Chapter 2

FORENSICS AND PRIVACY-ENHANCING TECHNOLOGIES

Logging and Collecting Evidence in Flocks

Martin Olivier

Abstract Flocks is a privacy-enhancing technology (PET) used to hide the web usage patterns of employees in an organization against profiling or mere inspection by administrators and other officials. However, Flocks is intended to support the identification of senders of malicious requests by means of a legitimate forensic investigation.

This paper formalizes what should be logged for an appropriate forensic investigation. Also, it considers exactly what evidence should be explored once a malicious request has been noticed. It argues that (i) evidence that would have been collected about a malicious request if the PET were not used, should still be collected, and (ii) evidence that becomes visible by some legitimate means because the PET is used, should be collected. However, information that has not become visible by such legitimate means, but is available because the PET is being used, should not be collected. In the latter case, privacy concerns override the fact that a malicious request might be uncovered by investigating more logged information. These positions are defended and formalized using mathematical notation.

Keywords: Privacy-enhancing technologies, logging, evidence collection

1. Introduction

The relationship between Privacy-Enhancing Technologies (PETs) and forensics has always been an uncomfortable one. The former is used to keep information about individuals private. The latter is used to uncover information about crimes or other relevant incidents. Often the information relevant to these two areas overlap: It is possible to argue that an individual who commits a crime gives up his or her right to pri-

vacy – at least as far as data about the crime is concerned. On the other hand, even if a demonstrable crime has been committed, the privacy of individuals who have not been demonstrated to be involved with the crime, should not be compromised. Two examples of this uneasy relationship include the infamous Clipper chip [19] and the `anon.penet.fi` case [8] that led to improvement of remailers to the point where they made forensic investigations extremely difficult [11].

We previously proposed a PET – referred to as Flocks – that allows anonymous web browsing in an organizational context [16]. It uses a number of proxies that randomly forward requests to one another, or – with some probability α – to the destination server outside the organization.[1] The intention was to propose a technology that provides an acceptable level of anonymous browsing, but one that allows a request to be associated with a sender under appropriate conditions. The previous paper [16] focused on the two primary parameters that influence the operation of Flocks should be chosen. The two parameters are α, the probability with which a request will be forwarded to the (real) destination server, and N, the number of proxies in the system. One of the explicit considerations to determine these parameters was the need to be able to conduct a forensic investigation. However, the exact information logged at each proxy for such an investigation was not considered.

The current paper has a dual purpose. First, it addresses the question of exactly what should be logged in Flocks to enable a forensic investigation. This question is answered by expressing the log contents using mathematical notation. Clearly, such a formal specification will contribute to the accuracy of analysis when the need for analysis arises. Second, the paper uses this formalized log content and explores which parts of it should be used (exposed) during a specific forensic investigation. From the discussion, the paper proposes a position that intends to allow legitimate forensic investigations, but preserves privacy to the greatest possible compatible degree once an investigation is underway.

The remainder of the paper is organized as follows. Section 2 describes the required background about PETs in general, and Flocks in particular. Section 3 considers logging in Flocks and formalizes the content of logs. Section 4 considers the information that should be uncovered during a forensic investigation and (formally) proposes the position that the paper supports. Section 5 presents the conclusions.

2. Background

Over the years a number of PETs have been introduced. Many of the current ideas were already present in Chaum's 1981 notion of a mix

[7]. In the mid 1990s attention turned to specific technologies used on the Internet (and, more specifically), the Web. This focus was based on the realization that interacting on the Internet often leaves a trail that may be used to learn more about an individual than should be tolerated. The PETs developed in the 1990s were mostly intended to allow the individual to exert control over what information is made known to other parties, by using appropriate intermediaries. These intermediaries could be (fixed) third parties, such as Anonymizer [6], Janus (or Rewebber) [22] or LPWA [10]. The third parties could also be (randomly or deterministically) selected from a set of available proxies or routers. Such ideas were used in Crowds [21] and Onion routing [12].

In more recent times, attention has turned to PETs that can be employed inside an organization to help the organization protect the information it has collected about individuals. Examples of developments in this regard include Hippocratic databases [1] and E-P3P [2, 14]. It has been argued that the costs (to the organization) associated with deploying such a PET will be fully recovered by customer satisfaction – and even lead to increased business opportunities [13, 20, 24]. In the case of Flocks a similar case could be made about the benefits employee satisfaction holds for the organization.

Flocks [16] was introduced as a PET based on technologies such as Crowds, but one intended for deployment within an organization. In this environment it was intended to minimize external traffic by caching web pages retrieved from the Internet as far as possible, but yet minimized an administrator's (or even a manager's) ability to breach the privacy of users by just browsing logs (or more actively profiling users). However, a fundamental tenet behind Flocks was the fact that the PET does not reduce users' accountability – and that forensic investigations should be possible where a legitimate reason exists for such an investigation. In essence, Flocks operates as follows. Each user operates a personal proxy and acts as trustee for the logs generated by that proxy. When a user submits a request, it is submitted to this proxy. The proxy serves the request from cache, if possible. Else it forwards the request to the external destination, with probability α. If it does not forward the request to the destination, it forwards it to another proxy that is chosen randomly. In the latter two cases, the result is cached (if possible), once it arrives. Requests from other proxies are dealt with similarly.

Anonymous remailers predate many of the other PETs mentioned here. Anonymous remailers have highlighted some of the most fundamental issues of collection of evidence weighed against the user's right to privacy [8, 11].

A system such as Flocks where logging is explicitly enabled assumes that these logs will be stored in a manner where they will not be abused. Such a PET will be useful when profiling of users and browsing of logs are the main threats. Where true anonymity is desired, a stronger form of PET will be required. The legal aspects of obtaining information from logs such as those maintained by Flocks have been discussed by others [5, 8, 9] and fall outside the scope of the current paper.

Many overviews of PETs have been published [9, 11, 17, 23]. For a structured view of PETs, see the Layered Privacy Architecture [15].

The computer forensic process is often divided into preparation, collection, analysis and presentation phases. For an overview of the forensic process, see the series of articles on the topic by Wolfe [25–30].

3. Logging in Flocks

The information to be logged by each proxy in a Flocks system has been implied in our earlier paper [16]. However, to conduct a proper forensic investigation in such a system, it is necessary to consider exactly what is logged. This section therefore formalizes this.

Consider some node $n_i \in P$ where P is the set of Flocks proxies in a given environment. Assume some message $m = \langle s, d, r \rangle$ arrives at some time t from some source s, where d is the (eventual) destination of the message and r is the request for a web page. The standard action for web proxies is to log $\langle t, s, d, r \rangle$. We will accept this standard behaviour as our initial solution.

Next, consider how this message is "rewritten" when it is forwarded. The request and destination remain unchanged. Expressed as it would be in Z-like notation, this could be written formally as $r' = r$ and $d' = d$. The proxy now acts as the source of the message; hence $s' = i$. The time of the new message is determined by the receiving system and (at best) we know $t' \approx t$. Note that clocks on proxies and servers will often (usually) differ significantly.

According to the way Flocks is designed, node n_i can forward the request (with probability α) to destination d, or to some node n_j. We argue that little will be gained by logging which choice has been made and, in the latter case, which node n_j has been selected. This will be discussed below when we discuss forensic investigations.

We represent the log at each node n_i as L_i where L_i is a relation over $T \times S \times D \times R$, where T is the set of possible times at which a request can occur (including date), $S = P$ is the set of nodes that could have (directly) sent or forwarded a message to the current proxy. Let D be the set of destinations; for simplicity we assume that $D \cap P = \emptyset$. R is the

set of possible requests; essentially R is the set of valid strings that form HTTP requests. We assume that the current practice of logging the first line of the full HTTP request is used – and hence that R is the set of such (potential) first lines. However, this can be expanded to include more of the request without affecting the remainder of the paper.

Below it will be useful to also consider an "augmented" log L_i^+ that consists of the quadruples of L_i, with the value i (the identity of the particular proxy whose log is being considered) added as a fifth column. Clearly, the domain of this column is the set of proxies P. However, for ease of reference we introduce another set $I = P$ as the domain of this fifth dimension. Therefore L_i^+ is a set of tuples from $T \times S \times D \times R \times I$.

Let us briefly consider the reliability of logged information. Clearly, s can be spoofed. However, given the fact that Flocks is cast in an environment where HTTP is used, the transport layer protocol used will be TCP. Therefore, a three-way handshake will be performed for any connection attempt [18]. What this implies is that host s has to have responded to the second part of the handshake before the request will be logged. This does not mean that the address of s cannot be spoofed, but that it will be somewhat harder to do this; in particular, if some node n_q masquerades as node s, n_q will have to be able to respond to appropriate messages to s; to some extent n_q will have to have taken control of the address of s. Therefore the log entry pointing to s will be somewhat more reliable than if UDP were used, but should still not be fully trusted. To address this problem it might be worth considering signed requests. Signed requests will n ot only help to determine the authenticity of the source of the request, but will also address non-repudiation: If an entry occurs in a log that states that the query has been received from another proxy, but no trace is found of the query in that proxy's log, it will be hard to determine which log is correct. Although signed requests will solve this problem, signing is not necessary to address the problems considered by this paper. We therefore leave signing of log entries for future research.

It has also been noted that the time logged depends on the particular server or proxy. This can be addressed to some extent by normalizing the times: If a "heartbeat" request is sent to each proxy at fixed intervals, and those heartbeats logged, such entries may be used to adjust to logged time to correspond in all logs to be used for an investigation. We do not consider this option further in this paper; in fact the paper will argue that time is not as important in this environment as it seems initially.

Given the logged information as described in this section, we now turn our attention to using this information for a forensic investigation.

```
10.229.33.5 - - [12/Mar/2002:14:00:01 +0200] "GET /default.ida?NNN
   NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
   NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
   NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
   NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090
   %u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u0
   0c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a HTTP/1.0" 400 32
   3 "-" "-"
```

Figure 1. A web server log entry caused by a Code Red request. The IP address of the source has been modified to disguise the identity of the attacking host.

```
10.18.67.106 - - [20/Jan/2005:09:20:33 +0200] "POST
  /cgi-bin/formmail/FormMail.pl HTTP/1.1" 404 299 "-" "-"
```

Figure 2. A web server log entry caused by an attempt to execute a (non-existent) FormMail script. The IP address of the source has been modified to disguise the identity of the attacking host.

4. Conducting a Forensic Investigation

4.1 Reasons for Launching an Investigation

The first issue to consider is why any forensic investigation will be conducted that requires access to the Flocks logs. There are essentially two scenarios. First, it is possible that some request that arrives at some external server warrants further examination. It is, for example, possible that the size (and other characteristics) of a message indicates that it is some attempt to attack this server. As an example, consider the web server log entry[2] given in Figure 1. This is typical of an entry that originated from a Code Red-infected machine. Figure 2 contains an example of an attempt to execute a `FormMail` script on a host that does not contain such a script. Anyone who has perused web server logs will know that such logs are typically littered with attempts to execute such scripts because of known vulnerabilities in them that can be exploited by an attacker. In such cases it could be necessary to trace the message to its origin – to disinfect the infected machine or identify the attacker.

The second scenario occurs when a suspicious message is noticed on the internal network. Such a message may either be noticed in one of the logs or be observed in transit on the network cables. The fact that Flocks is a PET does not mean that messages will not be intercepted on the wire or not seen in one of the logs: Flocks is intended to thwart automatic collection of profilable data in a single place and to prevent an administrator from learning much by just browsing the logs. Hence, if sufficient reasons exist to suspect inappropriate use of the network,

messages can indeed be intercepted. Many countries have laws that allow – or even require – the interception of messages under specific conditions. We assume that (social and technical) mechanisms exist that prevent recording of all messages, because such wide scale recording of messages will clearly render the PET useless. In particular do we assume that interception of messages are governed by an appropriate l aw and organizational policies that serve to protect the individual user.

As an example of this second case assume that an organization has a policy that employees are not allowed to visit pornographic websites. Monitoring the DNS lookup log files is a relatively unobtrusive way to verify that this policy is complied with. Suppose it is found that a DNS lookup occurred for `www.xx.co.za`. Suppose this site is known to contain pornographic material. The node that performed the DNS lookup will be the last proxy in the chain of proxies used. Examining that proxy's log file will indicate whether a full investigation is warranted, and the examination can proceed from there.

In both cases (internal and external) it is possible to precisely prescribe the log entries that are required for the investigation. In fact, without such a characterization of the entries required for the investigation, an investigation should be frowned upon – and the different custodians of the logs should make such searches – at least – very difficult to perform.

We assume that the log entries in an appropriate investigation can be characterized by a (possibly compound) condition c. In the case of an attack (such as the Code Red example used above), the request may be enough to characterise the log entries; here c will be of the form $R = \rho$ for some specific value ρ. In other cases a specific request sent to a specific host will be the cause of an investigation; hence, c will be of the form $(R = \rho)\&(D = \delta)$ for specific values of ρ and δ. More options exist for reasonable compositions of c; such conditions will, however, not be discussed in this paper.

Note that we will use relational algebra below to manipulate the log information. The R and D used in condition c above are used there to indicate the column that should be used when testing whether the condition is satisfied. In this context T, S, D, R and I will be used to identify the columns that have the (like-named) sets T, S, D, R and I as their domains.

In all cases it should be possible to identify the proxy from which the suspicious request has been sent. If the suspicious entry appears in an external log file, the last proxy that forwarded it will be identified as the sender. If the message is intercepted on the internal network, the sender at the point of interception will be one of the proxies. However,

we assume that this proxy is not part of condition c, but is identified separately as node x.

4.2 Investigation by Log Amalgamation

The first option to consider to investigate the incident is to amalgamate all the log files and search in the compound log file for suspicious log entries. Let Λ be this compound log file, with

$$\Lambda = \cup_{i \in I} L_i^+$$

The suspicious entries can then be isolated by selecting those for which condition c holds (with σ as the relational *select* operator):

$$\sigma_{c\&(I=S)}(\Lambda)$$

The additional condition $(I = S)$ ensures that requests are selected at their origin: Since we assume that all users run a proxy and send their own requests to their own proxies, all requests from node i will be logged with i as source in log L_i.[3]

There are, however, two privacy-related reasons why this solution is not acceptable.[4]

First, amalgamating all the log files is against the spirit of using multiple proxies in the first place: Once the logs have been amalgamated, much that does not concern the incident being investigated can be learnt from the log files. This may partly be addressed by performing an appropriate select prior to amalgamating the log files. This, however, still leads to the second reason why this solution is not acceptable.

This second reason is related to reasonable grounds to invade a person's communication privacy. The fact that evidence has been found that implicates one user does not imply that the browsing habits of all users who have submitted a similar request can necessarily be searched – and the query given above will find all such requests – and hence identify all originating senders. This does not mean that such a search is never warranted; however, we contend that it should not be the default mode of investigation. This is supported by RFC 3227 [4]: "Do not intrude on people's privacy without strong justification."

4.3 Investigating by Following the Chain

A more suitable approach will be one that starts at the implicated proxy and follows the chain back to the originating node. However, even this simple algorithm can be interpreted in different ways, and therefore needs to be formalized. Consider the following: Suppose log L_x indicates that it received the the suspicious request from some node y. If log L_y

indicates that the request was received from node z, one simply has to follow the link to L_z. However, it is possible that more than one "matching" request may exist in L_y. Suppose, for example, that the suspicious entry in L_x contains $e_1 = \langle t, y, d, r \rangle$, with y as the source (and t, d and r the time, destination and request — as used elsewhere in the paper). Now suppose L_y contains $e_2 = \langle t_2, z, d, r \rangle$. Note that d and r are identical in both entries. Does e_2 match e_1? Does the answer depend on the relationship between t and t_2? Clearly, if $t = t_2$ there is a case to be made that the two entries match. However, since different logs are based on different clocks, it is unlikely that the two times will be exactly the same. (It is even possible – albeit improbable – that an entry $e_3 = \langle t, z, d, r \rangle$ does not match e_1 exactly.) One solution will be to use $t \approx t_2$ as criterion, with some appropriate decision about how close two time values should be to be approximately equal.

Let us consider when two times are approximately close enough. If a page was requested by some node y from node x it usually means that y would not have requested the page immediately before this. (If it had, it would normally have cached it and would not have requested it again.) Similarly, if y requested a page at time t there will be no need to request it again soon after t (because y will still have it cached then). Hence, times do not have to be as accurate as in other [3] environments: t and t_2 may be considered approximately equal even if they differ by tens of minutes or more. (Obviously there are exceptions: Some pages cannot be cached and in some cases a page will be requested just prior to the expiration of the cache copy and just after that. However, these cases seem few enough to work with the general case until empirical evidence proves that a different approach is required.) [5]

4.4 Our Position

- Malicious requests that would probably have been noticed if the PET were not used, but were hidden by the PET, should be exposed for examination. The malicious requests exposed, should be limited to those that benefitted directly or indirectly from the noticed request.

- Malicious requests that are noticed because the PET is being used, should cause other requests that, in principle, could have caused the noticed malicious request to be issued, to be exposed. However, other requests (that is other from those that could have lead to the issuing of the malicious request) that benefitted from the fact that the malicious request has indeed been issued, should not be exposed.

It seems that the possible ambiguity in this description is unavoidable (unless the description is extended significantly). We therefore formulate it in mathematical terms.

POSITION 1 (UPSTREAM TRACKING) *Suppose that request ρ to destination δ from proxy x has been flagged. Let E be the set of log entries whose examination is justified. Then*

$$\sigma_{(D=\delta)\&(R=\rho)}(L_x^+) \subseteq E \tag{1}$$

and, if $\langle t, s, d, r, i \rangle \in E$ then

$$\sigma_{(D=\delta)\&(R=\rho)}(L_s^+) \subseteq E \tag{2}$$

Equation 1 ensures that the entries in the log of the implicated proxy (x) that could have forwarded request ρ to server δ are included. Equation 2 extends this recursively to include all the other entries earlier in the chain that, eventually, led to this final query.

This part of the specification will ensure that all the implicated entries are in E. The following restriction will ensure that no unnecessary entries are in E.

POSITION 2 (USE LIMITATION OF UPSTREAM ENTRIES) *Suppose that entry $e_1 = \langle t_1, s_1, d_1, r_1, i_1 \rangle \in E$. Then, if e_1 is not the entry that was directly implicated, there must exist an $e_2 = \langle t_2, s_2, d_2, r_2, i_2 \rangle \in E$ such that $d_1 = d_2 = \delta$, $r_1 = r_2 = \rho$ and $s_2 = i_1$. This clearly implies the existence of a chain of entries $e_1, e_2, \ldots, e_k$ with $k \geq 1$. The chain is terminated when the flagged entry is reached, i.e., k has to exist with $e_k = \langle i_k, t_k, s_k, d_k, r_k \rangle \in E$, $i_k = s$, with s the source from which the flagged request was sent, $d_k = \delta$ and $r_k = \rho$.*

Where Position 1 states that log entries should be followed towards their origin, Position 2 states that only log entries that could have lead to the request that caused the investigation, should be included.

The nodes that should be investigated are those from which the request originated, i.e., from $\Pi_I(\sigma_{I=S}(E))$, where Π is the *project* operator.

COROLLARY 1 (PROHIBITION OF DOWNSTREAM TRACKING) *As should be clear from the examples, Positions 1 and 2 imply that trails could be followed from the point of (valid) interception to its (possible multiple) "origins," but not towards its destination. By implication it cannot be followed from any point between the point of interception and the destination to other possible "origins." The formal requirements given above,*

already make provision for this. However, to stress the point consider two chains (as used above) $e_1, e_2, \ldots, e_k, \ldots, e_m$ *and* $e'_1, e'_2, \ldots, e'_n$ *with* $e'_n = e_m$. *Assume* e_k *is implicated, with* $1 \leq k < m$. *We argued that, although the evidence in* e_1 *and* e'_1 *are linked (via* $e'_n = e_m$*), this does not offer justification to use* e'_1. *Clearly, if other grounds exist,* e'_1 *can be used; such grounds will exist if there exists an* $e'_j = \langle i'_j, t'_j, s'_j, d'_j, r'_j \rangle \in E$ *such that* $i'_j = x$, $d'_j = \delta$ *and* $r'_i = \rho$ *and the communication has been observed legitimately at* x *– that is, it can be used if the evidence points directly to* e'_j *and* e'_j *points to* e'_1 *as its origin request.*

Arguably the last requirement is the most contentious of our position. However, we do not explore the implications that not accepting it will have on the investigation process in the current paper.

4.5 Time

Times at which requests have been issued have played a somewhat paradoxical role in our discussion thus far. On the one hand, it was assumed during the initial discussion that times can indeed be correlated – an assumption that was known to be unrealistic. However, in Positions 1 and 2 time played no explicit role. This was based on a number of arguments, the most important of which was the premise that the fact that a PET is being used, should not decrease accountability.

If the case can be made that time is not significant in cases where it is easy to correlate log entries, it clearly is possible to make a similar case where time is (more realistically) hard to correlate. Hence we accept that time is not one of the significant factors that will be used to correlate logs when conducting a forensic examination in Flocks.

However, time cannot be totally ignored: Should a malicious request that has just been reported, uncover a similar request that was issued a year ago? It is possible that the request issued a year ago was not malicious at the time, but changed circumstances make it appear malicious in the current environment. In addition to this possible objection, processing logs that stretch over years can be very expensive – especially when they can affect the results by introducing noise.

We therefore suggest that time plays a role when the logs for investigation are extracted. L_x should not normally be the log that was compiled over the entire lifespan of x, but some practical subset – one that is larger than apparently required for the investigation. We suggest that the time period to be taken into account should be determined at the start of the investigation and only changed if reasons for change are found. Empirical evidence is required to determine the effect of using generous log files for the investigation.

4.6 Importance of the Destination

In the example and positions above, the destination of the request was seen as an important facet of the investigation. It is, however, possible that a reported incident should not necessarily be associated with a given destination. A virus attack from some computer using the PET and that is noticed by some server δ does not mean that only those attacks that were launched on δ should be investigated; it will be prudent to investigate all attacks. The discussion should therefore be read as based on some (justifiable) condition c, that could include destination, request and other aspects – alone or in combination.

Therefore, if the destination is not a significant part of the investigation, the parts of the conditions using δ may be omitted in all the equations used in Section 4.

5. Conclusions

This paper considers issues that should be taken into account when a forensic investigation is conducted in the Flocks environment. It motivates what should be logged, and describes positions that govern the manner in which log file data is amalgamated for the purposes of the investigation. While these positions might be transferable to other PET environments, the argument depends on the caching that is done in Flocks. Therefore, care should be taken when generalizing these positions to other PETs.

Future work includes an empirical study on the effects of using logs for longer or shorter periods and the use of signed log entries to address spoofing and repudiation. It also remains to be formalized how the data collected should be analyzed to tie the evidence to a particular user, as well as appropriate presentation of this evidence.

Acknowledgements

This work is supported by the National Research Foundation under Grant 2054024, as well as by Telkom and IST through THRIP. Any opinion, findings and conclusions or recommendations expressed in this material are those of the author and therefore the NRF, Telkom and IST do not accept any liability thereto.

Notes

1. The analogy with Crowds [21] should be clear.
2. Note that the format of the server log entry differs in a number of respects from the proxy log entries discussed in the previous section; amongst others, the order of the fields are different.

3. If each node does not run its own proxy, originating hosts can be identified as those for which $s \notin I$.

4. This "solution" presents a third problem, but one that is easily solved by not allowing proxies to forward requests to themselves: In the original Flocks proposal, the possibility of a proxy forwarding a request to itself has not been excluded. If we use the suggested method to identify the origin of a message, forwarding by a proxy to itself cannot be allowed. Since such forwarding does not seem to be useful, we assume that it will not occur, and explicitly require from a Flocks proxy not to forward a request to itself.

5. For an example that illustrates some of the subtleties of matching entries, please see the extended version of this article, which is available at `http://mo.co.za/abstract/flfor.htm`.

References

[1] R. Agrawal, J. Kiernan, R. Srikant and Y. Xu, Hippocratic databases, *Proceedings of the Twenty-Eighth International Conference on Very Large Databases*, 2002.

[2] P. Ashley, S. Hada, G. Karjoth and M. Schunter, E-P3P privacy policies and privacy authorization, *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pp. 103-109, 2003.

[3] C. Boyd and P. Forster, Time and date issues in forensic computing – A case study, *Digital Investigation*, vol. 1(1), pp. 8-23, 2004.

[4] D. Brezinski and T. Killalea, Guidelines for evidence collection and archiving, RFC 3227, The Internet Society, February 2002.

[5] I. Brown and B. Laurie, Security against compelled disclosure, *Proceedings of the Sixteenth Annual Computer Security Applications Conference*, pp. 2-10, 2000.

[6] M. Caloyannides, Encryption wars: Shifting tactics, *IEEE Spectrum*, vol. 37(5), pp. 46-51, 2000.

[7] D. Chaum, Untraceable electronic mail, return addresses and digital pseudonyms, *Communications of the ACM*, vol. 24(2), pp. 84-88, 1981.

[8] G. Du Pont, The time has come for limited liability operators of true anonymity remailers in cyberspace: An examination of the possibilities and the perils, *Journal of Technology Law & Policy*, vol. 6(2), pp. 175-217, 2001.

[9] A. Froomkin, Flood control on the information ocean: Living with anonymity, digital cash and distributed databases, *University of Pittsburgh Journal of Law and Commerce*, vol. 395(15), 1996.

[10] E. Gabber, P. Gibbons, D. Kristol, Y. Matias and A. Mayer, Consistent, yet anonymous, web access with LPWA, *Communications of the ACM*, vol. 42(2), pp. 42-47, 1999.

[11] I. Goldberg, D. Wagner and E. Brewer, Privacy-enhancing technologies for the Internet, *Proceedings of the Forty-Second IEEE International Computer Conference*, pp. 103-109, 1997.

[12] D. Goldschlag, M. Reed and P. Syverson, Onion routing, *Communications of the ACM*, vol. 42(2), pp. 39-41, 1999.

[13] IBM, Privacy in a connected world (www-1.ibm.com/industries/government/doc/content/bin/private.pdf), 2002.

[14] G. Karjoth, M. Schunter and M. Waidner, Platform for Enterprise Privacy Practices: Privacy-enabled management of customer data, *Proceedings of the Second International Workshop on Privacy Enhancing Technologies*, 2003.

[15] M. Olivier, A layered architecture for privacy-enhancing technologies, *South African Computer Journal*, vol. 31, pp. 53-61, 2003.

[16] M. Olivier, Flocks: Distributed proxies for browsing privacy, in *Proceedings of SAICSIT 2004 - Fulfilling the Promise of ICT*, G. Marsden, P. Kotzé and A. Adesina-Ojo (Eds.), pp. 79-88, 2004.

[17] Organization for Economic Cooperation and Development (OECD), Inventory of privacy-enhancing technologies (PETs), Report DSTI/ICCP/REG(2001)1/FINAL, 2002.

[18] J. Postel, Transmission control protocol, RFC 793, Defense Advanced Research Projects Agency, Fairfax, Virginia, 1981.

[19] D. Price, Micro View - Clipper: Soon a de facto standard? *IEEE Micro*, vol. 14(4), pp. 80-79, 1994.

[20] PrivacyRight, Control of personal information: The economic benefits of adopting an enterprise-wide permissions management platform (www.privacyright.com/info/economic.html), 2001.

[21] M. Reiter and A. Rubin, Anonymous web transactions with Crowds, *Communications of the ACM*, vol. 42(2), pp. 32-48, 1999.

[22] A. Rieke and T. Demuth, JANUS: Server anonymity in the world-wide web, *Proceedings of the EICAR International Conference*, pp. 195-208, 2001.

[23] V. Seničar, B. Jerman-Blažič and T. Klobučar, Privacy-enhancing technologies: Approaches and development, *Computer Standards & Interfaces*, vol. 25, pp. 147-158, 2003.

[24] Wave Systems, User managed privacy: A new approach for addressing digital privacy and personal information on the Internet (www.wave.com/technology/PrivacyWhitePaper.pdf), 2000.

[25] H. Wolfe, Evidence acquisition, *Computers & Security*, vol. 22(3), pp. 193-195, 2003.

[26] H. Wolfe, Evidence analysis, *Computers & Security*, vol. 22(4), pp. 289-291, 2003.

[27] H. Wolfe, Encountering encryption, *Computers & Security*, vol. 22(5), pp. 388-391, 2003.

[28] H. Wolfe, Presenting the evidence report, *Computers & Security*, vol. 22(6), pp. 479-481, 2003.

[29] H. Wolfe, Forensic evidence testimony – Some thoughts, *Computers & Security*, vol. 22(7), pp. 577-579, 2003.

[30] H. Wolfe, Setting up an electronic evidence forensics laboratory, *Computers & Security*, vol. 22(8), pp. 670-672, 2003.