

REPORT DOCUMENTATION PAGE				<i>Form Approved OMB No. 0704-0188</i>	
<small>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</small> PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 31-10-2011		2. REPORT TYPE Final Technical Report		3. DATES COVERED (From - To) 05 Mar 2009 - 31 Oct 2011	
4. TITLE AND SUBTITLE Probabilistic Redaction				5a. CONTRACT NUMBER FA8750-09-C-0006	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Loughry, Joe				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Lockheed Martin Information Systems and Global Solutions (LM IS&GS) P.O. Box 277004 Littleton, CO 80127				8. PERFORMING ORGANIZATION REPORT NUMBER PRR-RPT-12729-000	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/RIEB 525 Brooks Road Rome, NY 13441-4505				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT DISTRIBUTION STATEMENT C. Distribution authorized to U.S. Government Agencies and their contractors; Critical Technology; 21 July 2011. Other requests for this document shall be referred to AFRL/RIEBB, 525 Brooks Road, Rome, NY 13441-4505, Attn: Michael J. Mayhew, CDIS Group Program Manager; phone: (315) 330-2898 DSN: 587-2898.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT An automated interactive redaction assistant prototype was developed. Based on the Web 1T 5-gram corpus, a list of every unique word and phrase in the English language, up to five words in length, that were observed on the World Wide Web and collected by Google, Inc. in 2006, the CLOAK system automatically flags candidate words, phrases, sentences, and paragraphs in documents under review that are likely classified and suggests redactions to make the document unclassified. Security classification guidance from more than one guide at a time is figured into each suggested redaction. The probabilistic aspect of operation is in the way the system prioritizes its suggestions according to the measured rate of occurrence of words and phrases observed in the language's current usage, avoiding low likelihood phrases in favor of more probable interpretations. Collocation correlation of phrases in security classification guides has automatically discovered missing, incomplete, and inconsistent rules in existing security classification guides, promoting forward quality improvement of security classification guides. Performance from the earliest prototype software, implemented in Perl using associative arrays, was surprisingly acceptable and suggests that an alternative					
15. SUBJECT TERMS Redaction, Linguistic Corpus, Probabilistic, Prototype, Security, Classified Documents, Security Classification Guide, PDF, Collocation Correlation					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 27	19a. NAME OF RESPONSIBLE PERSON Joe Loughry
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) 303-221-4380

TABLE OF CONTENTS

Section	Page
List of Figures	ii
List of Tables	iii
1.0 SUMMARY	1
2.0 INTRODUCTION	1
3.0 METHODS, ASSUMPTIONS, AND PROCEDURES	1
3.1 Concept	1
3.1.1 The Web 1T 5-gram Corpus	2
3.1.2 Normalization of the Corpus.....	2
3.1.3 Sources of Redaction Errors	2
3.2 Methodology	3
3.2.1 Planned Implementation	4
3.2.2 Actual Implementation.....	4
3.2.3 Complex File Formats.....	5
3.2.4 Model Security Classification Guide	6
3.2.5 Acknowledgements	7
3.2.6 Metrics	7
3.2.7 Statistical Analysis of Testing Results.....	8
3.3 Capabilities Achieved	9
4.0 RESULTS AND DISCUSSION	9
4.1 Demonstration and Delivery	9
4.2 Performance	10
4.2.1 Improvements to Performance	11
4.2.2 Alternate Network Representation.....	12
4.2.3 Test and Evaluation of Prototype.....	14
5.0 CONCLUSIONS	15
6.0 REFERENCES	16
APPENDIX – PUBLICATIONS AND PRESENTATIONS	18
LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS.....	21

LIST OF FIGURES

Figure		Page
1	Error Rate Minimization at Threshold T Based on Probability Density Functions.....	8
2	Main Screen of Graphical User Interface	10
3	Alternate Network Representation.....	13
A-4	Technical Council Poster	18
A-5	Data Sheet (front side)	19
A-6	Data Sheet (back side)	20

LIST OF TABLES

Table		Page
1	Phases of the Project	2
2	Acknowledgements	7

1.0 SUMMARY

This is the final technical report from the Probabilistic Redaction project. The report describes the concept inspired by the publication of Google, Inc. research database, the several technical approaches that were tried over the course of the project, the successful implementation, testing, and demonstration of a prototype redaction assistant application, and the technical challenges overcome. The planned design and actual implementation are compared. Measured performance numbers are presented alongside a statistical criticism of the test results, and indicating directions for further research and development of the system.

2.0 INTRODUCTION

The purpose of the project was to develop a new system for assisting human analysts with the task of redacting classified documents. The technical approach was to write new software that would use a Google-originated research database along with available security classification guides to find classified words and phrases in target documents with more flexibility and at a higher rate of success than could be achieved with conventional string matching methods. The system could be used by several analysts concurrently, it would have a Graphical User Interface (GUI) with good usability, and would support several different—nominally, twenty—Security Classification Guide (SCG) settings at the same time. It was hoped in this way to improve the throughput over manual methods by thirty to a hundred times while simultaneously reducing errors resulting in spills because of the inherent presence of automatic security checks. The concept was presented as a high technical risk, conceptual research effort with a good chance of failure. What was found actually was that the planned technical approach partly worked, that a variation of the original technical approach was surprisingly successful and fast, usability of the GUI was independently assessed as excellent, and the concept was partially validated.

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

The technical approach planned from project initiation employed a novel repurposing of the “Web 1T 5-gram Corpus” published by the Linguistics Data Consortium (LDC) at the University of Pennsylvania [1]. The corpus was assembled by Google, Inc. in 2006 from a total sampling of text found on the World Wide Web (WWW) in English and was intended for use in linguistics research. This data set was the only data or software that was purchased over the course of the project, not counting the operating system of the target hardware.

3.1 Concept

The principal investigator hypothesized that, representing as the Web 1T 5-gram corpus did a large instantaneous snapshot of mixed colloquial and formal written English containing current usage patterns and vocabulary, the Google database might be re-purposed to improve the performance of an automatic redaction assistant for human analysts. The approach is referred to as probabilistic because it uses the Web 1T 5-gram corpus as a guide, by reasoning from the measured relative incidence of numerically ubiquitous phrases, to find the most likely-to-occur matches at the level of words, phrases, and sentences in a classified document to one or more

tuples originating in a security classification guide, skipping over the much larger universe of low-probability phrases such as would be generated by a Markov model of the text.

3.1.1 The Web 1T 5-gram Corpus. Every unique five-word sequence—called *5-grams*—was counted by Google in preparing the corpus, followed by all the 4-grams, 3-grams, and 2-grams, analogously and respectively, terminating with a set of 1-grams, or individual words. The data set as distributed comprises a sorted list of 1-grams—which are case sensitive—together with several lists of pointers into the 1-gram list that encode the higher-order n -grams. The size of the data set as delivered was approximately 22 gigabytes (GB) on six DVD discs; the uncompressed size was 79 GB in 332 files in 27 directories. The smallest uncompressed file was 14 bytes; the largest uncompressed file was 334 megabytes (MB). Approximately 13.5 million unique “words” appear in the database, but this includes variable capitalization, misspellings, and numerous DNA sequences that are not English words.

3.1.2 Normalization of the Corpus. For the purpose of this project, the extraneous data were filtered out by a normalization process, yielding a shorter case-insensitive list of 1-grams that were then collated with a trimmed version of the GNU Moby Thesaurus to yield the base data set. This process requires a few hours on the target hardware, with performance assessed to be I/O bound during normalization. A thesaurus was found to be necessary in practice after the project proceeded to annotation of the database with key-value pairs derived from security classification guidance, especially later on when collocation correlation was attempted. The thesaurus was chosen from a range of available options because it was distributed in a form ideally suited for processing by the tool, it was free for use, and the list of synonyms it contained was comprehensive. In fact, the thesaurus was too comprehensive; for performance reasons, it was trimmed before use to eliminate very long words from the synonym list; this optimization was found to work well in practice.

Table 1. Phases of the Project

Phase	Dates of Phase	Purpose and Activities
I	March–September 2009	Research: gathered data, tools; experimentation
II	October 2009–September 2010	Prototype: working command-line prototype; design of GUI prototype
III	October 2010–September 2011	Development: GUI prototype developed and tested; documentation, demonstration, testing

3.1.3 Sources of Redaction Errors. Research was done into known sources of redaction errors and current recommendations for best practice, to inform the design of the prototype. There are some interactions that exist between design choices in the redaction method and attack potential against redacted documents that point to possible defenses against reverse analysis of redacted documents. Even when extreme technical precautions, such as scanning and re-printing the document, are taken to minimize the amount of residual metadata remaining in the bitmap, large-scale releases of documents or parallel releases from large organizations introduce a new hazard particular to the situation: human factors. Aside from the familiar redaction risk of certain or probable deduction of the unique identity of a missing word from typeface metrics and a list of dictionary words, one article looked at what can be “infer[red] from the length of the redacted material” (emphasis in original) [2]. A mistake of the type illustrated: the redaction appearance

of text published elsewhere led to the author of the book speculating about meta-inference of the content of the declassification guidance itself, in addition to leakage of redacted information. This is related to other discussion in open sources on the topic of redaction failures resulting from inference on missing items in sorted lists [3][4]. Echoing the cribs that enabled British cryptographers to break the German “Lorenz” cipher in 1941–2, “...if similar documents are being censored by *different* redactors, you’re apt to get the worst of both worlds—many pieces of the puzzle left exposed in one document or another, sufficiently parallel in structure to make them mutually completing, with the potential significance of each one highlighted by its absence from the others” [emphasis in original] [5]. Specific vulnerability mitigations possible against this attack include hashing for detection of duplicated content at the sentence and paragraph level, much as deduplication works in SAN file systems. It would need to be paired with workflow prioritization to route duplicate products to the same analyst, and the effects on security, accuracy, and throughput would be complex due to the positive influence on holistic grasp of material by a specializing analyst, negative influence of fatigue, technical opportunity for deduplication, and the risk/benefit calculation associated with avoidance of a second pair of eyes on some material in exchange for improved throughput and avoidance of the risks illustrated in [6]. Regardless, workflow prioritization is not an included feature of the tool in Phase III.

Our design accommodates a flexible defense against the vulnerabilities identified by Naccache and Whelan arising from typeface metrics and dictionary attack against the bitmap of a re-scanned document [7]. The successful attack described in [8] and further discussed in Zawinski (*op. cit.*) is rapidly mitigated by a larger granularity in a way reminiscent of the knapsack problem at the heart of the Diffie–Hellman key exchange. Conversely, rescanning, widely believed to be the most secure against residual information at the cost of hindering text search, introduces a further vulnerability when it is done at high resolution—as would be done to facilitate OCR conversion to re-enable full text searching—as font metrics, digitization, anti-aliasing and sub-pixel rendering artifacts robustly leak information about word length through the redaction barrier. The obvious defense against word-length dictionary comparison is use of variable-width typefaces and replacement of extended passages with ellipses, as discussed in [9].

A survey of redaction failures unattributable to any specific vulnerability is given in [10]. The author of the paper studied a 1.8 million document non-random sample of the 500 million document PACER database of U.S. Appellate, District, and Bankruptcy court records. Entries in PACER are redacted for personally identifiable information and commercial trade secrets. Searches were performed using specialized software for filled polygons indicating likely redactions in PDF files; approximately 2000 such files were found in addition to 3500 documents containing strings of X characters that were identified but not examined, because the methodology required looking under redaction rectangles for residual information. According to the paper, a few hundred such redaction failures were found and verified by a human. The risk of redaction failure through this familiar mechanism arises from a design decision in the system to use a vulnerable method in exchange for the advantage of not changing the document layout between original and released information. The impetus is believed to be a requirement of some users, notably court filings which have rigid formatting requirements.

3.2 Methodology

Because the present research project was intended to be speculative from the outset, it may be

advantageous to consider the methodology as having two aspects: firstly, the hypothesis as originally envisioned, and secondly, the actual system as it was eventually realized. The project was split into three phases (refer to Table 1). The first phase, lasting from March through the end of September 2009, was used for research and experimentation on how to handle the Web 1T 5-gram corpus and selection and evaluation of tools and algorithms for processing linguistic corpora. The second phase, running from October 2009 through the end of September 2010, was used for prototyping, ending with a working prototype with a command line interface and a design for a multi-user tool with a graphical user interface. The third phase, beginning in October 2010 and extending through the end of September 2011, developed the software into a practical tool with a usable GUI, and demonstrated it to several groups of people.

3.2.1 Planned Implementation. It was first thought that the corpus would be expanded in real memory into a tree structure, and then decorated with pointers to key-value pairs derived from automated or semi-automated analysis of classification guidance documents and stored in virtual memory. The size of the target hardware given in [11] was determined by modeling this technical approach. It was believed from the outset that the best algorithm to use for rapidly searching such a data structure would be the map-reduce algorithm [12], probably represented by the Hadoop implementation of map-reduce [13]. Protocol Buffers were investigated as an implementation option for serializing the data structures used in the tool—believed important for check-pointing the system during maintenance periods and for quickly restarting after changes to the hardware without needing to completely rebuild the data structures in memory. Protocol Buffers were felt important because they are tightly coupled to the implementation of Hadoop [14]. The system to be delivered and demonstrated at the end of Phase III would consist of a multi-user, networked, GUI-based application capable of assisting human analysts with redaction of several complex file formats by showing the analyst side-by-side views of the selected document and the same document reformatted with suggested redactions, justified by reference to one or more SCGs, selectable by the user. It would handle up to twenty SCGs at a time, a number suggested to the developer by the sponsor as being a typical and useful capacity for the tool.

3.2.2 Actual Implementation. The proposed system architecture, consisting of a single redaction server supporting an indefinite number of analyst workstations, pulling classified documents to be redacted from a file server, has survived unchanged except for the addition of an Extensible Markup Language (XML) based MAG Generic Interface (MGIF) Cross Domain Solution (CDS) interface to the redaction server through which a generic CDS could route difficult cross domain problems to the system for human review. Experiments with the corpus were begun using Perl, that language being well suited to the very large text handling problem of uncompressing the compressed distribution format of the Web 1T 5-gram corpus, to the tree structured problem of reconstructing the data structure in memory, and to the text processing problems of parsing SCGs and documents for redaction. In Phases I and II, the redaction tool would be command-line driven and limited to text only input and output, so the command-line Perl language was a good match. Concurrently, other experimentation was performed using the Common Lisp (CL) language, as it was anticipated before the decision was made to use Hadoop that CL or Scheme would be the implementation language.

The first indications occurred early in Phase II that normalization would yield a very fast data structure. A phrase matching function was written in Perl that searched all the way down into 5-

word phrases in the corpus. In the second step of the Perl prototype engine, a tree is generated. Child nodes are ordered by prefixes in a case-insensitive relation and abnormal words are flagged. Parsing a document in MAG naturally lends itself to producing a corpus, so minor corpora were often generated for routine tests. In the annotated tree structure, codes indicate whereby if an SCG term is found, the word, sentence, paragraph or document should be redacted. Type codes are used to distinguish between word hits and proximity matches, including siblings. Once every word, sentence and paragraph have been characterized and indexed, the tool walks the word list giving good scores to “self” and close variants of it. Distance metrics are computed in a separate conceptual pass. The final recommendations are presented to analysts in paragraph sized units, for context. A “clean word” list was added to the existing “dirty word” list after experience showed it was needed.

Comparison of several word variant matching functions including Soundex [15][16] and variations on Soundex that propagate information about the last letters of a word [17][18][19][20] into the index were investigated for matching of verb conjugates and irregular forms. Improvements to the matching algorithm came about by incorporation of the thesaurus as previously mentioned, settling on a variation of the Soundex algorithm for matching slightly misspelled words and proper names and words that are not proper names.

Target hardware was acquired, the specifications of which were determined by modeling of the expected size of the working set in real memory, expected storage requirements of leaf nodes in virtual memory, disk storage requirements for holding multiple copies of the corpus during software development, and processor configuration suitable for evaluating the performance of Hadoop on a High Performance Cluster (HPC) machine. The hardware chosen was a Sun Fire X4270 rack mount server from Oracle, Inc. with 32 GB of Random Access Memory (RAM), two quad-core 2.93 gigahertz (GHz) Central Processing Unit (CPU) modules, and nine 146 GB 2.5 inch hard disks configured as a Redundant Array of Independent Disks (RAID)-Z2 under the ZFS file system in Oracle Solaris 10. The underlying RAID-Z configuration was chosen for primarily for read-performance reasons after measurements of disk throughput on early experiments showed a pronounced bottleneck at the CPU–disk interface. The RAID-Z2 variant of RAID-Z was chosen for reliability reasons; it maintains additional copies of the RAID parity information across disks, and can therefore tolerate two disk failures before data loss occurs, an important advantage over RAID 5, which can only tolerate a single disk failure before information is lost. The unit is a 2U chassis with sixteen hard disk bays in addition to the DVD optical drive slot, and was selected for use as the target platform for the redaction server, to provide software development storage space, and as a Configuration Management (CM) repository for the proof of concept demonstration in Phase II and prototype system in Phase III.

3.2.3 Complex File Formats. It was recognized early on that classified information exists in a wide variety of file formats ranging from plain text to complex formatted documents such as Microsoft Office (comprising the very differently internally structured Word, PowerPoint, and Excel at least; possibly including Schedule, Access, and Outlook as well; and numerous incompatible versions of those file formats besides), Portable Document Format (PDF) files, multimedia, and Hypertext Markup Language (HTML) and Flash web pages. It would be necessary to parse and reformat the complex file formats on input and output, to present a usable user interface to analysts, and to maintain the data structures in a persistent and adaptable manner in the face of expected additions, changes, and updates to security classification guidance.

In anticipation of the need, experiments were performed with a method for parsing and formatting PDF files. The method takes advantage of the ancestry of PDF in PostScript language Level 2. The open source Ghostscript tool was used to convert PDF to readable PostScript which could then be parsed to extract plain text, metadata about font metrics, and measurements of page size, paragraph dimensions, headers and footers. Results of experiments indicated that decompiled PostScript—a Forth variant, stack oriented language using Reverse Polish Notation (RPN)—is quite readable and easily parsed for the needed information. PDF files generated by different applications (OpenOffice.org, Cute PDF Writer, Acrobat Writer, Microsoft Office, Ghostscript and Inkscape) were compared for variations. None were found that could not be handled. In some cases (Microsoft software) the printer driver produces very short phrases in PostScript “(text)” blocks which “was troubling but characteristic of the way Microsoft Word works internally (multi-level linked list of delimited words and paragraphs); however, the result is found to be possible to reassemble without too much difficulty.

What is lost in a naïve PDF to text conversion are primarily line breaks and some paragraph formatting. It was recommended to extract dimensional metrics from observed text block boundaries and font metrics, followed by re-flowing redacted text within those constraints for best visual effect. It is probably important not to change page numbering or cross-references intra-document, so the recommendation is to re-flow text with placeholder blocks computed from the content of the redacted text. Line breaks within a paragraph are expected not to be preserved after redaction. An improved but still conceptual algorithm for text extraction from PDF avoids the interleaved-margin problem by keeping a model of the page in memory. Extension of the same technique to de-flow columns was found to be practicable and the only remaining difficulty would be handling of “pulled” paragraphs, sidebars and similar formatted quotations. Research from Optical Character Recognition (OCR) page understanding techniques is expected to return a solution. Feature extraction of probable margin location and sizes is not dissimilar to image processing for machine vision; a partitioning algorithm that greys text to grayscale blocks and then combines contiguous rectangles, combined with a river detection filter that starts from the outside and moves inward to locate margins, is currently being investigated on the side. The most promising approach is believed to be a weighted score combining PostScript metrics with the image analysis results followed by cropping and extracting text from constrained regions with good homogeneity—even “color” in printers’ argot [21][22][23]. The next planned activity was to write a PostScript Level 2 page structure extraction parser in PostScript that would take a multi-page document and produce a description, per page, including page dimensions, paragraph dimensions and locations, a list of type fonts used and font metrics for them for use in dimensional calculations, word length metrics, page headers and footers, page numbers, and classification markings, if present). Following that, the next task would have been to write a multi-page document page re-formatter capable of taking a compatible page description structure and a list of text, then reassemble a redacted document containing the processed text but in as similar a physical format to the original as possible. That work was not completed.

3.2.4 Model Security Classification Guide. It was recognized early on that the wide variation in size, quality, style, amount of detail, and formatting of SCGs would present a difficult problem for parsing these into a form usable for automated processing; accordingly, two independent approaches were planned. First, it was assumed that in the operational environment, actively maintained SCGs would be updated from time to time by their responsible security

organizations, necessitating revision of the SCG tuples used by the tool. It was felt that the MAG parser/formatter, a reusable software component of the Radiant Mercury (RM) CDS, would be suited to the task of semi-automatically determining updates from revised copies of particular SCGs. Other SCGs, depending on writing style, size, internal consistency, frequency of expected updates and other qualities, might be best left to be processed by hand. The decision whether or not to write MAG specifications for a particular SCG would have to be made on a case by case basis. The second parallel approach to solving the SCG problem was to define a Model SCG (MSCG) that epitomized the qualities of completeness, consistency, freedom from ambiguity, and organization to be found in the best existing SCGs, or better. It was felt that an MSCG published to the community would be a useful contribution in itself.

The Minuteman and Peacekeeper SCG was acquired early on; anecdotal information from more than one source agreed that the Minuteman SCG was excellently written and should serve as one benchmark for the MSCG; in particular the way the Minuteman SCG began with a general statement of the philosophy or methodology of what information is to be protected, followed by an increasing level of detail in the implementation [24]. The advantage of this approach is that missing or ambiguous decision points can be inferred from the highest level statement if necessary.

3.2.5 Acknowledgements. The people in Table 2 are hereby acknowledged for their technical contributions to the success of the project.

Table 2. Acknowledgements

Name	Role	Contribution
Jeff Dutoit	Project Manager	Project management, planning, and finance.
Joe Loughry	Principal Investigator	Original concept, research, and writing.
Tom Marso	Software Engineer	Design and coding of prototype engine; collocation correlation concept and implementation.
David Neal	Software Engineer	Graphical user interface; CDS interface.

3.2.6 Metrics. It was planned from the outset that metrics would be collected both for the primary purpose of performance tuning and error rate estimation and for the secondary purpose of software engineering process improvement in cooperation with Quality Assurance (QA). Occurrence counts of phrases in the corpus allow for continuous performance tuning and error rate measurement. The benefit of keeping historical records of counter values for trend analysis is clear, but it occurred that indirect metrics derived from some of these counters would be valuable to the QA organization also—especially when collected from the earliest phases of a new project—for their own software process improvement research. It was intended to use a method for estimating the expected rate of errors of Type 1 and Type 2 in the redaction function based on Probabilistic Risk Assessment (PRA). Data are available that can be used in the future for monitoring and adjusting multiple threshold values controlling the overlapping area between probability density functions describing the likelihood of occurrence of false detection or non-detection of classified data according to a particular SCG. The intent is to use this method both to predict performance and to validate that the tool works. At the present time, metrics are collected and archived but have not been analyzed yet. See Figure 1.

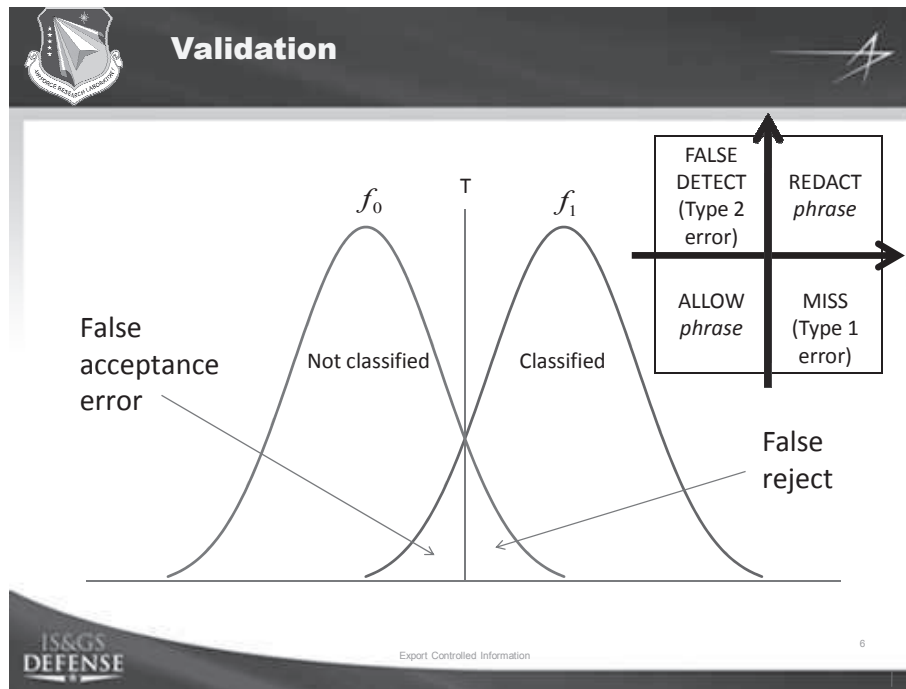


Figure 1. Error Rate Minimization at Threshold T Based on Probability Density Functions

Use of the PRA technique for estimating error rates and tuning the T parameter for optimum performance depends on empirical data for the shape of the observed error rate distributions that do not exist yet and will not until the system has been running in production for a while. All of the proposed forward quality improvement mechanisms including this one were never expected to yield measurable results in the development phase. Enough additional detail on the proposed PDF methodology is provided in the appendix, however, to allow statistics experts to fairly judge the methodology.

3.2.7 Statistical Analysis of Testing Results. False positives, also called Type 2 errors, are the real challenge to detect in testing; miss errors, called Type 1 errors, were found to be less of a problem. *Precision/recall* is another name for the same concept. *Precision* can be thought of to mean avoidance of Type 1 errors, meaning the tool did not overlook any classified information that ought to have been redacted. *Recall* is a measure of avoiding Type 2 error, so that the tool does not flood the user with too many irrelevant false positives. Where this tool shines is under the metric of precision; under the current state of development, it is recall that performs more poorly and where it is hoped to improve in the future. Quantitative measurements of precision and recall are not available because the Principal Investigator (PI) failed to design in the kind of test designed to yield statistical significance. *Post hoc* analysis of the test report shows that the number of trials run, a total of 159 tests using 20 test files, is insufficient for significance at a 0.95 confidence level, assuming an effect size of 11.5 bits/word in English and the file sizes used in the test, about 1500 words each. The test files used in acceptance testing unfortunately were not designed for measuring statistical significance; they were designed for exercising alternative code paths. The test files contain relatively little entropy; consequently, it would take a very

large number of them to approach a 0.95 confidence level, perhaps as many as twenty thousand files.

Qualitatively, precision is the strong point of the present method. The collocation correlation function detects, as described above, non-exact matches that would be missed by a regular expression matcher. The drawback is poorer results in the recall measure, because with a larger vocabulary the tool is likely to encounter more false positives. This is the reason for wanting to integrate the cosine correlator: it would let the tool rapidly match against several—potentially a large number of—custom corpora as soon as a document is presented for redaction, immediately giving a score to each document of how similar it appears to, for example, a collection of mixed unclassified and confidential documents, versus how similar it looks to a collection of SECRET documents. That score would affect the weighting of matches found in the normalized and annotated corpus, thereby improving the recall score.

3.3 Capabilities Achieved

The design of the system offered some obvious opportunities for workflow management throughout the redaction process, for incremental development of new capabilities, for gathering useful metrics on performance of the system and of human analysts for continuous improvement of throughput, accuracy, and security, and some non-obvious opportunities for using a system that had been running a while to generate new and unique test data by inverting the redaction function and figuring the relation backward. Some of these goals were accomplished.

4.0 RESULTS AND DISCUSSION

What was originally envisioned was an automatic tool for assisting human analysts with redaction of complex formatted documents in multiple file formats based on automatically applying rules from multiple security classification guides with extremely high throughput by means of a unique linguistic data structure. What was achieved was an automatic tool effective in the text domain with multiple security classification guides on difficult natural language problems at higher speeds than anticipated by using the observed structure of common usage normalized from the Web 1T 5-gram corpus of English and still applicable to the original technical approach.

4.1 Demonstration and Delivery

An early design assumption was that that analysts would be on a low-cost Personal Computer (PC) workstation, using a GUI implemented in Java. The user interface would be served to clients from a redaction server that retrieved products for redaction from a file server on the high side network. The straightforward networked architecture simplifies system integration and security boundaries. When the tool is started (see Figure 2), the analyst pulls down the File menu, chooses Open, and sees a standard file open dialog. After displaying both the original and redacted versions of the document side by side, the tool begins offering suggestions. The second time—a configurable parameter for the system administrator—that a particular redaction suggestion is presented to the analyst, the tool asks whether the same response to that type of suggestion in the future should be made automatic.

The analyst can walk through suggested redactions forward and backwards easily, in a way similar to how IBM Rational ClearCASE presents diffs of file versions in its user interface. The standard File, Edit, and Help menus appear, with an additional SCG menu following the Edit menu on the menu bar that contains sub-items for adding, deleting, or loading a group of SCGs. An important use case is to allow the analyst to select a favorite group of SCGs to all be loaded and active simultaneously. The Edit menu should have an Undo feature, ideally with unlimited undo/redo forward and backward, but this feature is not implemented yet. Under the Help menu should be online help, reference, and tutorial and About entries, but these are not implemented yet.

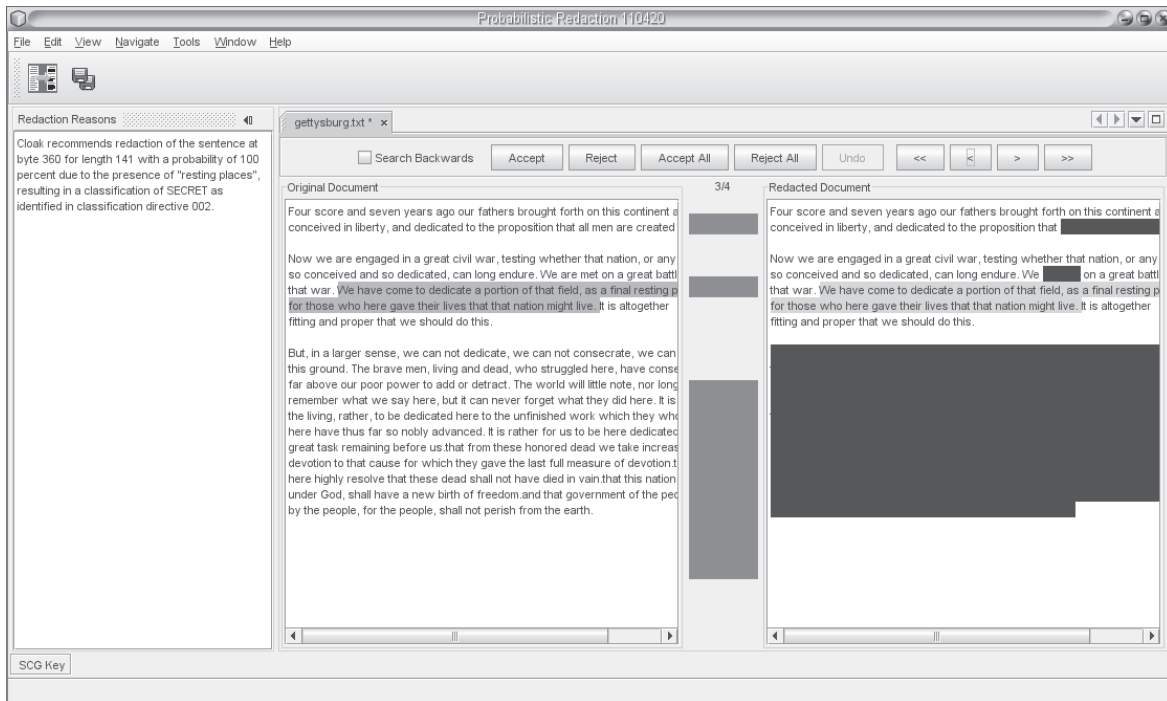


Figure 2. Main Screen of Graphical User Interface

Scrolling through the side-by-side windows showing original and redacted documents is synchronized. When the analyst is satisfied with the contents and appearance of the redacted document, saving the redacted file creates a new file containing the redacted document contents. At the present time, only plain text files are supported. Extensive research was done early in the project on known techniques for securely redacting different types of documents and avoiding the problem of spills; to the present implementation only a few of these methods apply [25][26][27].

4.2 Performance

It was reported in December 2009 that the majority of time required is taken by reading the Web 1T 5-gram corpus files (as expected) but that indexing through the resulting tree was very fast. It was a significant result of Phase II that the Perl prototype gave good performance at phrase matching with the 1-gram data set even before additional levels of tree analysis were needed.

Array size limitations were never found to be a limitation, but limits on the number of open file limits were expected to become a constraint eventually. A 64-bit Perl language runtime is expected to eliminate that problem [28]. Early on, it was expected that the probabilistic matching algorithm would be memory-bound, but it developed that while the Perl prototype used much more disk space than originally envisioned, performance was actually CPU-bound. The normalizer initially was set up to elide excessive punctuation—defined as all special characters—from candidate words, but it was also discovered in operation of the prototype that this was not as important a precaution as was first thought. While it might prove useful on noisy input such as might be obtained from an OCR input scanner, the performance of the classifier on dirty input was found to be almost unaffected.

A practical problem discovered during proof-of-concept was in the handling of root words appearing in the Dirty Word list. English contains many variant word forms—irregular verb conjugations being the least part of it—that can fool a simple Universal Character Set (UCS) Transformation Format—8-bit (UTF-8) compatible bitwise matching algorithm. The solution came from adapting known techniques used in computer spelling checkers. Mitton [29] describes the operation of modern semantic matching algorithms, which work interestingly differently from the original spelling checker methods invented by McIlroy and developed by Kernighan and Plaughter [30][31]. A variant of the original method augmented by the prefix and suffix-handling methods, but not based on phonetic structure, as described in Mitton (*op. cit.*) was expected to prove helpful in two instances: firstly, the problem of misspelled words in the corpus, source material, and SCGs, and secondly, for detecting variant word forms in grammatical English that can be expected to occur regardless of writing style.

The speed of the correlator was tested and reported to take approximately a day to ingest the full corpus from a cold start and be prepared to run; subsequently it takes a few minutes to redact a reasonably sized text document of a few hundred kilobytes in the first run, with subsequent documents processing faster after in-memory data structures have been established. The time to annotate the corpus, at present about twenty-four hours, doubled when the collocation correlation capability was added. The annotation step needs to be re-run, however, only when changes are made to an SCG. The technical approach chosen, investing a great deal of time in preprocessing, minimized submission-to-response latency even for large documents and large SCGs.

4.2.1 Improvements to Performance. The current Perl runtime hogs only one CPU core; as a future performance enhancement it would be a matter of a few hours of programming time to distribute the work across multiple processors. As previously mentioned, performance of the matching function was found to be acceptable when only the 1-gram database was loaded. What multigrams bring to the problem is information about which words are used in combination. Colocation correlation was the most exciting innovation discovered during Phase II. If the tool spots word or concept *C*, the Bayesian probability of encountering *A* or *B* thereafter is raised. (The colocation function could be made limited in the analogy of geographic scope if needed.) This new capability opened the possibility of important new functionality. It is believed that colocation correlation could be used to improve the quality of existing SCGs. This is separate from any concomitant improvement in performance of the matching function. In operation, conceptually as the tool encountered more and more target documents in a semantic domain, it would begin to notice colocation correlation patterns. As the tool operated, it would emit a stream of suggested improvements to the loaded SCGs based on its experience using them with the target population. Automatic detection and resolution of conflicting SCG guidance was

already a research goal, and recommendations for a Model SCG structure was a planned outcome, but the opportunity for continuous improvement of existing SCGs was never expected.

On the inverse function problem, some progress was made. It was expected early on that use of the map-reduce algorithm would be essential to solving the inverse matching function problem. The inverse matching problem is interesting because it would enable forward quality control on SCGs; instead of being limited to a set of pre-defined test cases, the system would have the capability of generating candidate test cases *ex nihilo*. The design of the software allows for a number of interesting opportunities for continuous improvement, most notably SCG suggestions and forward quality control. In SCG suggestions, the tool in operation emits a stream of suggested improvements to the loaded SCGs based on collocation of terms in the target population. In forward quality control, the system can generate its own test cases; once the tree has been annotated with all forms, all that would be necessary is to walk a re-expanded version of the original corpus and collect phrases with high scores. While this is a time-consuming linear operation and storage-bound, it is not actually a common use case and could be performed off-line with a highly parallel cluster if desired in Phase IV.

Surprisingly, the current technical approach was found to be sufficiently powerful to implement the capability without the need of re-architecting for Hadoop or another map-reduce implementation at the time [32][33][34]. Once the tree was annotated with all ancillary forms, all that turned out to be necessary was to walk the entire corpus and collect things with high scores. While map-reduce would make the operation of mapping over the entire corpus a fast operation, it is not thought to be a common use case and would not be necessary to be done often. Linear processing likely suffices. Use of map-reduce was therefore held in reserve for when performance again showed itself to be a bottleneck.

Further to improving performance, it was proposed that a software component from a previous redaction project, the cosine correlator, be applied for making a rapid high-water-mark classification determination on input material for the purpose of workflow prioritization in analysts' work-off queues. Trained on a representative sample of classified and unclassified documents, the cosine correlator would emit a pair of scores to serve as the input to a continuous-valued Bayesian decision as to whether a particular document appears to be classified above or below a given threshold [35]. While mute on the question of where the classified portions are in the document, thereby necessitating both algorithms, the rapid correlator is expected to provide a greater level of assurance of security, improved throughput on high-priority products, and a degree of defense in depth. In addition to determining the likelihood of classified content, the cosine correlator, trained on different sets of sample input, may provide a useful method for rapidly identifying documents containing non-classified or orthogonally relevant concepts, in the manner of a topic identifier [36][37][38].

4.2.2 Alternate Network Representation. In the interest of completeness, a further result of research and experimentation is disclosed, even though it did not yield results by the end of the project. A suitable data structure for employing Hadoop on, that would have solved the reverse mapping problem and led to automatic generation of test data if it had worked, was the so-called "alternate network representation." The advantages of having a parallel technical approach—not for the purpose of replacing the prototype engine but rather to act as a second opinion on problematic cases—are thought to be primarily in the area of synonym detection because the network representation, based on a Hadoop implementation, follows the original probabilistic redaction idea back to inverting the corpus and rapidly searching the tree of empirically



Figure 3. Alternate network representation

occurring phrases in the forward direction, complementing the search strategy of the current implementation which works upward from the SCG guidance to words, sentences and paragraph units in the text [39][Pearl, *op. cit.*]. The alternative network representation is illustrated in **Error! Reference source not found.** The data structure represents a break from the tree structured data used heretofore. It is rather a network of 1-grams and variations on those 1-grams in a digraph. The new term “*ur*-phrase” was suggested to denote the stripped-down essence of a sentence: for example, the *ur*-phrase of “digging a deep hole on Wednesday” would be *dig deep hole wednesday* [sic]. This is related to the concept of min-hashes in which related but not necessarily identical objects, for example candidate fingerprint matches in a fingerprint database, reduce to the same min-hash. Unlike a cryptographic hash having the cascade property, min-hashes are designed to maximize the likelihood of hash collisions for similar inputs.

It was thought before that conflicting guidance in SCGs was the real problem, but it is now clear that the real key to the issue is overlapping recommendations in the SCG, not conflicting ones. If the alternate network representation works, then the best advantage of using Hadoop will be in the pre-processing. The network structure will multiply the size of the pre-processing problem exponentially, but the result would be much more reliable redaction. It was pointed out by a reviewer that Hadoop can also assist on the front end by rapidly searching the pre-processed network for matches; it can do the search in an instant rather than necessitating a linear search.

The SCG problem requires a supply of actual security classification guides. Classification Directive (CD) tuples appear in the SCG in one of four forms of a pattern similar to the following quotation: “if the phrase appears, within n words, then the entire semantic unit—phrase, sentence, paragraph, or document—is classified. If all of the words in the following set {} appear, then...” and so on. In a tree of phrases based on the sub-root “digging holes”, for example, based on the example SCD, the phrase “not digging holes” should probably not be classified, but “digging holes yesterday” or “digging big holes” should be classified. In summary, it is believed that the alternate network representation can be used in addition to the existing method—not replacing it—to improve the handling of longer phrases. The existing method works quite well at present on shorter phrases; it integrates well with a Hadoop solution of the alternate network representation in which the two work together synergistically. The preprocessing step currently takes about twenty-four hours with collocation correlation turned on; preprocessing in the alternate network representation might take thousands of hours; the clustered scalability of Hadoop would be advantageous here. Hadoop offers at least three advantages: (1) scalability, (2) handling of extremely large data sets during preprocessing, and (3) search speed in operation.

Mason *et al.* presented research at CCS’09 reminiscent somewhat of the inverse problem [40]. The authors sought to find plausible sequences of natural language, using a restricted alphabet of symbols whose binary encoding coincidentally matched valid sequences of Intel X86 machine language instructions, to form candidate English phrases evaluated by a fitness function based on a Markov chain generator trained on email messages. Being limited by the extended X86 instruction set as to what they could implement, the authors nevertheless managed to get a decoder running for the encrypted data block where most of their payload resided. As a proof of concept it was valid, but more importantly it suggests that the alternate network approach will be successful.

4.2.3 Test and Evaluation of Prototype. The GUI is written in the Java language and runs on any low-cost commodity workstations so long as they have a current Java Virtual Machine (JVM). The test director noted in the test report [41] that “the new GUI makes the tool very easy to use. With very minimal basic knowledge of what the tool does a user could very easily figure out how to use the tool through the GUI.” In addition, the Test Director noted the following some issues regarding performance that are significant to evaluation: “Scalability: The test was only run on the one type of hardware (as defined in the test report)...the test team would venture to claim that this tool would not be affected greatly by increasing CPU and RAM specs. The system appears to be hitting the hard disk enough that increasing the CPU and RAM would probably be bottlenecked by the hard disk. Increasing the hard disk speed and the disk interface speed would increase performance,” [*ibid.*].

The CLOAK tool—the name given to the system for publicity purposes—was demonstrated in the first week of August 2011 at Chicago for the 5th Unified Cross Domain Management

Office (UCDMO) Conference. For the purpose of demonstration, all of the software was resident on an Intel architecture laptop in four virtual machines with virtual Local Area Network (LAN) connections between them, simulating the existence of an NFS file server on a simulated high side network, the CLOAK redaction server, Radiant Mercury (RM) Cross Domain Solution (CDS), and a simulated low-side network. Note that the CLOAK server acts as a kind of CDS. The laptop was plugged into a large screen display on the conference floor. The setup allowed demonstration of all functionality including the CDS interface without the necessity of either transporting the relatively fragile rack-mounted server to the conference or provisioning a secure and reliable network connection from the conference floor back to the plant. A product data sheet was prepared with the assistance of a graphic designer and approved by the Air Force Research Laboratory (AFRL) for limited distribution at the conference provided it bore the appropriate Distribution Statement C, which was provided. The data sheet highlighted the user interface and CDS integration for automatically routing products to CLOAK for human interaction. Prior to the conference, a dry run of the demonstration was conducted at Lockheed Martin IS&GS-Defense. The most useful result of the dry run, besides the assurance it provided that the demonstration at the upcoming conference would go smoothly, was observation of the reactions of observers to the operation of the tool; an audience previously unacquainted with the concept of probabilistic redaction was most favorably impressed with the collocation correlator's discovery, through automated analysis of SCG tuples, that the abbreviation "Mbps" for megabits per second, while never specifically appearing in the SCG as a classified term, was sufficiently correlated with the classified concept "bandwidth"—which did appear in the SCG—to make the phrase "1.5 Mbps" a candidate for redaction in one of the test documents. This was a completely unexpected result.

5.0 CONCLUSIONS

The project was successful. Some, but not all, of the original technical objectives were achieved and unexpected discoveries were made. A working prototype was delivered and demonstrated on time and under budget. It was found necessary to restrict the focus of work early on to plain text, leaving the parsing and formatting of complex file formats like PDF to be performed as a service by existing Commercial Off-The-Shelf (COTS) software, directing attention instead to validation of the probabilistic redaction concept. The original design for an automated interactive redaction assistant was sound and in practice works well, either stand-alone or as a side-car to a CDS. The biggest surprise of the project was finding from the earliest prototypes in the Perl language, using associative arrays, that performance was acceptable even without the planned implementation of HPC map-reduce. Independently, collocation correlation proved to be a significant advance in and of itself, delivering on the promise of forward quality improvement of SCGs even before the MSCG could be developed.

The way ahead is clear: to further develop this successful prototype into an effective tool, it is imperative to explore immediately the performance boost and capability increases offered by a Hadoop implementation of the parallel technical approach of **Error! Reference source not found.**, the improved assurance and precision/recall performance offered by the cosine correlator, and assessing the operational effectiveness of an interactive redaction assistant on PDF, Microsoft Office, and complex format web content. Further improvements in accuracy and throughput of both interactive and automatic redaction are expected from the application of

measured analyst efficiency metrics to the routing of workflow and automatic forward quality improvement of security classification guides.

6.0 REFERENCES

- [1] Linguistics Data Consortium. “Web 1T 5-gram Corpus”. LDC catalog number LDC2006T13 (2007).
- [2] Sale, T. *The Lorenz Cipher and how Bletchley Park broke it*. Unpublished manuscript.
- [3] Schneier, B. “NSA Style Manual”. *Schneier on Security* blog. June 23–July 1, 2011. http://www.schneier.com/blog/archives/2011/06/nsa_style_manua.html.
- [4] Zawinski, J.W. “Comments on blog posting ‘[REDACTED]’ ”. 10–11 May 2004. <http://www.jwz.org/blog/2004/05/redacted/>.
- [5] Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco: Morgan Kaufmann, revised 2nd printing, 1997.
- [6] Sanchez, J. “The Redactor’s Dilemma”. December 8th, 2009. <http://www.juliansanchez.com/2009/12/08/the-redactors-dilemma/>.
- [7] Naccache, D. and Whelan, C. “9/11: Who alerted the CIA? (And Other Secret Secrets)”. Rump session paper, EUROCRYPT 2004.
- [8] Butler, D. “Censored words unmasked”. *Nature News*, 13 May 2004. <http://www.nature.com/news/2004/040513/full/news040510-8.html>.
- [9] Zawinski, J.W. “Comments on blog posting ‘Redacting the Redactors’ “. 25–27 May 2011. <http://www.jwz.org/blog/2011/05/redacting-the-redactors/>.
- [10] Lee, T.B. “Studying the Frequency of Redaction Failures in PACER. *Freedom to Tinker* blog, May 25th, 2011. <http://freedom-to-tinker.com/blog/tblee/studying-frequency-redaction-redaction-failures-pacer>.
- [11] Lockheed Martin Information Systems and Global Services (LM IS&GS). “Cognitively Assisted Cross Domain Information Sharing: Research and Development (R&D) Quarterly Progress Report for January through March 2010”. PR-RPT-11694-005, dated 08 April 2010.
- [12] J. Dean and S. Ghemawat. “MapReduce: A Flexible Data Processing Tool”. *Comm. ACM* **53**(1), pp. 72–77 (2010).
- [13] T. White. *Hadoop: The Definitive Guide*. Sebastopol, California: O’Reilly & Associates, 2009.
- [14] K. Weil. “Protocol Buffers and Hadoop at Twitter”. *Bay Area Hadoop Meetup*. February 23, 2010. URI: <http://www.slideshare.net/kevinweil/protocol-buffers-and-hadoop-at-twitter>.
- [15] Davidson, L. “Retrieval of misspelled names in an airline’s passenger record system”. *Comm. ACM* **5**(3), pp. 169–171, March 1962.
- [16] Knuth, D. E. *The Art of Computer Programming, Volume 3: Sorting and Searching*. Second edition. Upper Saddle River, New Jersey: Addison–Wesley Publishing Company, 1973.
- [17] Larsen, K. “The Science of Word Recognition”. *Proceedings of ATypI 2003*, Vancouver, Canada, September 25–28, 2004.
- [18] National Archives and Records Administration. *Federal Population Censuses 1790–1890*. Washington, D.C.: National Archives, 1971.
- [19] R. C. Russell. *Index*. U. S. Patent no. 1,261,167, issued April 2, 1918.
- [20] ———. *Index*. U.S. Patent no. 1,435,663, issued November 14, 1922.

- [21] Adobe Systems Incorporated. *PostScript Language Reference*. Third edition. Boston: Addison-Wesley, 1999.
- [22] ———. *PostScript Language Tutorial and Cookbook*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1986.
- [23] ———. *PostScript Language Program Design*. Boston: Addison-Wesley, 1988.
- [24] Department of the Air Force, Headquarters Ogden Air Logistics Center, Hill Air Force Base, Utah. *Minuteman and Peacekeeper Weapon Systems*. OO-ALC/LMDM, 75 SFS/SFAI (undated, no classification marking).
- [25] Cluley, G. “How NOT to redact a PDF—Nuclear submarine secrets spilled. 18th Apr 2011. <http://nakedsecurity.sophos.com/2011/04/18/how-not-to-redact-a-pdf-nuclear-submarine-secrets-spilled/> .
- [26] ———. “How NOT to redact a PDF—Military radar secrets spilled”. 9th Oct 2011. <http://nakedsecurity.sophos.com/2011/10/09/how-redact-pdf-air-defence-radar-secrets-spilled/> .
- [27] Cross Domain Solutions Product Division, National Security Agency. *Inspection and Sanitization Guidance for Portable Document Format*, Version 1.0, dated 2nd May 2011.
- [28] Pott, T. “Reg Server Management: But it said so in the manual: NTFS limits provoke scoffs, RAM purchases”. *The Register*, 28th September 2010. URI: http://www.theregister.co.uk/2010/09/28/sysadmin_ntsf_file_limits/ .
- [29] Mitton, R. “Spellchecking by computer”. *Journal of the Simplified Spelling Society* **20**(1), pp. 4–11 (1996).
- [30] Bentley, J. “A Spelling Checker”. *Communications of the ACM* **28**(5), pp. 456–462 (1985).
- [31] Kernighan, B. W. and Plaugher, P.J. *Software Tools*. Boston: Addison–Wesley Professional (1976).
- [32] Ho, R. “Designing algorithms for Map Reduce”. *Pragmatic Programming Techniques* blog. Sunday, August 29, 2010. URI: <http://horicky.blogspot.com/2010/08/designing-algorithms-for-map-reduce.html> .
- [33] Pavlo, A. and Paulson, E. and Rasin, A. and Abadi, D. J. and DeWitt, D. J. and Madden, S. and Stonebraker, M. “A Comparison of Approaches to Large-Scale Data Analysis”. *SIGMOD ’09*, Providence, Rhode Island, June 29–July 2, 2009.
- [34] Stonebraker, M. and Abadi, D. and DeWitt, D. J. and Madden, S. and Paulson, E. and Pavlo, A. and Rasin, A. “MapReduce and Parallel DBMSs: Friends or Foes?” *Comm. ACM* **53**(1), pp. 64–71, January 2010.
- [35] Jones, M. “Google Ngram database tracks popularity of 500 billion words”. *Neowin.net*. URI: <http://www.neowin.net/news/google-ngram-database-tracks-popularity-of-500-billion-words> .
- [36] Harrison, C. “Web trigrams: visualizing Google’s tri-gram data”. URI: <http://www.chrisharrison.net/projects/trigramviz/index.html> .
- [37] Rajarman, A. et al. *Mining of massive datasets*. Stanford University, 2010.
- [38] Bernstein, A. “Processing Large Graphs”. University of Zurich, 2011.
- [39] Bourne, C. P. and Ford, D. F. “A Study of Methods for Systematically Abbreviating English Words and Names”. *J. ACM* **8**(4), pp. 538–552, October 1961.
- [40] J. Mason, S. Small, F. Monrose, and G. MacManus. “English Shellcode”. In *Proceedings of CCS’09*, Chicago, IL, November 9–13, 2009.

[41] Lockheed Martin Information Systems and Global Solutions (LM IS&GS).
 “Cognitively Assisted Cross Domain Information Sharing: CLOAK Probabilistic
 Redaction Tool v0.9 Test Report”. PR-RPT-12660-000, dated 26 September 2011.
APPENDIX - PUBLICATIONS AND PRESENTATIONS

The poster in Figure A-4 was prepared for the Technical Council at the request of the sponsor.

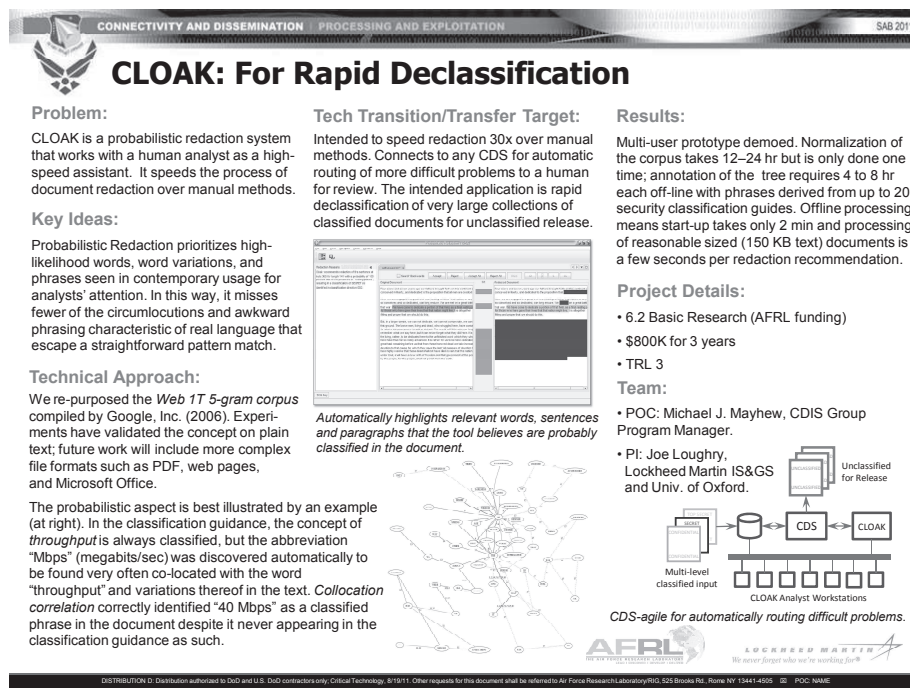



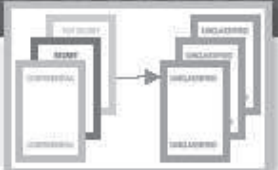
Figure A-4. Technical Council Poster.

The data sheet in Figures A-5 and A-6 was prepared for the UCDMO conference.



CLOAK

FOR RAPID DECLASSIFICATION



CLOAK is a probabilistic redaction system enabling near real-time declassification of text documents with the assurance of human review. High-probability matches—derived from the trillion-word Google research corpus—are weighted to the front of the search results list, thereby wasting no time on low-probability-of-occurrence phrases before high-probability phrases are located in the document in front of your analyst.

CLOAK speeds the process of document redaction over manual methods by automatically highlighting relevant words, sentences and paragraphs in the classified document and matching them to terms and phrases from multiple security classification guides for the analyst. One CLOAK server supports a whole team of analysts using low-cost commodity workstations.

CLOAK can be connected as a sidecar to the cross domain solution to automatically detect and route potentially classified free text in the message stream to security analysts for human review and release.

Multiple security classification guides, even conflicting declassification guidance, are handled with ease. The bulk of processing happens off-line and off the clock, yielding high analyst throughput during operation.

CLOAK learns from experience to spot potentially troublesome aggregation hazards and incomplete security classification guidance. It makes suggestions for improving your security classification guidance, removing inconsistencies or logical impossibilities, and adding clarity.

CLOAK emerged from a high-risk research project undertaken by the Air Force Research Laboratory to find a way to improve the speed and accuracy of human redaction analysts, boosting throughput while reducing the chance of human error by putting another set of eyes on the final product. CLOAK combines a number of technically diverse approaches, cross-checking results for high resistance to accidental spillage or tailored-product attack.

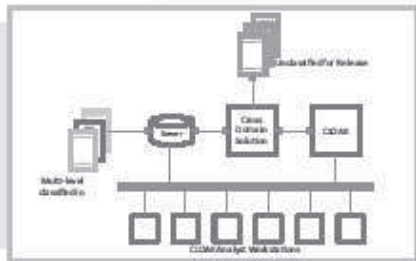
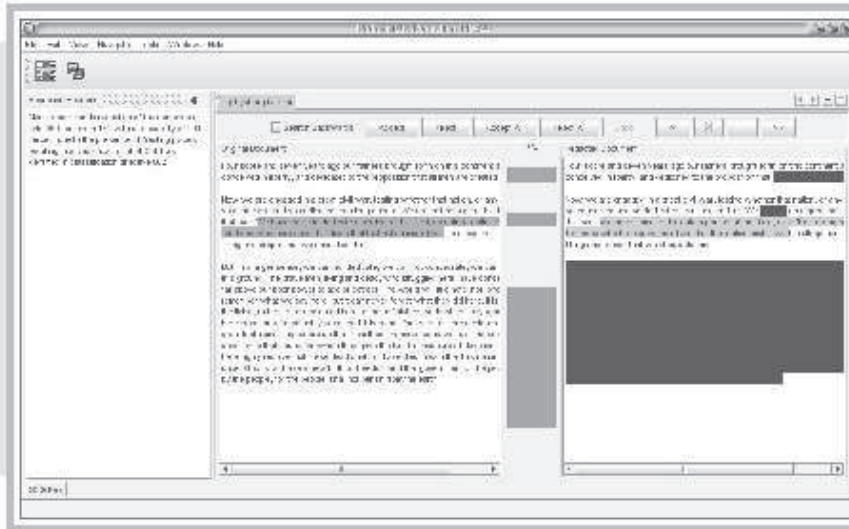


Figure A-5. Data Sheet (front side)

CLOAK

Analysts can do their work from low-cost COTS workstations—the heavy lifting takes place on the CLOAK server.

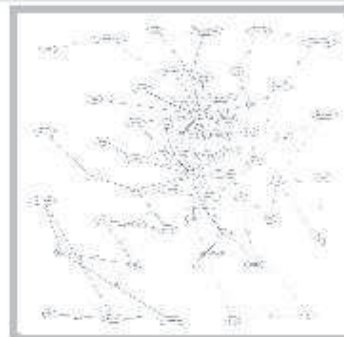
Documents are analyzed and suggested redactions displayed for the analyst to approve. Analysts can choose preferred sets of security classification guides and switch among sets with ease.



Integration with cross domain solutions (CDS) means that a CDS can hand-off difficult problems to CLOAK for human review before release.

Combining the throughput of automatic processing for routine messages with the assurance of reliable human review for free text, CLOAK is intended to speed the process of redaction over currently manual methods.

CLOAK breaks through the logjam by probabilistically looking first at high-likelihood words and phrases from security classification guides, prioritizing them for analysts' attention. Collocation correlation gathers data to spot opportunities where security classification guidance may be improved.



DISTRIBUTION STATEMENT C. Distribution authorized to U.S. Government Agencies and their contractors; Critical Technology; 21 July 2011. Other requests for this document shall be referred to AFRL/RIEBS, 525 Brooks Road, Rome, NY 13441-4505, Attn: Michael J. Mayhew, CDIS Group Program Manager; phone: (315) 330-2898 DSN: 587-2898.



Figure A-6. Data Sheet (back side)

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

AFRL	Air Force Research Laboratory
CD	Classification Directive
CDS	Cross Domain Solution
CL	Common Lisp
CLOAK	The name of the system
CM	Configuration Management
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
DNA	Deoxyribonucleic Acid
DVD	DVD ROM disc
GB	10^9 or 2^{30} bytes
GNU	GNU's Not UNIX
GUI	Graphical User Interface
HPC	High Performance Cluster
HTML	Hypertext Markup Language
I/O	Input/Output
LDC	Linguistics Data Consortium
IS&GS	Information Systems and Global Solutions
JVM	Java Virtual Machine
LAN	Local Area Network
LM	Lockheed Martin
MAG	Message Analysis and Generation
MB	10^6 or 2^{20} bytes
Mbps	Megabits per second
MGIF	MAG Generic Interface
MSCG	Model Security Classification Guide
NFS	Network File System
NSA	National Security Agency
OCR	Optical Character Recognition
OS	Operating System
PC	Personal Computer
PACER	Public Access to Court Electronic Records
PDF	Portable Document Format
PI	Principal Investigator
PR	Probabilistic Redaction
PRA	Probabilistic Risk Assessment
QA	Quality Assurance
R&D	Research and Development
RAID	Redundant Array of Independent Disks
RAID-Z2	RAID 5 with copy-on-write and duplicated parity across disks
RAM	Random Access Memory
RM	RADIANT MERCURY
ROM	Read-Only Memory
RPT	Report

SAN	Storage Area Network
SCG	Security Classification Guide
T	10 ¹²
UCDMO	Unified Cross Domain Management Office
UNIX	UNIX operating system
UTF-8	UCS Transformation Format—8-bit
UCS	Universal Character Set
WWW	World Wide Web
X86	Intel microprocessor instruction set architecture compatible
XML	Extensible Markup Language
ZFS	ZFS File System