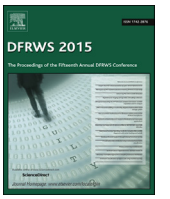
Digital Investigation 14 (2015) S127–S136

Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin



DFRWS 2015 USA

Privacy-preserving email forensics

Frederik Armknecht [a, *], Andreas Dewald [b]

[a] Universität Mannheim, Germany
[b] Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany

Keywords:
Forensics
Privacy
Keywords
Non-interactive searchable encryption
Dictionary attack

A B S T R A C T

In many digital forensic investigations, email data needs to be analyzed. However, this poses a threat to the privacy of the individual whose emails are being examined and in particular becomes a problem if the investigation clashes with privacy laws. This is commonly addressed by allowing the investigator to run keyword searches and to reveal only those emails that contain at least some of the keywords. While this could be realized with standard cryptographic techniques, further requirements are present that call for novel solutions: (i) for investigation-tactical reasons the investigator should be able to keep the search terms secret and (ii) for efficiency reasons no regular interaction should be required between the investigator and the data owner. We close this gap by introducing a novel cryptographic scheme that allows to encrypt entire email boxes before handing them over for investigation. The key feature is that the investigator can non-interactively run keyword searches on the encrypted data and decrypt those emails (and only those) for which a configurable number of matches occurred. Our implementation as a plug-in for a standard forensic framework confirms the practical applicability of the approach.

Introduction

Digital forensics denotes the application of scientific methods from the field of computer science to support answering questions of law (Carrier, 2005; Casey, 2011), which usually includes the acquisition, analysis and presentation of digital evidence. In such investigations, privacy is protected by individual domestic laws prohibiting unreasonable searches and seizures, such as the Fourth Amendment to the United States Constitution. Formally, privacy is a human right, which means – following the definition of the U.N. universal declaration of human rights – that "[n]o one shall be subjected to arbitrary interference with his privacy, […] or correspondence […] [and] [e]veryone has the right to the protection of the law against such interference or attacks" (International Community and U. N, 1948, Article 12). In many nations, domestic laws are very stringent and regarding investigations within a company further require companies to have works councils and data protection commissioners to protect the privacy of employees and customers. Especially the analysis of email data is subject to controversial discussions, especially when the private use of the corporate email account is allowed. It is in this duality between forensics and privacy where special requirements to forensic tools arised (Adams, 2008).

Motivation

Whenever a company becomes victim of accounting fraud or is involved in cartel-like agreements, this usually prompts large-scale investigations. In such investigations, mostly management documents and emails are reviewed, because especially communication data often provides valuable pieces of evidence. However, it also contains very sensitive and private information. As stated above, the company may be legally obligated by domestic law to

* Corresponding author.
E-mail address: armknecht@uni-mannheim.de (F. Armknecht).

http://dx.doi.org/10.1016/j.diin.2015.05.003

protect the privacy of the employees during the investigation as far as possible.

Current methods that are applied to this end usually make use of filtering to prevent all emails from being read: The investigator is allowed to run keyword searches and only those emails that contain at least some of the keywords are displayed. In this way, the investigator shall find such emails that are relevant to the case, but as few unrelated emails as possible to preserve privacy. The conceptually most simple realization is that the data remains at the company and that the investigator sends the search queries to the company who returns the search results. However, this comes with two problems. First, the company would learn all search queries while for investigation-tactical reasons, it may be necessary to keep them secret. Second, the need for constantly contacting the company's IT puts further burden on both sides. Therefore, common practice is that the investigator is granted access to the entire data and conducts the search. There, other privacy problems arise: Because of the risk to overlook the so-called smoking gun when not reading every email, investigators may be tempted to sidestep the filter mechanism. A prominent example of such deviation from the search order is the case United States v. Carey (1999), even though in this case no filtering was actually used at all.

Contributions

Our contribution is a novel approach for privacy-preserving email forensics, which allows for non-interactive threshold keyword search on encrypted emails (or generally text). That is a user holding the encrypted text can autonomously search the text for selected keywords. The search process reveals the content of an email if and only if it contains at least t keywords that the investigator is searching for. Otherwise, the investigator learns nothing – neither about the content of the email nor whether any of the selected keywords are contained. Our scheme allows for privacy-preserving email forensics using the following process:

1. Company's IT extracts requested email boxes.
2. Company (IT or data protection officer) encrypts the emails by applying our scheme.
3. Encrypted data is extradited to third party investigators.
4. Investigators are able to run keyword searches on the encrypted data and can decrypt those (and only those) emails that contain at least t of the keywords that they are looking for.

This way, the investigators are able to conduct their inquiry in the usual way, while not threatening privacy more than necessary. We want to emphasize that the investigator holds the entire data and can search within the data without further interaction with the company. In particular, the company learns nothing about the search queries executed by the investigators. For the investigator, nothing changes and keyword search can be performed as usual, because the decryption is handled transparently in the background after a successful keyword search, as demonstrated later in Section Practical implementation.

The only overhead introduced for the company is to determine the threshold t, i.e. decide how many relevant keywords must be matched for decryption, and to encrypt the email data. Besides choosing the threshold t, the company may further decide, which keywords should be relevant for the case (i.e. whitelisting keywords for which the investigator might obtain a match), or in the contrary decide which words shall not allow the investigator to decrypt emails, such as pronouns, the company's name and address, salutations or other simple words (i.e. blacklisting keywords). Note that both approaches can be applied in our scheme, however, we further refer to the blacklisting approach, as we assume it to be more practical in most cases, because a blacklist can easily be adopted from case to case once it has been built. A whitelist has to be determined strongly depending on the case, but there might be cases where it is worthwhile to apply the whitelist approach. After configuring the threshold t and the white-/blacklist (once), the company can just invoke the encryption algorithm on the email data and hand it over for investigation.

The proposed scheme uses only established cryptographic building blocks: a symmetric key encryption scheme, a hash function, and a secret sharing scheme. The facts that for each of these allegedly secure implementations do exist and that the security of the overall scheme solely relies on the security of these building blocks allow for concrete realizations with high confidence in the security.

Last but not least, we present an implementation of our scheme as a plug-in for the forensic framework *Autopsy* (Carrier, 2009). Experiments and performance measurements conducted on real email folders demonstrate the practical applicability of our approach.

Outline

The remainder of the paper is structured as follows: We introduce our solution and give a high level description of the scheme and algorithms in Section The proposed scheme. In Section Detailed description of the components, we provide an in-depth explanation of the scheme and Section Practical implementation shows our prototype implementation and evaluates the practical applicability. In Section Related work, we give an overview of related work and conclude our paper in Section Related work.

The proposed scheme

This section provides a description of our scheme, focusing on explaining the motivation behind the design, the overall structure and work flow. For readability, we omit several details on the concrete realization of the components. These will be delivered in Section Detailed description of the components.

Because an investigator should only be able to decrypt and read such emails that match his search terms, the investigator should not be able to learn anything about the encryption key of other emails when successfully decrypting one email. Thus, we use a cryptographic scheme that protects each email individually. The overall situation

with respect to one email can be described as follows: A plaintext P is given together with a list of *keywords* $w := \{w_1,\ldots,w_n\}$. In our context, P would be the email that needs to be protected and w denotes the set of words within this email. For security reasons, this list excludes trivial words like "the", or other general words that can be configured using the already mentioned blacklist. The scheme provides two functionalities: a protection algorithm *Protect* and an extraction algorithm *Extract*. The task of the protection algorithm is to encrypt the given plaintext P to a ciphertext C to prevent an outsider to extract any information about the plaintext. In addition the protection algorithm outputs some helper data. The helper data enables a user who knows at least t keywords to extract the plaintext P from the ciphertext C by applying the extraction algorithm.

The scheme uses three established cryptographic primitives: a symmetric key encryption scheme, a hash function, and a secret sharing scheme. We explain these 'on the go' when needed in our description.

The protection mechanism

The protection algorithm (see Algorithm 1) has the task to encrypt an email in a way that allows for a keyword search on the encrypted data. In a nutshell, it can be divided into the following three different steps:

Algorithm 1 The Protection Algorithm *Protect*

Input: A plaintext P, a set of n keywords $\mathcal{W}$, and an integer threshold $t \geq 1$.

Output: A ciphertext C, a mapping F (representing the helper data)

{**Step P.1: Encrypt Message**}

1: Sample a random key $K \in \{0,1\}^{\ell_K}$

2: Encrypt the plaintext P to ciphertext C under the secret key:

$$C := Enc_K(P)$$

{**Step P.2: Split the Secret Key into Shares**}

3: Invoke the *Split* algorithm on K with inputs t (= threshold) and n (= number of shares) to get n shares of K:

$$(s_1,\ldots,s_n) := Split(K,n,t)$$

where $s_1,\ldots,s_n \in \{0,1\}^{\ell_K}$

{**Step P.3: Connect Keywords to Shares**}

4: Create a bijective mapping F such that

$$F(w_i) = s_i$$

for all $i = 1,\ldots,n$.

5: **return** Ciphertext and mapping (C,F)

Step P.1: Encrypt Message. To protect the content of P, it is encrypted to C using an appropriate encryption scheme (line 2). To this end, a secret key K is used which is randomly sampled (line 1). If the encryption scheme is secure, the plaintext becomes accessible if and only if the secret key can be determined. The two remaining steps ensure that if a user knows t keywords in w, he can reconstruct K and thus decrypt the plaintext P from C.

Step P.2: Split Secret Key into Shares. Recall that we want only users who know at least t correct keywords from w (i.e. keywords that occur in the email and that are not blacklisted), to be able to extract P. This *threshold-condition* is integrated in this step, while in the subsequent (and last) step, the connection to the keywords in w is established. More precisely, in this step a secret sharing scheme (as proposed by Shamir (1979), for example) is used to re-encode the secret key K into n shares $s_1,\ldots,s_n$ (line 3). This operation, referred to as *Split*, ensures that (i) the secret key K can be recovered (using an algorithm *Recover*) from any t pairwise distinct shares $s_{i1},\ldots,s_{it}$ but (ii) if less than t shares are available, no information about K can be deduced at all. Observe that the number of shares is equal to the number n of keywords in w, that is the keywords within the email except those in the blacklist.

So far, only standard cryptographic techniques have been used in a straightforward manner. It remains now to chain the shares derived in this step to the keywords in w. This can be seen as one of the main novel contributions of the proposed scheme.

Step P.3: Connect Keywords to Shares. In this final step, a mapping F is constructed which maps each keyword w_i to one share s_i. That is $F(w_i) = s_i$ for all $i = 1,\ldots,n$. This ensures that a user who knows t distinct keywords $w_{i_1},\ldots,w_{i_t} \in \mathcal{W}$ can derive the corresponding shares $s_{i_1} = F(w_{i_1}),\ldots,s_{i_t} = F(w_{i_t})$ using F. Given these, the secret key K is recovered by $K := \text{Recover}(s_{i_1},\ldots,s_{i_t})$ that allows to decrypt $P := \text{Dec}_K(C)$. The calculation of this mapping F is more complicated and is explained in the details in Section Detailed description of the components.

The output of the protection mechanism is the ciphertext $C = \text{Enc}_K(P)$ and information that uniquely specifies the mapping F. Observe that F actually represents the helper data mentioned before. For efficiency reasons, it is necessary that F has a compact representation. In the next section, we now explain how a forensic investigator might reconstruct the plaintext of an email P from (C,F), if and only if this email contains the t keywords he is searching for.

The extraction mechanism

The task of the extraction mechanism (see Alg. 2)) is to allow an investigator to decrypt the encrypted email, if this email contains the terms he is searching for. Formally speaking, the mechanism allows a user who has knowledge of t correct keywords to extract the original plaintext P from (C,F). Like for the protection mechanism, the extraction mechanism is divided into three steps. Each step is more or less responsible for reverting one of the steps of *Protect*. This needs to be done in reversed order (i.e., the first step E.1 reverts P.3, and so on). Assume now that the ciphertext C and helper data F is given and the user aims to

recover the original plaintext using t keywords $w'_1, \ldots, w'_t$, which are the words he is searching. The three steps are the following:

Step E.1: Compute Possible Shares From the keywords $w'_1, \ldots, w'_t$, we compute t possible shares by applying the mapping F (line 2). That is $s'_i := F(w'_i)$ for $i = 1,\ldots,t$.

Algorithm 2 The Extraction Algorithm *Extract*

Input: A ciphertext C, a mapping F, t keywords $w'_1, \ldots, w'_t$
Output: A plaintext P'

{**Step E.1: Compute Possible Shares**}
1: **for all** $i = 1, \ldots, t$ **do**
2: Apply the mapping F to keyword w'_i to get a possible share, i.e.,
$$s'_i := F(w'_i).$$
3: **end for**

{**Step E.2: Determine Possible Secret Key**}
4: Given the t candidates for shares of K, the corresponding recovery algorithm of the underlying secret sharing scheme is applied to get a candidate for the secret key:
$$K' := Recover(s'_1, \ldots, s'_t).$$

{**Step E.3: Try to Decrypt Ciphertext**}
5: Apply the decryption algorithm to C with key candidate and get a plaintext candidate P':
$$P' = Dec_{K'}(C).$$
6: **if** P' is a valid plaintext **then**
7: **return** plaintext P'
8: **else**
9: **return** $\perp$
10: **end if**

Step E.2: Determine Possible Secret Key In this step, the user aims to recover the secret key K from the shares $s'_1, \ldots, s'_t$. To this end, he invokes the corresponding algorithm *Recover* from the secret sharing scheme on $s'_1, \ldots, s_t$ and receives a value K' (line 4). Observe that by assumption on *Recover*, it holds that only if those t keywords really occurred in the email and were not blacklisted, the output of F will be t correct shares of the secret key. Otherwise (when there were less than t or no matches in this email) K' is a random value and the user does not learn which of the used search terms were correct, if any. More formally, it holds that $K' = K$ if and only if $\{s'_1, \ldots, s'_t\} \subset \{s_1, \ldots, s_n\}$.

Step E.3: Decrypting the Encrypted Plaintext In the final step, the user aims to extract P from C. That is he computes $P' = Dec_{K'}(C)$. According to the common requirements on secure encryption algorithms, it holds that $P' = P$ if $K' = K$. Otherwise P' should be some meaningless data that is not obviously related to P. We assume an efficient algorithm that can reject false plaintexts (explained later) and outputs $\perp$ in such cases. Otherwise the final output is P. This means, in the last step the algorithm is able to determine if K' was valid and otherwise does not display the useless data to the investigator.

Detailed description of the components

In this section, we provide both, a generic description based on appropriate cryptographic primitives, but also suggest a concrete choice of algorithms and parameters. In principle the following three mechanisms need to be realized:

- An encryption/decryption scheme that allows to efficiently identify wrongly decrypted plaintexts (steps P.1, E.3)
- A secret sharing scheme (steps P.2, E.2)
- A method to create mapping F (steps P.3, E.1)

In the course of this section, we explain those three building blocks in the given order.

Encryption/decryption scheme

Reasonable choices are encryption schemes that (i) have been thoroughly analyzed over a significant time period and (ii) are sufficiently efficient. A good candidate is AES (Daemen and Rijmen, 2002), being the current encryption standard.

Although AES has been designed to encrypt 128 bit blocks only, it can encrypt larger chunks of data if used in an appropriate mode of operation like the Cipher Block Chaining (CBC) Mode (Bellare et al., 2000). That is a plaintext P is divided into several 128-bit blocks $P = (p_1,\ldots,p_\ell)$ with $p_i \in \{0,1\}^{128}$ for each i (if the bit size of P is not a multiple of 128, some padding is used at the end). With respect to the key size, the AES standard supports 128 bit, 192 bit, and 256 bit. As according to the current state of knowledge, 128 bit security is still sufficient, we propose to use AES-128. That is in the following the keylength is $\ell_K = 128$ and *Enc* refers to AES-128 in CBC-mode; other choices are possible.

Given a plaintext P, it is first transformed into a sequence $(p_1,\ldots,p_\ell)$ of 128-bit blocks. To allow for identifying wrongly decrypted plaintexts (line 6), we prepend a 128-bit block p_0 which consists of a characteristic padding *pad*, e.g., the all-zero bitstring pad $= \langle 0,\ldots,0\rangle$. Then, given a 128-bit key $K \in \{0,1\}^{128}$, the sequence of 128-bit blocks $(p_0,p_1,\ldots,p_\ell)$ is encrypted to $(c_0,\ldots,c_\ell)$. More precisely, the CBC-mode involves a randomly chosen initial value IV that will be chosen at the beginning. Then, the ciphertext blocks c_i are computed as follows:

$$c_0 := Enc_K(pad \oplus \mathrm{IV}) \tag{1}$$

$$c_i := Enc_K(p_i \oplus c_{i-1}), \quad i = 2, \ldots, \ell \tag{2}$$

The output of the encryption process is $(\mathrm{IV},c_0,\ldots,c_\ell)$. With respect to the decryption procedure, it holds that

$$pad := Dec_K(c_0) \oplus \mathrm{IV} \tag{3}$$

$$p_i := Dec_K(c_i) \oplus c_{i-1}, \quad i = 2, \ldots, \ell \tag{4}$$

Observe that also a legitimate user may be forced to re-try several selections of keywords until he is able to access the plaintext. Hence, it is desirable that the decryption procedure can be aborted as soon as possible if a wrong key is tried. This is the reason for the introduction of the characteristic padding *pad*. Given the encrypted blocks $(c_0,\ldots,c_\ell)$ and a key candidate K', a user can find out if $K' = K$ by checking (3). Only if this is fulfilled, the decryption process is continued. Otherwise, it is aborted and the next email in the data set is tested for a valid match of all the t search terms.

Secret sharing scheme

The concept of secret sharing is well known in cryptography since long. In our proposal, we adopt the secret sharing scheme (SSS) proposed by Shamir (1979). As its concrete form has impact on the realization of the mapping construction, we recall its basic ideas here:

Let $K \in \{0,1\}^{128}$ denote the secret key. In the following, we treat K as an element of the finite field $\mathrm{GF}(2^{128})$. Next, let the threshold value t be given and an integer n which should represent the number of shares (equally to the number of different, non-blacklisted keywords in the email). We aim to derive n shares $s_1,\ldots,s_n$ such that any choice of t shares allows for reconstructing K. To this end, let $n + 1$ pairwise different values $x_0,\ldots,x_n \in \mathrm{GF}(2^{128})$ be given. More precisely, the values $x_1,\ldots,x_n$ will be the result of computations explained in Section Creating the mapping. Then, x_0 is chosen randomly in $\mathrm{GF}(2^{128})\backslash\{x_1,\ldots,x_n\}$. These values will later play the role of x-coordinates.

Next, a polynomial $p(x)$ of degree $t - 1$ is chosen such that $p(x_0) = K$. Finally, the n shares are computed by $s_i := (x_i, y_i) \in \mathrm{GF}(2^{128}) \times \mathrm{GF}(2^{128})$ with $y_i = p(x_i)$. We denote this overall procedure as

$$(s_1, \ldots, s_n) := Split(K, x_1, \ldots, x_n, t) \tag{5}$$

Observe that each share represents one evaluation of the polynomial p at a different point. It is well known that a polynomial of degree $t - 1$ can be interpolated from any t evaluations. Hence, recovery is straightforward: Given t distinct shares $s_{i_j} = (x_{i_j}, p(x_{i_j}))$, the first step is to interpolate the polynomial $p(x)$. Once this is accomplished, the secret is determined by evaluating $p(x)$ at x_0, that is $K := p(x_0)$. However, if less shares are known or if at least one of them is incorrect (that is has the form (x,y') with $y' \neq p(x)$), the derived value is independent of K. Hence, this realization fulfills the requirements mentioned above. This procedure is denoted by

$$K := Recover(s_{i_1}, \ldots, s_{i_t}). \tag{6}$$

Creating the mapping

It remains to discuss how a mapping F can be constructed that maps the keywords $w_i \in \{0,1\}^*$ to the secret shares $s_i \in \{0,1\}^{256}$. We propose a mapping F that is composed of two steps as follows: The first step uses a cryptographically secure hash function $H:\{0,1\}^* \to \{0,1\}^{256}$. For example, one might instantiate H with SHA-256 (National Institute of Standards and Technology, 2002) or the variant of SHA-3 (Bertoni et al., 2011) that produces 256 bit outputs. In addition, a random value $salt \in \{0,1\}^*$ is chosen. Then, the first step is to compute the hash values h_i of the concatenated bitstring $salt||w_i$, that is

$$h_i = H(salt||w_i) \quad \forall i = 1, \ldots, n. \tag{7}$$

Observe that $h_i \in \{0,1\}^{256}$. In the following, we parse these values as $(x_i,z_i) \in \mathrm{GF}(2^{128}) \times \mathrm{GF}(2^{128})$. The values $(x_1,\ldots,x_n)$ will be the x-coordinates as explained in Section Secret sharing scheme. This requires that they are pairwise distinct. As each x_i is 128 bits long, it holds for any pair (x_i,x_j) with $i \neq j$ that the probability of a collision is 2^{-128} if the hash function is secure. Due to the birthday paradox, the overall probability for a collision is roughly $n^2/2^{128}$. This probability can be neglected in practice as n is usually by several magnitudes smaller than 2^{64}. In case of the improbable event that two values collide, this step is repeated with another value for *salt*.

Next, we assume that the secret sharing scheme has been executed (see eq. (5)) based on the values $x_1,\ldots,x_n$ determined by the results from (7). That is we have now the situation that n hash values $h_i = (x_i,z_i) \in \mathrm{GF}(2^{128}) \times \mathrm{GF}(2^{128})$ are given and likewise n shares $s_i = (x_i,y_i) \in \mathrm{GF}(2^{128}) \times \mathrm{GF}(2^{128})$. We complete now the construction of F. First, a function g is constructed which fulfills

$$g(x_i) = y_i \oplus z_i \quad \forall i = 1, \ldots, n. \tag{8}$$

We elaborate below how g can be efficiently realized. Given such a function g allows to realize a bijective mapping

$$G : \mathrm{GF}(2^{128}) \times \mathrm{GF}(2^{128}) \to \mathrm{GF}(2^{128}) \times \mathrm{GF}(2^{128})$$

$$(x, y) \mapsto (x, g(x) \oplus y)$$

This mapping G can be easily inverted and hence is bijective. Moreover, it maps a hash value $h_i = (x_i,z_i)$ to

$$G(h_i) = G(x_i, z_i) = (x_i, g(x_i) \oplus z_i) \tag{9}$$

$$G(h_i) \overset{(8)}{=} (x_i, (y_i \oplus z_i) \oplus z_i) \tag{10}$$

$$G(h_i) = (x_i, y_i) = s_i. \tag{11}$$

Now F is defined as the composition of H and G, that is

$$F = G \circ H. \tag{12}$$

Algorithm 3 summarizes the steps of F. As we suggest the use of a known and widely analyzed hash function H, we can omit its specification in the output of *Protect*. In fact, the only information that is necessary for evaluating F is the function $g(x)$. The realization that we propose below requires only n field elements which is optimal in the general case. As we suggest the use of $\mathrm{GF}(2^{128})$, this translates to a

representation using $128 \cdot n$ bits. The bijectivity of F ensures that only correct hash values map to valid shares.

Algorithm 3 The Mapping F

Input: A keyword $w \in \{0,1\}^*$, a hash function $H : \{0,1\}^* \to \{0,1\}^{256}$, some salt $salt \in \{0,1\}^*$, and a polynomial $g(x)$

Output: A possible share $s = (x, y)$

1: Compute $h := H(salt||w)$ with $h \in \{0,1\}^{256}$
2: Parse $h = (x, z) \in \mathrm{GF}(2^{128}) \times \mathrm{GF}(2^{128})$
3: Compute $s := (x, z \oplus g(x))$
4: **return** Possible share s

It remains to explain how g:GF(2^{128})→GF(2^{128}) can be efficiently realized. Observe that any mapping from a finite field to itself can be represented by a polynomial. Hence, an obvious approach would be to use the tuples $(x_i, y_i \oplus z_i)$ to interpolate a polynomial $p(x)$ of degree w such that $p(x_i) = y_i \oplus z_i$ for $i = 1,\ldots,n$. However, the asymptotical effort for interpolation is cubic in the degree (here n) and our implementations revealed that this approach is too slow even for the average case. Therefore, we derived another approach: The core idea is to split the input space (here: GF(2^{128})) into several distinct sets and to define for each set an individual 'sub-function'. That is given an input x, the function g is evaluated on x by first identifying the subset of GF(2^{128}) that contains x and then to apply the corresponding sub-function to x. This approach requires to derive the corresponding sub-functions which we accomplish by plain interpolation. However, as each set now contains significantly less values x_i than w, the overall effort is significantly reduced. To see why, let ℓ denote the number of sets and assume for simplicity that each set contains n/ℓ values x_i. Then the overall effort to interpolate these ℓ subfunctions of degree n/ℓ each is $\ell \cdot (n/\ell)^3 = n^3/\ell^2$. That is compared to the straightforward approach of interpolating g directly, our approach is faster by a factor of roughly ℓ^2.

For the splitting of GF(2^{128}), i.e., the set of all 128-bit strings, we suggest to use the k least significant bits where $k \geq 1$ is some positive integer.[1] More precisely, for a value $x \in \mathrm{GF}(2^{128})$, we denote by $x[1\ldots k]$ the first k least significant bits (LSBs). This gives a natural splitting of GF(2^{128}) into 2^k pairwise disjunct subsets. Two elements $x,x' \in \mathrm{GF}(2^{128})$ belong to the same subset if and only if their k LSBs are equal, that is $x[1\ldots k] = x'[1\ldots k]$. Let $X := \{x_1,\ldots,x_n\}$ denote the n values determined by (7). For $I \in \{0,1\}^k$, we define

$$X_I := \{x \in X | x[1\ldots k] = I\}. \tag{13}$$

This yields a separation of X into 2^k pairwise distinct subsets: $X = \dot{\cup}_{I \in \{0,1\}^k} X_I$. For each set X_I, a corresponding sub-function g_I is determined which fulfills

$$g_I(x_i) = y_i \oplus z_i \quad \text{for all } x_i \in X_I. \tag{14}$$

As already stated, this is accomplished by simple interpolation. That is if $n_I := |X_I|$ denotes the size of X_I, the function g_I can be realized by a polynomial of degree n_I which will be close to $n/2^k$ on average.

This completes the construction of g and hence of G as well. The description of g requires only to specify the sub-functions g_I, that is

$$g \mathrel{\widehat{=}} \left(g_{(0\ldots00)}, g_{(0\ldots01)}, \ldots, g_{(1\ldots11)}\right) \tag{15}$$

where each index represents a k-bit string. As each g_I is of degree n_I, it can be described by n_I coefficients in GF(2^{128}). Because of $\sum_{I \in \{0,1\}^k} n_I = n$ it follows that g can be fully specified by n field elements.

The evaluation of g is as explained above. Given an input $x \in \mathrm{GF}(2^{128})$, first $I := x[1\ldots n]$ is determined, and then $g_I(x)$ is computed which represents the final output. That is $g(x) := g_I(x)$ where $I := x[1\ldots n]$.

Practical implementation

In this section, we show the prototype implementation of the encryption tool and the integration of the search and decryption routines in the standard forensic framework *Autopsy* (Carrier, 2009) (Version 3) exemplarily. We further evaluate the performance of the approach on sample cases.

Realization

We want to sketch how our prototype implementation works. For brevity, we omit implementation details, as all the important algorithms are already explained in Sections The proposed scheme and Detailed description of the components and the implementation is straightforward.

The encryption Software is implemented platform independent as a Python script. The command syntax is as follows:

python ppef_gen.py<pst|mh|mbox|rmbox> <in> <out> <thr> <blist>

We support different inbox formats, like the mbox format which is amongst others used by Mozilla Thunderbird, the pst format that is used by Microsoft Outlook, the MH (RFC822) format and Maildir format as used by many UNIX tools. For encryption, the threshold parameter t needs to be specified and the blacklist can be provided as a simple text-file.

As for the encryption tool, the decryption software is implemented as a platform independent Python script. For better usability in real world cases, we also integrated it into Autopsy by implementing an Autopsy-plugin called PPEF (Privacy Preserving Email Forensics). To perform his analysis on the resulting encrypted ppef file, the investigator can just add an encrypted ppef file to a case in Autopsy as a logical file and in step 3 of the import process, select the PPEF ingest module in order to handle the encrypted data. After successfully importing a ppef file, the file is added to the data sources tree. When selecting the file, Autopsy first shows the random-looking encrypted data in the hexdump. The investigator can then use the PPEF search form to run keyword searches on the encrypted data using our previously explained *extract* algorithm.

[1] In our realization, k ranges from 1 to 8.

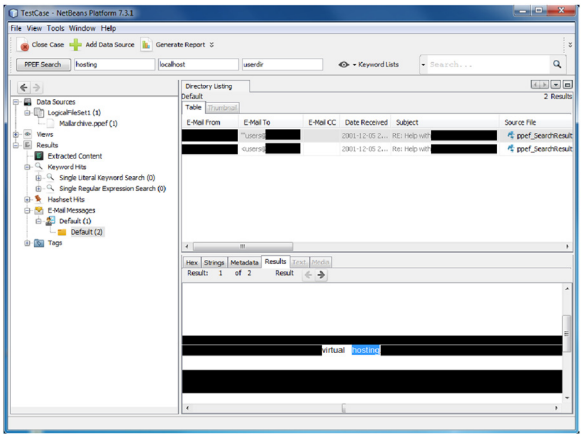


Fig. 1. Viewing the successfully decrypted email matches.

The search bar is shown in Fig. 1. Note that the number of keywords that have to be entered is obtained from the ppef file automatically. When clicking on the PPEF Search button, the searching algorithm is run transparently in the background and successfully decrypted emails (i.e. emails that match the entered keywords) are automatically imported to Autopsy. The emails are parsed using Autopsy's standard mbox parser, so that emails are added within the EMail Messages tree and can be filtered, sorted, and viewed using the standard mechanisms of the tool, as also displayed in Fig. 1 (the contents of the emails have been sanitized for privacy reasons). After shortly demonstrating the practical realization, we next use those prototypes to evaluate the performance of our solution.

Practical evaluation

In order to evaluate the practical usage of our scheme using the developed prototype, we use a data set of different real world email boxes. We evaluate the performance of the prototype for encryption, search and decryption. All measurements where performed on one core of a laptop with Intel® Core™ i5-3320M CPU at 2.60 GHz and 16 GB RAM.

The data set

Our data set consists of the following 5 mailboxes:

- The Apache httpd user mailing list from 2001 to today consisting of 75,724 emails (labeled *Apache*)
- One mailbox of 1590 work emails (*Work*)
- Three mailboxes of private persons with 511, 349, and 83 emails (*A, B, C*)

Blacklist

As example blacklist, the 10,000 most common words of English, German, Dutch and French (University Leipzip, 2001) are used. In a real case, the blacklist may of course be choosen less restrictive, but this is a generic choice used for our performance evaluation. The keyword threshold t was set to 3. In real cases, a more case specific blacklist must be created. It must consider words such as the company name, which likely can be found in every email, email signatures, and generally terms often used in emails, as long as they are not especially relevant to the case. We focus on the encryption scheme itself and consider building an appropriate blacklist to be a separate problem that is highly dependant on the specific case and can not be solved in general.

Encryption performance

Table 1 shows statistics about the time it takes to encrypt the sample emails. It also lists the total time it takes to encrypt each sample mailbox. The rate to encrypt the mailboxes is on average only around 13.5 emails per second. This value is still acceptable because the encryption needs to be done only once during an investigation since our approach is non-interactive.

Encryption storage overhead

Because the sole encryption of the emails does not add much on the size of the data, the main factor of storage overhead is the storage for the coefficients of our polynomials used to map hashes to shares. Their number is determined directly from the keywords, with a slight overhead introduced by padding the coefficients of the subfunctions to degree t. This introduces a total average overhead of 582.4 bytes for the polynomial coefficients, plus 16 byte salt per email. The overhead introduced by the AES encryption is between 33 and 48 bytes consisting of 16 bytes each for the IV and padding prepended to the plaintext to determine correct decryption and between 1 and 16 bytes PKCS#7 padding to the AES 128 blocksize. Altogether, the size increase in kilobytes of the raw emails within the mailboxes compared to our ppef format can be seen in Table 2. This leads to an average storage overhead of 5.2%.

Search and decryption performance

As the prototype always tries to decrypts each email, the search and decryption is not dependent on a keyword hit and hence the times listed for search in Table 3 include the decryption time of successful keyword hits. Because the rate the mailboxes are searched is on average only around 98 emails per second, searches take a while, but searches are still feasible.

Attack performance

We now discuss the effort needed by an attacker to execute a dictionary attack on an entire email box, such that on average, a fraction of π of all emails are decrypted

Table 1
Encryption performance of prototype.

	Apache [s]	Work	A	B	C
Min	0.004	0.005	0.005	0.005	0.005
Max	31.745	1.403	1.932	1.117	0.460
Avg	0.082	0.136	0.122	0.110	0.173
Med	0.072	0.115	0.101	0.074	0.150
σ	0.133	0.120	0.132	0.153	0.071
Σ	6243.511	217.242	62.842	38.535	14.367

Table 2
Search/decryption performance.

	Apache	Work	A	B	C
Size raw [KB]	376,551	418,680	16,386	47,486	6,676
Size PPEF [KB]	418,870	420,418	16,885	47,821	6,806
Overhead	11.2%	0.4%	3.0%	0.7%	1.9%

where $0 \leq \pi < 1$. We provide a detailed security analysis and a security proof in a technical report (Armknecht and Dewald, 2015) due to space limitations. This analysis shows that a higher level of security can only be achieved by restricting the possibilities of the investigator and that the strongest possible attack is a brute-force dictionary attack, as long as the underlying ciphers are secure. The effort of such an attack is expressed in the number q of trials an attacker needs to make. To this end, we use the following sufficient condition to decrypt a single email with probability $\geq \pi$:

$$q \geq \log_2(1-\pi)/\log_2\left(1-\frac{\binom{n}{t}}{\binom{N}{t}}\right) \tag{16}$$

This involves several highly context-dependent parameters, making a general statement impossible. But to give an intuition nonetheless, we estimate the attack effort in the cases considered in our evaluation. For each of the cases, Table 4 displays the effort for different scenarios: We considered the cases that each email contains the measured n_{avg} keywords for each mailbox. With respect to the dictionary size, we take as orientation the Second Edition of the 20-volume Oxford English Dictionary. It contains full entries for 171,476 words in current use (Oxford University Press). So, we first choose $N := 171{,}476$. For the success rate π, we chosen the value 0.99 as this ensures for an attacker with a high probability that almost all emails are decrypted. Using this parameters, we can calculate the number of tries q an attacker has to perform for the respective mailboxes. For example for the *Apache* mailbox, this results in $q = 4.17 \cdot 10^{12}$ queries to brute force a single mail. We multiply this by the number of mails in the mailbox and in order to give a better intuition, calculate the time needed for the attack based on the measured average query speed as presented in Section Search and decryption performance before. This leads to a time of $1.15 \cdot 10^{8}$ years needed for this attack on the *Apache* mailbox, as shown in Table 4. Of course, an attacker might reach some speedup by implementing the decryption software using the C language instead of python, as we did for our prototype. The fastest attack in our setting is the one on mailbox *C*, as this is the smallest mailbox in our data set and the attack would take 5,373.15 years.

Table 3
Search/decryption performance.

	Apache [s]	Work	A	B	C
Min	0.0090	0.0096	0.0098	0.0097	0.0098
Max	0.0598	0.1645	0.0139	0.1508	0.0148
Avg	0.0115	0.0137	0.0114	0.0123	0.0117
Med	0.0115	0.0117	0.0113	0.0113	0.0116
σ	0.0007	0.0103	0.0007	0.0086	0.0009
Σ	876.8591	21.7977	5.8650	4.2982	0.9750

As one might argue that not all those words in current use might really be used within a particular company and the attacker might have some kind of auxiliary information about which words are commonly used, for example from a sample set of already decrypted emails, we also calculate the effort for a worse scenario, where only 50% of the words in current use from the Oxford English Dictionary are used and the attacker exactly knows which words belong to this 50% so he can run an optimized attack with $N := 85{,}738$. The effort needed for this attack is obviously lower than before, as shown in the second line of Table 4. From those figures, we can see that in this case, the attack takes 808.74 years for our smallest mailbox *C* and even longer for the other ones.

Next, we consider an attacker that wants to decrypt a random half of the email corpus (i.e. he does not care, which emails are decrypted and which are not, just the number of successfully decrypted emails is important). If the attacker wants do decrypt a specific half of all the mails (those that contain some important information), the attack performance was identical to that of full attack from our first scenario, because due to our scheme, the attacker does not know which encrypted data belongs to which portion of the corpus prior to decryption. This means, the mails decrypted after this attack might not reveal the information an attacker is looking for, but it may be sufficient and obviously takes far less time, as shown in Table 4 in the third line for $\pi = 0.99$. We further consider a very strong attacker, who combines both previously mentioned attack scenarios: On the one hand, he has prior knowledge about the vocabulary used within the company, so he ran reduce his wordlist by 50%, and on the other hand, it is sufficient for him, if only half of the emails can be decrypted ($N := 85{,}738$ and $\pi = 0.5$). The respective figures are shown in the fourth line of Table 4.

Finally, we also want to evaluate what happens, if the attacker possesses even more prior knowledge about the words used within the emails. Therefore we set the number N of words needed for a successful brute force attack to 20.000, which corresponds to the vocabulary in daily speech of an educated person and 10.000 words, beeing the number of words used in the vocabulary of an uneducated person. If we combine this with the restrictions before, we see that the worst case drops to 0.16 years (1.92 months) to decrypt half of the mails of mailbox *C* that contains only 83 emails with a probability of 50% if the attacker exactly known which restricted vocabulary is used within the emails and none of those words has been blacklisted. For the larger mailboxes it still takes more than 3 years even in this worst case scenario. We consider this case to be rather unrealistic, because besides the high prior knowledge of the attacker, mailboxes in companies usually contain far more than 83 emails and we see that even for mailbox *B* with only 349 emails the attack is rather costly (3.5 years). Thus we feel that our solution provides good privacy protection even in very bad scenarios.

Table 4
Attack time in years to decrypt the entire mailbox with a probability of π using a wordlist with N words.

π	N	Apache	Work	A	B	C
0.99	171,476	$1.15 \cdot 10^8$	$3.26 \cdot 10^5$	$1.23 \cdot 10^5$	$1.17 \cdot 10^5$	5373.15
0.50	171,476	$1.73 \cdot 10^7$	49,072.84	18,565.34	17,638.94	808.74
0.99	85,738	$1.44 \cdot 10^7$	40,753.36	15,418.00	14,648.51	671.63
0.50	85,738	$2.17 \cdot 10^6$	6133.99	2320.64	2204.82	101.09
0.99	20,000	$1.83 \cdot 10^5$	517.23	195.68	185.91	8.52
0.50	20,000	27,510.37	77.85	29.45	27.98	1.28
0.99	10,000	22,843.44	64.64	24.46	23.24	1.07
0.50	10,000	3438.28	9.73	3.68	3.50	0.16

Related work

There is some related work regarding forensic analysis of emails, which describe the analysis of email data using data mining techniques (Stolfo and Hershkop, 2005; Stolfo et al., 2006). Others perform similar techniques on mail collections to identify common origins for spam (Wei et al., 2008). However, those works have in common that they work solely on the raw email data and do not consider privacy. Regarding privacy issues, there are some papers that argue broadly on different aspects of privacy protection in forensic investigations in general (Stahlberg et al., 2007; Aminnezhad et al., 2012; Caloyannides, 2004). Agarwal and Rodhain (2002) give a very good overview of privacy expectations especially with email in work environments. Hou et al. (2011a, 2011b) address a related area to our work when proposing a solution for privacy preserving forensics for shared remote servers. However, the techniques are very different to our work. First, they deploy public-key encryption schemes for encrypting the data which incurs a higher computational overhead and larger storage requirements compared to our scheme that uses secret-key schemes only. Second, the proposed solutions require frequent interaction between the investigator and the data holder while our solution is non-interactive. Similarly, Olivier (2005) introduces a system to preserve privacy of employees when surfing the web, while allowing to track the source of malicious requests for forensics. To the best of our knowledge, there has been no solution to ensure privacy in forensic email investigations by cryptographic means so far.

Regarding this work's cryptographic part, to the best of our knowledge, no existing solution is adequate for what we achieve. We, however, review directions that are the most related. First, searchable encryption schemes, e.g., (Song et al., 2000; Chang et al., 2005; Curtmola et al., 2006), encrypt files in such ways that they can be searched for certain keywords. More precisely, the scenario is that one party, the data owner, encrypts his data and outsources it to another party, the data user. The data owner now can create for keywords appropriate search tokens that allow the data user to identify the encrypted files that contain these keywords. The identified ciphertexts are given to the data owner who can decrypt them. Such schemes differ in two important aspects from the proposed scheme. First, most schemes enable search for *single* keywords while only few support conjunctive queries, e.g., see (Cash et al., 2013; Golle et al., 2004; Ballard et al., 2005). Second, they require interaction between the data owner and data user while in our scenario the investigator needs to be able to autonomously search and decrypt the encrypted files without requiring the data owner (e.g., the company) to do this.

Conclusion and future work

Summary

In the course of this paper, we described a modified process of forensically analyzing email data in a corporate environment. This process protects the privacy of the email owners as far as possible while still supporting a forensic investigation. In order to achieve this goal, the modified process needs to make use of a novel cryptographic scheme that is introduced in this paper. Our scheme encrypts a given text in such a way, that an investigator can reconstruct the encryption key and hence decrypt the text, if (and only if) he guesses a specific number of words within this text. This threshold t of the number of words that have to be guessed can be chosen when applying our scheme. As a further configuration of the scheme one can configure a list of words that shall not provide the key reconstruction. This kind of blacklist is meant to contain words that occur in almost every text, such as pronouns or salutations.

As a proof of concept, we implemented a prototype software that is able to encrypt email inboxes of different formats to a container file. To show the feasibility of the practical application of our approach, we further implemented a plugin to the well-known open source forensic framework *Autopsy* that enables the framework to handle the encrypted containers. The plugin features a search mask to enter keywords for search. The plugin then checks, which of the encrypted emails contain those words and can thus be decrypted and shows the matches within Autopsy. Using this prototype implementation, we finally measured both, runtime performance of encryption and searching/decryption, and the storage overhead introduced by our encryption scheme to show that it can be used in real cases. Thus, we can conclude that we introduced a novel, secure, and practical approach to enhance privacy in forensic investigations that can not only be applied to email analysis, but also to other areas of digital forensics, as further elaborated in the next sections.

Limitations

Caused by the design of our cryptographic scheme, there are some limitations, which we want to name here.

First of all, our scheme allows for keyword searches on the encrypted text, but hereby only reveals exact matches. Thus, our approach is prone to spelling errors and does also not allow to show partial matches or to use wildcards and regular expressions for the search queries, which is supported by other tools.

The second limitation of our scheme is that it is possible to run brute force/dictionary attacks against the encrypted data. As explained, this is an inevitable consequence of the scenario, i.e., of the fact that an investigator should be allowed to run keyword searches. To maximize the effort of an attacker, we ensured that the attacker has to try all combinations of t keywords out of his dictionary, because he is not able to learn which and if some of the already tried keywords were correct, if not all the keywords were correct. We have argued that the effort of a successful attack is sufficiently high in real cases. In order to render such attacks even more ineffective, it is important to choose a good blacklist when applying our method, or (depending on the case) even switch to the whitelist approach, if necessary.

Future work

As future work on this topic, we want to check the expandability of our cryptographic scheme in a way that supports the use of wildcards within the search keywords, making our approach more comfortable for forensic investigators without weakening the security of the scheme too much. Further, we want to apply our scheme to other areas of digital forensics, such as logfile or network traffic analysis.

Acknowledgments

We thank Michael Gruhn for his cooperation in this project.

References

Adams CW. Legal issues pertaining to the development of digital forensic tools. In: SADFE. IEEE computer society; 2008. p. 123–32. URL, http://dblp.uni-trier.de/db/conf/sadfe/sadfe2008.html#Adams08.

Agarwal R, Rodhain F. Mine or ours: email privacy expectations, employee attitudes, and perceived work environment characteristics. In: System sciences, 2002. HICSS. Proceedings of the 35th annual Hawaii international conference on. IEEE; 2002. p. 2471–80.

Aminnezhad A, Dehghantanha A, Abdullah MT. A survey on privacy issues in digital forensics. Int J Cyber-Secur Digital Forensics (IJCSDF) 2012; 1(4):311–23.

Armknecht F, Dewald A. Privacy-preserving email forensics. Tech. Rep. CS-2015-03. Department of Computer Science, University of Erlangen-Nuremberg; 2015.

Ballard L, Kamara S, Monrose F. Achieving efficient conjunctive keyword searches over encrypted data. In: Qing S, Mao W, Lopez J, Wang G, editors. ICICS. vol. 3783 of lecture notes in computer science. Springer; 2005. p. 414–26.

Bellare M, Kilian J, Rogaway P. The security of the cipher block chaining message authentication code. J Comput Syst Sci 2000;61(3): 362–99. URL, http://dblp.uni-trier.de/db/journals/jcss/jcss61.html#BellareKR00.

Bertoni G, Daemen J, Peeters M, Assche GV. The Keccak SHA-3 submission. January 2011. http://keccak.noekeon.org/.

Caloyannides MA. Privacy protection and computer forensics. Artech House; 2004.

Carrier B. File system forensic analysis. Boston, MA, USA: Addison-Wesley Pub. Co. Inc.; 2005.

Carrier B. The sleuth kit and autopsy: forensics tools for Linux and other Unixes, 2005. 2009. URL, http://www.sleuthkit.org.

Casey E. Digital evidence and computer crime: forensic science, computers, and the Internet. 3rd ed. Academic Press/Auflage; 2011.

Cash D, Jarecki S, Jutla CS, Krawczyk H, Rosu MC, Steiner M. Highly-scalable searchable symmetric encryption with support for boolean queries. In: Canetti R, Garay JA, editors. CRYPTO (1). vol. 8042 of lecture notes in computer science. Springer; 2013. p. 353–73.

Chang YC, Mitzenmacher M. Privacy preserving keyword searches on remote encrypted data. In: Ioannidis J, Keromytis AD, Yung M, editors. ACNS. vol. 3531 of lecture notes in computer science; 2005. p. 442–55.

Curtmola R, Garay J, Kamara S, Ostrovsky R. Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM conference on computer and communications security. CCS '06. New York, NY, USA: ACM; 2006. p. 79–88. URL, http://doi.acm.org/10.1145/1180405.1180417.

Daemen J, Rijmen V. The design of Rijndael: AES – the advanced encryption standard. Berlin, Heidelberg, New York: Springer Verlag; 2002.

Golle P, Staddon J, Waters B. Secure conjunctive keyword search over encrypted data. In: Jakobsson M, Yung M, Zhou J, editors. Proc.. of the 2004 applied cryptography and network security conference. LNCS 3089; 2004. p. 31–45.

Hou S, Uehara T, Yiu S, Hui LC, Chow K. Privacy preserving multiple keyword search for confidential investigation of remote forensics. In: 2011 Third international conference on multimedia information networking and security. Institute of Electrical & Electronics Engineers (IEEE); 2011a. URL, http://dx.doi.org/10.1109/MINES.2011.90.

Hou S, Uehara T, Yiu SM, Hui LCK, Chow K. Privacy preserving confidential forensic investigation for shared or remote servers. In: Intelligent information hiding and multimedia signal processing (IIH-MSP), 2011 seventh international conference on. IEEE; 2011. p. 378–83.

International Community, U. N. The universal declaration of human rights. 1948. URL, http://www.un.org/en/documents/udhr.

National Institute of Standards and Technology. FIPS 180–2, secure hash standard, federal information processing standard (FIPS). Publication 180-2. Tech. rep. DEPARTMENT OF COMMERCE; Aug. 2002. URL http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf.

Olivier M. Forensics and privacy-enhancing technologies. In: Advances in digital forensics. Springer; 2005. p. 17–31.

Oxford University Press, ???? The second edition of the 20-volume Oxford English dictionary. URL http://www.oxforddictionaries.com/words/how-many-words-are-there-in-the-english-language?q=171%2C476

Shamir A. How to share a secret. Commun ACM Nov. 1979;22(11):612–3. URL, http://doi.acm.org/10.1145/359168.359176.

Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Proceedings of the 2000 IEEE symposium on security and privacy. SP '00. Washington, DC, USA: IEEE Computer Society; 2000. p. 44. URL, http://dl.acm.org/citation.cfm?id=882494.884426.

Stahlberg P, Miklau G, Levine BN. Threats to privacy in the forensic analysis of database systems. In: Proceedings of the 2007 ACM SIGMOD international conference on management of data. ACM; 2007. p. 91–102.

Stolfo SJ, Creamer G, Hershkop S. A temporal based forensic analysis of electronic communication. In: Proceedings of the 2006 international conference on digital government research. Digital Government Society of North America; 2006. p. 23–4.

Stolfo SJ, Hershkop S. Email mining toolkit supporting law enforcement forensic analyses. In: Proceedings of the 2005 national conference on digital government research. Digital Government Society of North America; 2005. p. 221–2.

United States v. Carey. United States v. Carey 172 F.3d 1268 (10th Cir. 1999). 1999.

University Leipzip. Wortlisten. 2001. visited on Feb. 26, 2014. URL, http://wortschatz.uni-leipzig.de/html/wliste.html.

Wei C, Sprague A, Warner G, Skjellum A. Mining spam email to identify common origins for forensic application. In: Proceedings of the 2008 ACM symposium on applied computing. ACM; 2008. p. 1433–7.