both the first and second runs with the exception of the DITSSC SPAM dataset (as seen in Figure 4). Furthermore, we can see that the index response time is not static from the first run to the second. So there is no consistent reduction in response time from run 1 to run 2. From Figure 6 we see that the real time and response time are proportionally closer in *Solr* than in *Elasticsearch*. Therefore, more time is spent on delta in *Elasticsearch* than *Solr*.

Figure 5: One iteration of the indexer

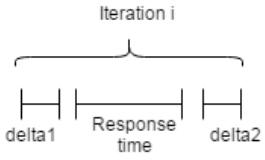
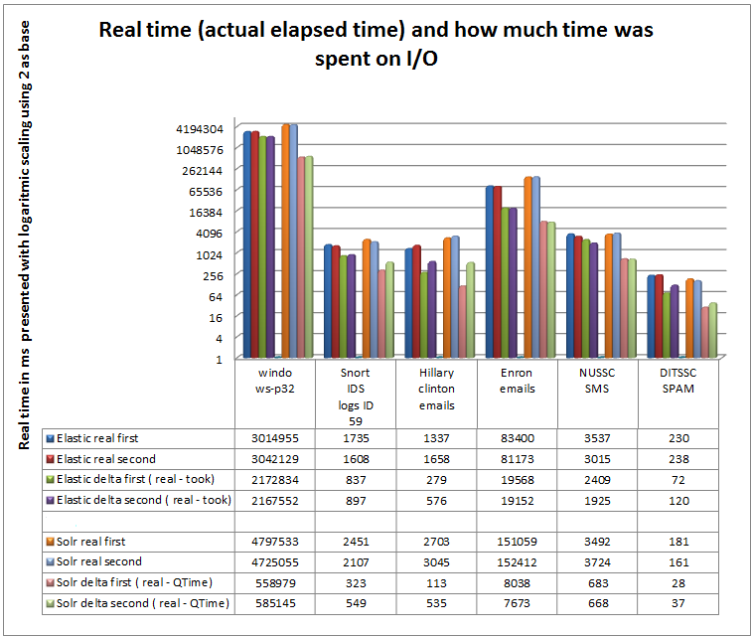



Figure 6: Index real time and delta




| | windows-p32 | Snort IDS logs ID 59 | Hillary clinton emails | Enron emails | NUSSC SMS | DITSSC SPAM |
|---|---|---|---|---|---|---|
| Elastic real first | 3014955 | 1735 | 1337 | 83400 | 3537 | 230 |
| Elastic real second | 3042129 | 1608 | 1658 | 81173 | 3015 | 238 |
| Elastic delta first ( real - took) | 2172834 | 837 | 279 | 19568 | 2409 | 72 |
| Elastic delta second ( real - took) | 2167552 | 897 | 576 | 19152 | 1925 | 120 |
| | | | | | | |
| Solr real first | 4797533 | 2451 | 2703 | 151059 | 3492 | 181 |
| Solr real second | 4725055 | 2107 | 3045 | 152412 | 3724 | 161 |
| Solr delta first ( real - QTime) | 558979 | 323 | 113 | 8038 | 683 | 28 |
| Solr delta second ( real - QTime) | 585145 | 549 | 535 | 7673 | 668 | 37 |

Figure 5 shows a single iteration of the *Solr* or *Elasticsearch* indexer. The indexer would run i=(number of bulk files for dataset d) times. The indexer is still working but the actions is not part of the request handler. It is assumed that delta is composed of some actions that is done both before and after the actions of the request handler [1,8,30]. The second assumption is that the actions that can be attributed to delta and response time is similar for *Elasticsearch* and *Solr*, with some minor exceptions. Possible actions that could contribute to the delta time are reading the bulk Elasticsearch/Solr files to be indexed, transferring with curl, sending request to the request handler, writing to console, etc. Furthermore, the request handler is responsible for creating the index structure, creating ID automatically for the JSON document, commit

(*Solr* specific command), and writing index to disk.

The two main reasons were identified as to why *Elasticsearch* uses more time with Delta than *Solr*. The first is that the *JSON* bulk index file format in *Elasticsearch*, with the { "index":{} } lines, resulted in a "Elastic bulk index file" that was about twice as large as in the case of *Solr*. The second is that it has been observed that *Elasticsearch* writes far more information to the console than *Solr* during indexing. We consider the latter to be the highest contributing factor.

*Elasticsearch* outperformed *Solr* on index creation time as more emphasis was put on the request handler actions as opposed to I/O. One aspect that influences indexing creation time is the frequency of commits. The commit operation was performed at every iteration of the indexer script and may therefore be partially responsible for slowing down the overall indexing process. While the commit operations are needed in *Solr* in order to make the documents available for search, the operation is not free [17,30]. It was recommended to limit the frequency of commits to improve indexer performance with respect to index creation time. Another influencing factor is the document size and their line lengths. Large gaps in response time between *Solr* and *Elasticsearch* may be seen in Figure 4, but in particular the largest gaps occurred when there are many lines with low character counts (for example the PE32 dataset). So, if *Elasticsearch* is slightly faster at indexing smaller documents over *Solr* (where there are many of these smaller documents), then it can explain the gap. Given the big difference in size between the indexes in *Solr* and *Elasticsearch* (see Figure 3), the longer index creation time in *Solr* can partly be explained by having to write more to disk and more processing of the documents to create the index.

*Solr* cloud has a dependency on a server/program called *Apache ZooKeeper* and its database [24]. In our experiment where the *Zookeper* server and *Solr* was located on the same virtual machine, it might cause *Solr* and *Zookeper* to fight over I/O resources and therefore trigger a *Zookeper* timeout [24]. This could explain the gap in indexing performance.

There is a consideration to be made in *Solr*, if the segment count is set to a low number, then more merges will occur when indexing, that negatively effects indexing performance [22]. On the upside, queries will be faster as there are fewer places to search. If on the other hand the segment count is high, then we get the opposite situation with improved indexed speed and degraded search performance. A low segment count can be the reason why *Solr* seems to perform badly with indexing yet be good at searching.

## 4.2 Search performance

Search performance was evaluated with three different cases on both *cold* (first run) and *warm* search phrases (second run). The meaning of *"cold"* is the use of search phrase for the first time against a given data set, while *"warm"* infers that the query has been run before. The search cases for evaluation is described below.

1. Single term search (e.g. a word or sequence of symbols)
2. Multiple term search (for AND+ and OR queries)
3. Searching for something that is not present in the dataset (i.e. MD5 hash of the string "Joachim Hansen" = "02edbd94746bc69677e969a89c4eb0d8").

Figure 7 shows the aggregated search hits and Figure 8 presents the aggregated search time for the three search cases, where the timings and hits for the different dataset experiments were

aggregated with respect to the different search cases.

Figure 7: Aggregated search hits

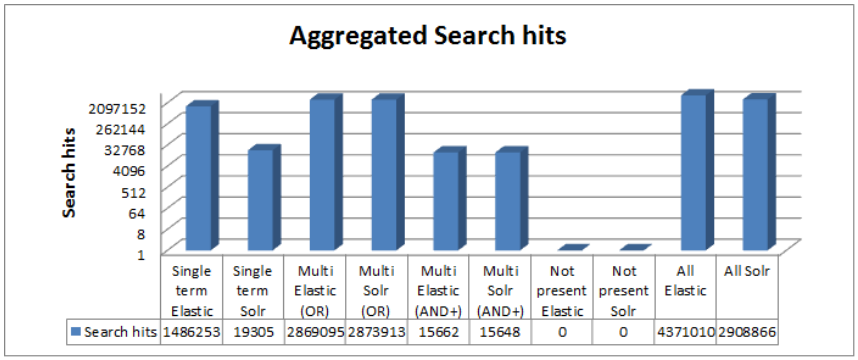
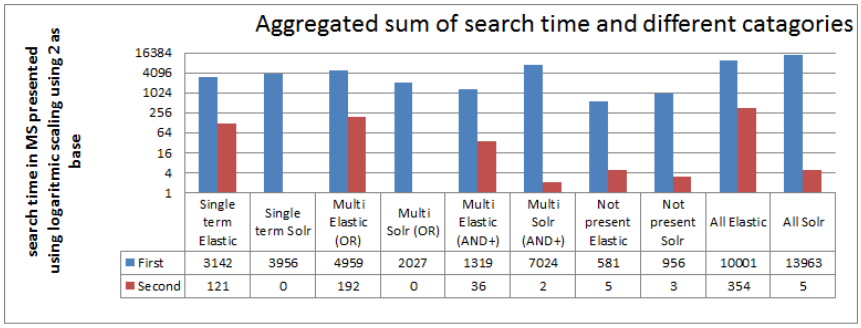



Figure 8: Search time aggregated




From Figure 7 we can see that for the majority of the search categories that there is only a small difference between the number of search hits between *Solr* and *Elasticsearch*. The exception to this finding is the single term search where *Elasticsearch* had 7,598% more search hits over *Solr*. The reasons for this was either that the search queries do not behave the same way, parsing index data was done differently in the search engines, or one search engine parses more information (more exhaustive search) than the other. It was also assumed that OR and AND operators did not affect single terms search. However, this assumption could be incorrect and can be the reason for the large difference between the number of search hits in *Solr* and *Elasticsearch* with respect to single term search. Interestingly, both *Solr* and *Elasticsearch* has 2 clear cut misses, meaning that no search hits were returned when there is at least one matching occurrence of the search string present in the dataset.

Figure 8 shows the aggregated sum of the search time for all datasets categorized by search case. Our experiments show that *Elasticsearch* is better than *Solr* for 3 out of the 4 search categories for the first run, but *Solr* proved to be much better than *Elasticsearch* for the second run. For instance, *Elasticsearch* has a 6,980% increase of search time as an aggregate for the second run over *Solr*. We believe more importance should be placed on the second run, as it is more like a real operating environment with a warm cache.

As mentioned before, a high segment count could negatively affect search performance and that

both *Solr* and *Elasticsearch* emphasize search performance over compression as their default behavior. If multiple shards are present on the same node/host/computer (not distributed) then search performance can be negatively affected according to [12]. This is the case with *Elasticsearch* where 5 shards are on the same host machine. *Elasticsearch* outperforms *Solr* in 4 out of 5 search categories but *Solr* performs better at the second run. We argue that the latter observation was because *Solr* is better (e.g. caching more items or caching the right set of items) at using the cached items from the previous search then *Elasticsearch* [29].

The memory statistics in Table 3 are captured just prior to initiating a search query, during first and second attempt of the same search and the capturing process is ended just seconds after the searches complete. There was in total 84 searches (44 for *Solr* and 44 for *Elasticsearch*) that contribute to the statistics in the table below.

Table 3: Memory stats Elasticsearch and Solr during search

| **Elasticsearch** | | | | | **Solr** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Virtual memory(VIRT): Size in GiB | | | | | Virtual memory(VIRT): Size in GiB | | | | |
| Average | Max | Min | Delta | Mode | Average | Max | Min | Delta | Mode |
| 42.831 | 42.831 | 42.831 | 0 | 42.831 | 29.144 | 29.144 | 29.144 | 0 | 29.144 |
| Physical memory (not swappable) - RES:size in GiB | | | | | Physical memory (not swappable) - RES: size in GiB | | | | |
| 2.807 | 2,898 | 2.666 | 0.232 | 2.898 | 2.936 | 2.964 | 2.881 | 0.083 | 2.964 |
| shared memory (SHR): size in GiB (rounded up) | | | | | shared memory (SHR): size in GiB | | | | |
| 0.34 | 0.43 | 0.2 | 0.23 | 0.43 | 2.296 | 2.323 | 2.241 | 0.082 | 2.323 |

While there are differences in memory performance, they are not significant. One question that had arisen is what was considered when the memory was measured. The code should have summed up all threads for the process under question. But what about forking short lived sub processes that are helping with a search task, and furthermore, what part of *Solr* are captured. *Solr* can be divided into *Solr*, *Solr Cloud* and *Zookeper*. So, which one of these was parts of the memory summary? *Solr* did perform better, but if not all of these processes were taken into account, then the memory performance might not favour *Solr* after all.

# 5 Discussions & Conclusion

In this work we performed a review of the functionality of popular and well documented open source forensic tools and search engines, conducted a literature review of publicly-available forensic datasets, and then performed a benchmarking experiments. These included memory and time consumption comparison of the indexing and fulltext searching processes of *Elasticsearch* and *Solr*.

*Solr* and *Elasticsearch* supported much of the same functionality, but there were some important distinctions. From one side *Solr* has support of deduplication [28], approximate hash based matching, keyword suggestions, and search results clustering [18]. On the other hand, *Elasticsearch* has support for capabilities such as phonetic search. All of these functions are important for digital forensic search, but *Solr* has more unique capabilities that would assist search in large-scale datasets.

The benchmarking experiments show that *Elasticsearch* index creation time was faster than *Solr* for 11 out of 12 trials of the indexer. The one exception was the first run of the smallest dataset. Furthermore, in *Elasticsearch* the resulting index size is reduced from the original dataset by around 99%, while we see an increase in index size in *Solr* ranging from a few percentage to up

to around 240%. Our experiments indicate that *Elasticsearch* is favorable over *Solr* regarding both index size and index creation time.

*Elasticsearch* was better than *Solr* at 3 out of 4 search test categories for the first run, but *Solr* was performed better than *Elasticsearch* for the second run. There was also the same number of clear cut misses for both search engines. The second run should be more like a real operating environment with a warm cache. With respect to memory consumption, there was a big difference between *Elasticsearch* and *Solr* usage of Virtual memory. *Elasticsearch* uses around 13 more GiB on Virtual memory than *Solr*.

Future work can improve our results in the following ways. A comparison on specific search algorithms should be performed on the specific indexing and search methods used by *Elasticsearch* and *Solr*. What would greatly assist our results are performing our experiments in a multi-virtual machine environment and spread the databases, shards and servers amongst the hosts. This environment would be suitable for testing how well the search engines perform at distributed search. The experiments could also test different import methods, importing using multiple threads and using different bulk sizes to test which approach minimizes index creation time.

# References

[1] Wiki Apache. General, 2013. Last accessed (DD/MM/YYYY) 18/11/2017. URL: https://wiki.apache.org/solr/SolrTerminology.

[2] Autopsy. Autopsy - the sleuth kit. accessed: 09/10/2017. URL: http://www.sleuthkit.org/autopsy/.

[3] Tao Chen and Min-Yen Kan. Creating a live, public short message service corpus: the nus sms corpus. *Language Resources and Evaluation*, 47(2):299–335, Jun 2013. URL: https://doi.org/10.1007/s10579-012-9197-9, doi:10.1007/s10579-012-9197-9.

[4] William Cukierski. The enron email dataset - 500,000+ emails from 150 employees of the enron corporation, 2016. Last accessed (DD/MM/YYYY) 21/11/2017. URL: https://www.kaggle.com/wcukierski/enron-email-dataset.

[5] Sarah Jane Delany, Mark Buckley, and Derek Greene. Sms spam filtering: Methods and data. *Expert Systems with Applications*, 39(10):9899 – 9908, 2012. URL: http://www.sciencedirect.com/science/article/pii/S0957417412002977.

[6] Elastic. Elastic search. accessed 24.04.17. URL: https://www.elastic.co/.

[7] Elasticsearch. Install elasticsearch. Last accessed (DD/MM/YYYY) 4/12/2017. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/deb.html#deb-repo.

[8] Elasticsearch. The search api. Last accessed (DD/MM/YYYY) 18/11/2017. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/_the_search_api.html.

[9] Cinthya Grajeda, Frank Breitinger, and Ibrahim Baggili. Availability of datasets for digital forensics and what is missing. *Digital Investigation*, 22:S94 – S105, 2017.

[10] hachoir. hachoir-subfile program. Last accessed (DD/MM/YYYY) 12/10/2017. URL: http://hachoir3.readthedocs.io/subfile.html.

[11] howtoforge. How to install and configure solr 6 on ubuntu 16.04. Last accessed (DD/MM/YYYY) 4/12/2017. URL: https://www.howtoforge.com/tutorial/how-to-install-and-configure-solr-on-ubuntu-1604/.

[12] Ben Hundley. Small static dataset, 2-3 gb, 2015. Last accessed (DD/MM/YYYY) 7/12/2017. URL: https://qbox.io/blog/optimizing-elasticsearch-how-many-shards-per-index.

[13] Kaggle. Hillary clinton's emails, 2016. Last accessed (DD/MM/YYYY) 02/10/2017. URL: https://www.kaggle.com/kaggle/hillary-clinton-emails.

[14] Mark Krellenstein. Starting a search application, 2009. accessed 25.04.17. URL: https://whitepapers.em360tech.com/wp-content/files_mf/white_paper/lucid2.pdf.

[15] Yunyao Li, Ziyang Liu, and Huaiyu Zhu. Enterprise search in the big data era: Recent developments and open challenges. *Proc. VLDB Endow.*, 7(13):1717–1718, August 2014. URL: http://dx.doi.org/10.14778/2733004.2733071, doi:10.14778/2733004.2733071.

[16] Lucidworks. Full text search engines vs. dbms (whitepaper). accessed 25.04.17. URL: https://lucidworks.com/2009/09/02/full-text-search-engines-vs-dbms/.

[17] lucidworks. Understanding transaction logs, soft commit and commit in solrcloud. Last accessed (DD/MM/YYYY) 7/12/2017. URL: https://lucidworks.com/2013/08/23/understanding-transaction-logs-softcommit-and-commit-in-sorlcloud/.

[18] S. Mascarnes, P. Lopes, and P. Sakhare. Search model for searching the evidence in digital forensic analysis. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, pages 1353–1358, Oct 2015. doi:10.1109/ICGCIoT.2015.7380677.

[19] MIG. mig/conf/mig-agent.cfg.inc, 2017. Last accessed (DD/MM/YYYY) 11/10/2017. URL: https://github.com/mozilla/mig/blob/a2fe0fed53fb75d6c1ece5f917268c79774ca0ef/conf/mig-agent.cfg.inc.

[20] mig. mig/doc/concepts.rst, 2017. Last accessed (DD/MM/YYYY) 11/10/2017. URL: https://github.com/mozilla/mig/blob/master/doc/concepts.rst.

[21] Andrii Shalaginov, Lars Strande Grini, and Katrin Franke. Understanding neuro-fuzzy on a class of multinomial malware detection problems. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 684–691. IEEE, 2016.

[22] Solr. Indexconfig in solrconfig. Last accessed (DD/MM/YYYY) 7/12/2017. URL: https://lucene.apache.org/solr/guide/6_6/indexconfig-in-solrconfig.html.

[23] Solr. Learn more about solr. Accessed on 22.03.2017. URL: http://lucene.apache.org/solr/.

[24] Solrwiki. Solrcloud, 2017. Last accessed (DD/MM/YYYY) 7/12/2017. URL: https://wiki.apache.org/solr/SolrPerformanceProblems.

[25] Sphinx. Sphinx - open source search server. accessed:22/10/2017. URL: http://sphinxsearch.com/.

[26] United states military academy west point. Data sets, 2009. Last accessed (DD/MM/YYYY) 26/09/2017. URL: Unitedstatesmilitaryacademywestpoint.

[27] Volatility. Volatility framework, 2017. accessed:11/10/2017. URL: https://github.com/volatilityfoundation/volatility/.

[28] W. B. Wang, M. L. Huang, L. Lu, and J. Zhang. Improving performance of forensics investigation with parallel coordinates visual analytics. In *2014 IEEE 17th International Conference on Computational Science and Engineering*, pages 1838–1843, Dec 2014. doi:10.1109/CSE.2014.337.

[29] wikiApache. Cache autowarm count considerations, 2014. Last accessed (DD/MM/YYYY) 7/12/2017. URL: https://wiki.apache.org/solr/SolrPerformanceFactors.

[30] wikiapache. How can indexing be accelerated, 2016. Last accessed (DD/MM/YYYY) 7/12/2017. URL: https://wiki.apache.org/solr/FAQ#Why_does_the_request_time_out_sometimes_when_doing_commits.3F.

[31] W. M. S. Yafooz, S. Z. Z. Abidin, N. Omar, and Z. Idrus. Managing unstructured data in relational databases. In *2013 IEEE Conference on Systems, Process Control (ICSPC)*, pages 198–203, Dec 2013. doi:10.1109/SPC.2013.6735131.

[32] S. Zawoad and R. Hasan. Digital forensics in the age of big data: Challenges, approaches, and opportunities. In *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, pages 1320–1325, Aug 2015. doi:10.1109/HPCC-CSS-ICESS.2015.305.