Processing Algorithms for Components within a Forensic Evidence Management System

K.K. Arthur and M.S. Olivier

Information and Computer Security Architectures Research Group
Department of Computer Science, University of Pretoria, South Africa
e-mail: karthur@cs.up.ac.za

Abstract

It is well established that the integrity and reliability associated with digital evidence is integral to the successful prosecution of digital crimes. Consequently, forensic specialists continue to employ investigative tools and processes that maintain the integrity of digital evidence throughout the investigation cycle. Understandably, such tool-sets and processes are often non-trivial, and can be improved upon.

As a contribution to such improvements, we present an architecture for a forensic evidence management system (FEMS), whose core components are a rule base, a knowledge base, an inference engine, and a data component. Given these system components, we develop a finite state automaton (FSA) to model the FEMS' general behaviour. In so doing, we demonstrate the interactions amongst these core system components. Ultimately, the purpose of the FEMS is to preserve the integrity of digital evidence, thereby improving the quality of investigative inferences made by forensic specialists.

In this paper we develop processing algorithms for the hypothesis state and the rule state described in our FEMS automaton. This elaboration is achieved through the use of flowcharts; we present the processing steps of these states, we present the input and output parameters of the transitions, and we provide the decision points that influence the probative value of the inferences within the FEMS.

Keywords

Forensic evidence, Evidence management, Data integrity, Information flow, Digital evidence

1. Introduction

Unlike crimes within the physical world, evidence within digital investigations is predominantly electronic in nature, volatile, and sparsely distributed, either within the target system, or within the network environment hosting the target system (Austen 2003, Hosmer 2002). Given these circumstances, the verification of the source(s) of a cyber incident is often non-trivial. Furthermore, several factors contribute to the expansive nature of the 'search area' within digital investigations – for instance, ever-increasing hard disk capacities.

To address the topic of data integrity, with reference to digital evidence, we propose the construction of a Forensic Evidence Management System (FEMS). The value in the FEMS lies in its ability to provide forensic specialists with a holistic view of the investigation landscape, and to contribute towards profiling the source(s) of an incident, thereby honing search activities within investigations. The FEMS would therefore aid in the efficient allocation and utilization of (limited) investigative resources – whether human, software, or instrumentation.

In this paper we illustrate the framework for our FEMS. Thereafter, we utilize a finite state automaton (FSA) to describe the FEMS behaviour; this is achieved by providing a mapping of the system components to the commensurate states within the automaton. The fundamental contribution in this paper is an expansion of the hypothesis and rule states within the FEMS FSA; this is achieved through the use of flowcharts. The flowcharts are utilized to depict the processing mechanisms for the hypothesis and rule components within the FEMS. In so doing, we establish some fundamental algorithms for the processing of information (by these components) within the FEMS.

The remainder of the paper is structured as follows: In section 2 we depict the framework for the FEMS. In addition, the states and transitions within our FEMS automata are illustrated and described. In section 3 we provide a set of assumptions under which the elaborations in the subsequent sections are based. Section 4 marks the beginning of the paper's core contribution – in this section flowcharts for the hypothesis process and the rule process are provided. Thereafter, we discuss each flowchart, with emphasis on the decision points and information flow control within them. In section 5 we provide the processing algorithms for the hypothesis and rule processes; these algorithms are extrapolated from the flowcharts generated in the preceding section. The paper is then concluded in section 6.

2. Background

As depicted in Figure 1, the FEMS consists of the following components: a system interface, a rule base, a meta-evidence base, an inference engine, an investigation logbook, a digital evidence base, and a generic knowledge base.

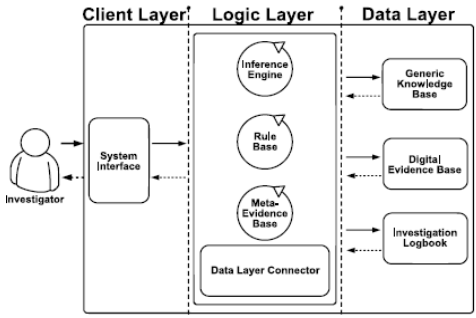

Figure 1: The Forensic Evidence Management System framework.

For the purpose of this paper we only describe the System Interface and the Rule Base components. However, the reader is encouraged to refer to (Arthur et al. 2007) for a comprehensive overview of the FEMS components. The System Interface provides the means for an investigator to access digital evidence – facts – within the evidence management system. This interface also enables an investigator to query, or test hypotheses against facts within the system. The Rule Base is a representation of the facts – knowledge – within the system, and the action(s) to be taken by the system, given any particular fact (Burns et al. 2001). For instance, a rule may specify a storage destination for router log files within the network environment.

The FEMS component processing algorithms in this paper are based on the FEMS' finite state automaton. Therefore, it is necessary to provide extensive detail with regards to the FEMS' FSA; the remainder of this section achieves exactly that. Furthermore, the remainder of the section is largely based on earlier work, published in (Arthur et al. 2008).

We begin by defining the following predicates in our FSA: S_i will represent evidence sources (or subjects), E_i will represent digital evidence (or objects), $I(x)$ will represent certainty values assigned to subjects or objects within our framework. That is, x can either be a claim, a fact, or an inference. We also define R_i to represent rules within the rule base. In all instances $i \in \mathrm{N}$.

Our FSA consists of four core states, namely the hypothesis, decision, rule, and data states. These states are mapped to the respective components within the FEMS architecture; that is, the System Interface, the Inference Engine, the Rule Base, the Generic Knowledge Base and the Digital Evidence Base respectively. Figure 2 provides an illustration of the FSA; definitions for the transitions within the state automaton are defined below. Thereafter, the rationale of each state and its transitions within the automaton is provided.

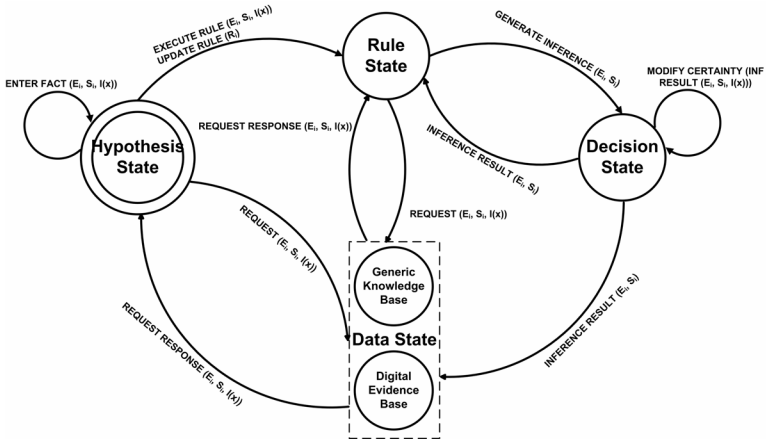

Figure 2: Finite state automaton depicting the FEMS' behaviour.

The FSA transitions are listed as follows: ENTER FACT(E_i, S_i, *I(x)*); EXECUTE RULE(E_i, S_i, *I(x)*); UPDATE RULE(R_i); GENERATE INFERENCE(E_i, S_i); INFERENCE RESULT(E_i, S_i, *I(x)*); MODIFY CERTAINTY(INFERENCE RESULT(E_i, S_i, *I(x)*)); REQUEST(E_i, S_i, *I(x)*); REQUEST RESPONSE(E_i, S_i, *I(x)*)

As depicted in Figure 2, the hypothesis state acts as the start and end state of our automaton. This is because initial hypotheses and final outputs are entered and returned to this state respectively. It may also be required that a fact be amended within an investigation; this is also achieved in the hypothesis state.

The rule state is entered when an investigator requires a rule to be executed on input data, or would like to amend a rule R_i. Based on the applied rule, actions are either triggered to the decision state, the data state, or both. The interaction between the rule and data states is a significant one. Certain rules may simply request information from the data state, while other rules may specify amendments to be made within the data state. The cyber crime profile forms the core of the rule base. That is, the cyber crime profile is encapsulated within this state. Therefore, a specific sequence of rules is defined, and must be executed before the crime profile is realized.

Once a rule is applied, an inference action is triggered. The inference action is executed within the decision state, where the certainty value of the evidence or source in question is updated accordingly. Where applicable, an inference result may be required to update a rule(s) within the rule state. For example, if the certainty associated with a log file is degraded by the fact that the log file has been tampered with, then rules applicable to the log file should be amended accordingly.

A significant action within the decision state is the *MODIFY CERTAINTY* action. This action ensures that all certainties within the inference base are updated whenever an inference result is generated. This action is iterative to ensure that the influence of any inference result on system objects is known at all times. Similarly to the rule state, there are instances where inference results may require an amendment to data. Using the example of a tampered log file, the hash value of the log file may be inconsistent with the hash value within the generic knowledge base; this would certainly need to be amended.

The data layer connector and the meta-evidence base were intentionally omitted within this elaboration. This was done in an effort to reduce overall complexity and transactions within the automaton. For this reason, the *REQUEST* and *REQUEST RESPONSE* actions are depicted directly between the hypothesis and data states.

The aim of this section was to provide an overview of the finite state automaton for the FEMS. Furthermore, the section has provided a backdrop upon which our later sections are based on.

3. Assumptions

In order to effectively demonstrate the processing algorithms and component interactions within the hypothesis and rule phases of the FEMS, the following assumptions are maintained for the remainder of this paper:

- The FEMS is applied within the context of a managed network environment, where all network components are known (to the FEMS), and are capable of generating and storing log evidence.

- The application of the FEMS is extendible to a number of environments, one of which is the Internet. However, due to the potential complexity of an analysis exercise, the application of the FEMS is limited in this work. For instance, within the context of the Internet, the FEMS would need to consider a number of evidence sources, and our analysis would need to incorporate factors inherent to this environment.

- We predominantly consider the analysis phase within an investigation. In so doing, we assume the pre-existence of evidence such as data integrity checksums, images of source evidence, and even log files, all of which are evidentiary artefacts collected prior to the analysis phase.

On the whole, these assumptions enable us to provide the processing algorithms, unconfined by the inherent details contained within a network environment.

4. FEMS component processing algorithms

We now develop on the hypothesis state and rule state provided in Figure 2. We make use of flowcharts to depict the flow of information and component interactions within these states, thereby providing an outline for the algorithms for these FEMS components.

In each of the following figures, a distinction is drawn between the manual activities within an investigation and the FEMS processing activities. Furthermore, the interaction between manual and automated activities within our model illustrates the necessity for human intervention and interpretation within any forensic investigation – these are tasks that cannot be discounted from investigations, even within an automated investigative system.

4.1. Hypothesis process flowchart

The hypothesis process flowchart is depicted in Figure 3. This flowchart has four distinct components: the initial decision phase, the information processing phase, the information update phase, and the final decision phase.

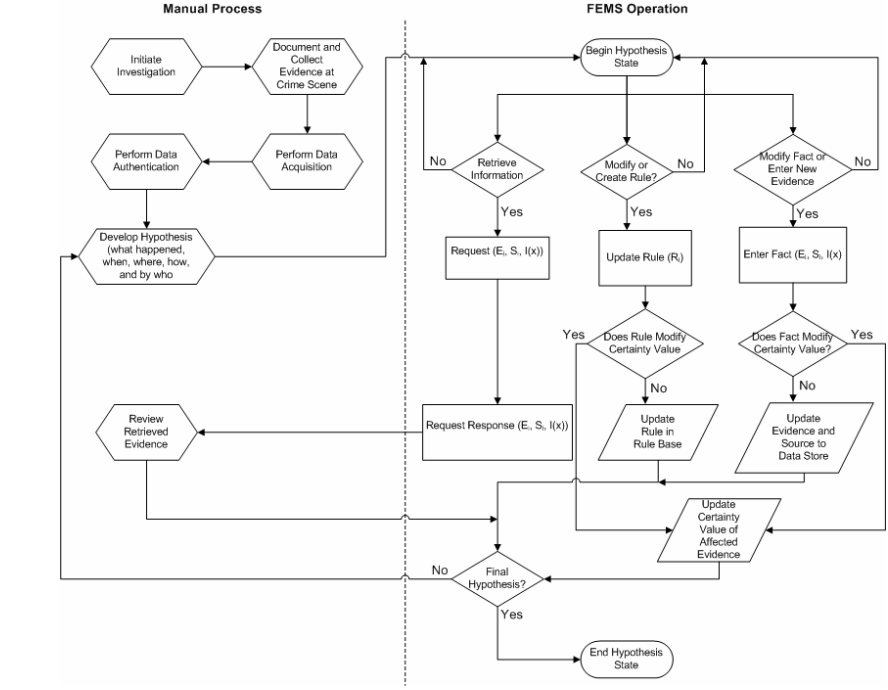

Figure 3: Flowchart for the Hypothesis process

The intentions of an investigator are established within the initial decision phase – that is, the system prompts the investigator on whether (s)he needs to retrieve information stored within the system, modify or enter facts (evidence) into the system, or to modify or create rules to be effected on evidence within the system (during the course of the investigation).

The processes within the information processing phase are executed after the investigator's initial actions are determined within the initial decision phase. As a result, auxiliary tasks within this phase may interrogate, retrieve, or update evidence within the data layer of the system. For example, in the instance where the investigative decision is to only retrieve information, the *Request Response* process would trigger an auxiliary task to the appropriate data layer component.

In the information update phase, the appropriate data components within the FEMS are updated with the investigators decisions, and or new information.

The final decision phase is initiated subsequent to the low-level activities resulting from the information processing phase; as depicted in Figure 3, whenever data is updated to the data layer, or returned to the system's user, a final decision is made on whether another iteration of the hypothesis phase is required by the system user.

It should be noted here – and in the following subsections – that references to the *update* of information or evidence within the flowcharts does not refer to the tampering of digital evidence. In our context, the word "update" is used to refer to information transformations within the FEMS' storage mechanisms, thereby enabling the system functionalities. For example, amendments to the integrity associated to evidence within the FEMS are deemed as amendments to meta-data within the FEMS' data store.

4.2. Rule process flowchart

As suggested in Figure 4, the rule process flowchart is typically activated within the data analysis phase of an investigation. The process begins where the system collates all the rules to be effected on the evidence within the FEMS, based on the investigative hypothesis at hand – this approach is consistent with the manner in which manual investigations are conducted, especially since there are specific consideration and evidence stores that are interrogated throughout the analysis cycle.

The sequential execution of the rule-set commences after the rule collation step. One of the more significant decisions within the rule process is the verification of whether an effected rule generates an inference result or not. Therefore, if a decision result is 'Yes', the certainty values of affected evidence within the FEMS would then be updated accordingly.

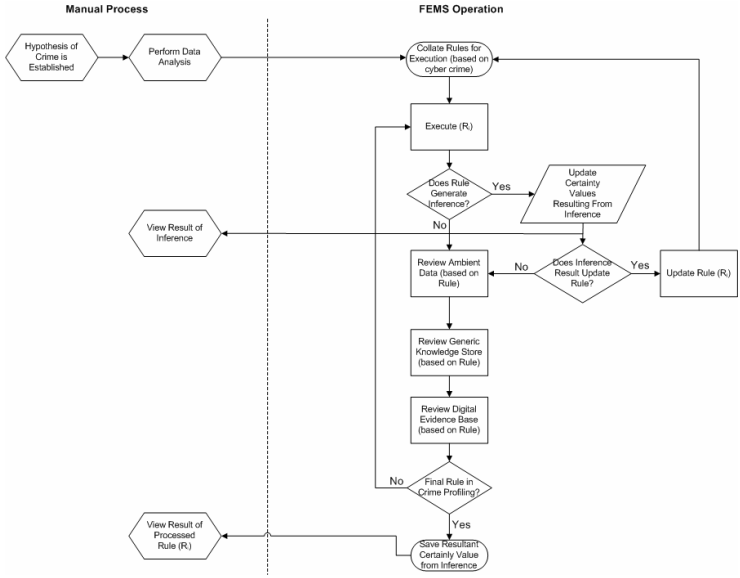

Figure 4: Flowchart for the Rule execution process

Furthermore, it is necessary to determine whether an inference result has influence on the rule currently in effect, or whether the inference result affects another rule within the rule base. If the result of this decision is affirmative, the relevant rule(s)

are adjusted accordingly, and control within the flowchart is returned to the collated rule-set, that is, the start state.

The change of information flow to the start state when a rule is modified is an essential one. For example, a rule may specify that all encrypted data files discovered on the storage media must be decrypted. However, such a rule may not be necessary (or even effected) if it occurs that no encrypted files are identified on the subject computer system. Alternatively, it may occur that the encryption strength applied on the identified files exceeds the capabilities of any augmented decryption solution employed within the FEMS.

Subsequent to the two initial decisions within the rule process, the *Review* processes enable the application of a rule within the significant stores of the disk image. Thereafter, the consecutive rule within the rule-set is established and executed.

In the following section we provide the Hypothesis process and Rule process algorithms.

5. Hypothesis and Rule process algorithms

The Hypothesis process and Rule process algorithms are depicted in Figure 5 and Figure 6 respectively. The algorithms are presented in pseudo-code – this approach was chosen to provide further glimpses into details yet to be developed.

```
1   boolean     final_hypothesis ← FALSE
2   int         task
3
4   Print ("Enter input character for the task to be performed")
5   Print ("Enter EOF character")
6   while (! final_hypothesis && (task = getchar()) != EOF)
7           switch (task)
8                   case    'Review Information'
9                           Print ("Enter evidence")
10                          Evidence ← getchar()
11                          Source ← getchar()
12                          Result ← REQUEST (E_i, S_i, I(x))
13                          if (Result ≠ NULL)
14                          then
15                                  Print Result to screen; break
16                          else
17                                  Print ("No data to return"); break
18                  case    'Rule'
19                          Display rule amendment interface
20                          if (certainty_value_updated(R_i))
21                          then
22                                  for all affected evidence in the Data Store
23                                          write new certainty value to Data Store
24                                          break
25                          else
26                                  Rule Base ← UPDATE RULE (R_i)
27                                  break
28                  case    'Modify Evidence'
29                          Display fact amendment interface
30                          if (certainty_value_updated(E_i))
31                          then
32                                  for all affected evidence in the Data Store
33                                          write new certainty value to Data Store;
34                                          break
35                          else
36                                  Data Store ← ENTER FACT (E_i, S_i, I(x))
37                                  break
```

Figure 5: Hypothesis process pseudo-code

A step through the algorithms reveals the flow of transaction and information within the system. Although we utilize a C-like syntax, the logic within the code is consistent with the flowcharts provided earlier. Of particular interest is the control of transactions within Figure 6; the reader will notice that the variable *i* is decremented within the code. The logic for this is that, if a rule (that is in effect) generates an inference, and that inference modifies any rules within the collated rule set, it is then necessary for all the rules within the rule set to be re-executed. Furthermore, the subsequent increment in *i* ensures that the logic is applicable to the first rule that is executed within the system.

```
1   int                 i
2   boolean Generate Inference(System Obj)
3
4   Collate rules for execution
5
6   while (i ≤ Rule Collection number)
7           EXECUTE (Ri)
8           if (Generate Inference (Ri))
9           then
10                  for all affected evidence(& sources) in the Data Store
11                          write new certainty value to Data Store
12                  if (Generate Inference (Ri) Updates Rx)
13                  then
14                          UPDATE RULE (Rx)
15                          i = i - 1
16                  endif
17          else
18                  Review Ambient Data based on Ri
19                  Review Generic Knowledge Store based on Ri
20                  Review Digital Evidence Base based on Ri
21                  i = i + 1
22          endif
```

Figure 6: Rule process pseudo-code

Although the algorithms provided within this section are not fully analysed, they provide a concrete development outline for the said components within the FEMS.

6. Conclusion

This paper focused on developing processing algorithms for the Hypothesis state and Rule state of our Forensic Evidence Management System (FEMS). The background section provided an overview of the FEMS framework and the FEMS state automaton. Thereafter, flowcharts depicting the flow and control of information within the hypothesis and rule processes were developed. Two significant limitations are noticeable from our use of flowcharts: firstly, significant details cannot readily be portrayed within the diagrams. Secondly, the use of natural language poses a risk – natural language is often subjective, unlike mathematical notation which is precise in its descriptions, and hence interpretation. Nevertheless, such flowcharts and processing algorithms provide the foundation for a future implementation of the FEMS (or subsets of the system).

7. References

Arthur, K.K., Olivier, M.S. and Venter, H.S. "Applying the Biba Integrity Model within a Forensic Evidence Management System". In P. Craiger and S. Shenoi (eds), *Advances in Digital Forensics III*, pp. 317 – 327, Springer, 2007.

Arthur, K.K., Olivier, M.S., Venter, H.S and Eloff, J.H.P. "Considerations Towards a Cyber Crime Profiling System". In S. Jakoubi, S. Tjoa and E.R. Weippl, *Proceedings of AReS 2008 - The Third International Conference on Availability, Security and Reliability*, pp. 1388 – 1393, IEEE, 2008.

Austen, J. "Some stepping stones in computer forensics". *Information Security Report*, Volume 8, Issue 2, pp. 37 – 41, September 2003.

Burns, L., Hellerstein, J., Ma, S., Perng, C., Rabenhorst, D. and Taylor, D. "Towards discovery of event correlation rules". *Proceedings of the IEEE/IFIP International Symposium on Integrated Network Management*. pp. 345 – 359, 2001.

Hosmer, C. "Proving the Integrity of Digital Evidence with Time". *International Journal of Digital Evidence*, Volume 1, Issue 1, pp. 1 – 7, 2002.