D R A F T
(Please do not quote.)
CryptoToolJ: An Extensible Cryptology Tool for Historical Ciphers

Ralph Morelli
Computer Science Department
Trinity College
Hartford, CT 06106
ralph.morelli@trincoll.edu

March 22, 2004

Abstract

This paper describes CryptoToolJ (Cryptology Tool – Java), an extensible Java-based software program for implementing and analyzing historical ciphers. CryptoToolJ is designed to support computer-based research and teaching in historical cryptography. It contains default implementations of a number of historical cipher engines. It also incorporates several easy-to-use cryptanalysis programs, thus providing an integrated software platform for historical cryptologists. What distinguishes CryptoToolJ from similar tools is its generality and extensibility. Because of its open, object-oriented design, new cipher engines and programmer-defined analyzers can be easily incorporated into the program. The software, including source code and documentation, is available for free download. It is hoped that as others utilize this tool, they will contribute to its further development.

1 Introduction

CryptoToolJ is a Java program for the creation and cryptanalysis of historical ciphers. It can be used to encrypt, decrypt, and cryptanalyze messages written with any of its default cipher engines – Caesar, simple substitution, Vigenere, transposition, Playfair, polysubstitution, and others. Moreover, because of its open and extensible design, it can be used by programmers to create new cipher engines and cryptanalyzers, which can then be easily incorporated into the tool. The tool, as well as its underlying cryptography library, documentation, and source code, is available for free download.

1.1 General Features

CryptoToolJ has the following features.

- It is based on *HcryptoJ*, an extensible Java library that supports the creation of historical cipher engines. HcryptoJ is modeled on the *Java Cryptography Extension (JCE)*, Sun Microsystem's Java library for cryptography ([4]). However, HcryptoJ was designed specifically for use with historical ciphers. (For more on HcryptoJ, see [3]).
- It has an easy-to-use, menu-driven, graphical interface. An applet version of the tool is available at [1], which also contains links to the fully-documented downloadable version.
- It can be used to encrypt and decrypt messages written in Greek, Hebrew, Katakana, and other non-English character sets. This feature derives from its use of Java's 16-bit *Unicode* character set.
- It has an extensible interface, that allows *plugin* cipher engines and plugin analyzers to be incorporated into the tool as they become available. As CryptoToolJ is developed further, its website contains a repository of plugin resources.
- It is written in Java, so it will run without recompilation on any hardware platform that supports the freely available Java Runtime Environment, including Windows, Macintosh, Linux, and Unix. Its *object-oriented design* makes it easy to extend the tool.

2 Using CryptoToolJ To Encrypt/Decrypt Messages

Figure 1 shows CryptoToolJ's basic user interface. It presents an easy-to-use platform for encrypting and decrypting messages in a variety cipher systems. The messages themselves can be stored in files or simply typed into the tool's text window. There are two steps required for encrypting or decrypting:

1. Select a cipher engine.
2. Create an appropriate cryptographic key for that cipher engine.

Because the key's format depends on the cipher engine, CryptoToolJ provides a context-dependent dialogue window to guide the user:
Note that in addition to specifying the data that make up a key – e.g., the keyword, or Caesar shift, or an alphabet permutation – the key dialogue also prompts the user to specify both the plaintext and ciphertext alphabets. In CryptoToolJ, an *alphabet* is a set of Unicode character ranges. The Unicode character set is a 16-bit code that incorporates most international character

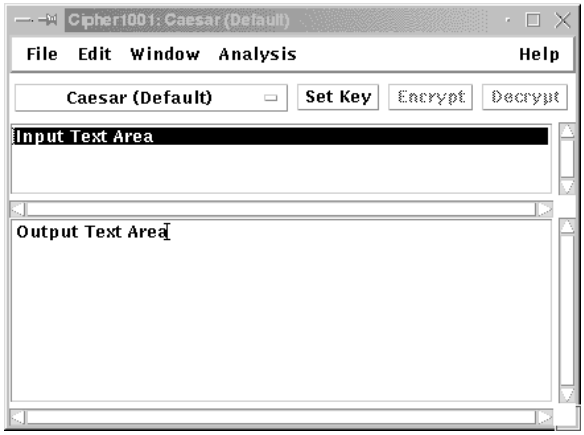

Figure 1: CryptoToolJ's basic interface.

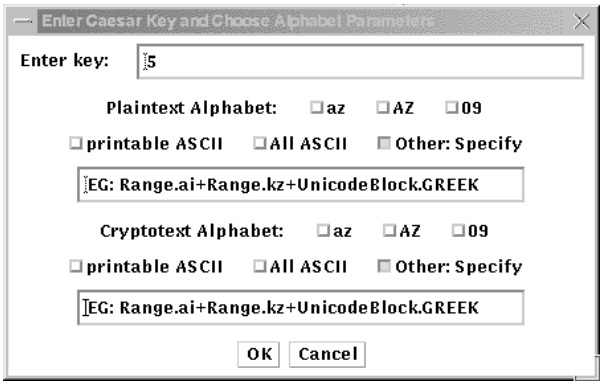

Figure 2: CryptoToolJ's key dialog window.

sets ([2]). The CryptoToolJ user can specify traditional character ranges, such as 'a' to 'z' or can create customized ranges, such as 'a' to 'i' and 'k' to 'z', including ranges in international character sets such as Greek or Katakana or Hebrew.

Once the key has been specified, CryptoToolJ will enable the Encrypt and Decrypt buttons. Text that has been encrypted can be cut and pasted into other documents or saved to a file.

2.1 Default and Plugin Cipher Engines

In its present configuration, CryptoToolJ contains the cipher engines listed in Table 1. The only difference between default and plugin ciphers is in the way

they are loaded into the program. As described below, cipher engines that are placed in a special plugin directory are loaded into the program at runtime.

Cipher Engines		
Affine	Caesar	Substitution
Playfair	Polysubstitution	Vigenere
Transposition	Autoclave (plugin)	NullCipher (plugin)

Table 1: Cipher engines in the current version of CryptoToolJ.

3 Using CryptoToolJ for Cryptanalysis

Table 2 provides a list of cryptanalysis tools that are included in the current version of CryptoToolJ. These tools may be applied to any text that is loaded into the tool's input or output text areas. Most of the tools perform an analysis and report their results in a separate window. For example, the Caesar Analyzer uses the Chi-square test to identify the probable shift of a Caesar-encrypted message and then attempts to decrypt the message using that shift. The Vigenere and substitution analyzers work the same way. However, some of the tools are more interactive. For example, the Histogram tool displays the messages frequency histogram and allows the user to interactively query about the frequency of particular characters.

Cryptanalysis Tools		
Frequency Analyzer	Histogram Tool	Index of Coincidence
Caesar Analyzer	Affine Analyzer	Substitution Analyzer
Vigenere Analyzer	Pattern Word Searcher	Null Analyzer (plugin)

Table 2: Cryptanalysis tools in CryptoToolJ.

CryptoToolJ's analyzer interface has also been designed to be extensible. Thus it is very easy to develop one's own cryptanalysis tools and add them to the program as plugins. To illustrate how this is done, the program comes with a couple of simple examples of plugin analyzers.

4 Extending CryptoToolJ

As pointed out above, CryptoToolJ is designed to grow in functionality as new cipher engines and cryptology resources become available. Of course, the code for these new resources must follow certain design specifications in order to be incorporated into CryptoToolJ. But the interface is quite simple and accessible. This section provides a brief description of the design features that contribute to the CryptoToolJ's extensibility.

4.1 Incorporating a New Cipher Engine

New cipher engines can be incorporated into CryptoToolJ via its plugin interface. When CryptoTooJ is installed, two directories are created in its home directory. The *plugins* directory is the repository for cipher engines and their corresponding keys, and the *providers* directory is the repository for searchable indices that tell the program what cipher engines are available. When the program is started, it searches these directories, incorporating any cipher engines it finds into its menu system.

The *provider* interface used in this design is based on the *Java Cryptography Extension (JCE)* ([4]). This feature enables software developers to *provide* their own implementations of cipher systems. For example, even though CryptoToolJ includes an implementation of a Caesar engine as part of its *default* provider, another developer can provide a different implementation of Caesar cipher under a different provider name. Cipher engines are identified both by their name – e.g., Caesar – and their provider's name – e.g., default.

Cipher engines incorporated into CryptoToolJ through the plugin interface, all have the provider name *plugin*. The current version of CryptoToolJ contains a single plugin provider, *TestProvider*, whose Java code is as follows:

```
package providers;
import hcrypto.provider.*;
public class TestProvider extends Provider {

  public TestProvider(String name) {
    this.name = name;
    put("Autoclave", "plugins.AutoclaveEngine", "plugins.AutoclaveKey");
  }
}
```

As this code shows, this provider creates an implementation of the *Autoclave* cipher. The code associates the name of the cipher with the names of the Java classes that implement it: *plugins.AutoclaveEngine* and *plugins.AutoclaveKey*. When these two classes are added to the *plugins* directory, CryptoToolJ will incorporate the Autoclave plugin cipher into its menu system.

Of course, in order for the program to utilize these classes, they must follow the design specifications laid out in HcryptoJ. In brief, a cipher engine must be implemented as a subclass of *hcrypto.cipher.BlockCipher* and it must have a key that is a subclass of *hcrypto.cipher.HistoricalKey*. (For more on this, see [3].)

4.2 Incorporating a New Analyzer

Incorporating a new analyzer resource into CryptoToolJ works in a similar fashion. The Java classes (code) that implement the analyzer must be placed to the *analyzers* subdirectory. When CryptoToolJ is started it searches this directory and adds analyzer plugins to its Analysis menu. In order for the program to be able to run the analyzer, the analyzer must provide an implementation of

the *hcrypto.analyzer.Analyzer* interface, which contains the necessary details on how the two programs will communicate with each other.

5 Related Work

There are numerous cryptography and cryptanalysis tools available. The following list provides a brief summary:

- Cipher Clerk – a Java applet that contains implementations of a large collection of historical ciphers (http://members.magnet.at/wilhelm.m.plotz/Bin/CipherClerk.html).
- Secret Code Breaker – Javascript implementations of Caesar and other substitution ciphers (http://codebreaker.dids.com).
- Advanced Cryptography Tool – A cryptographic tool that uses strong encryption (http://maga.di.unito.it/security/resources/ACT/act.html).
- Cypher – A cryptologic tool for historical ciphers that was developed as a software engineering project at the University of North Carolina (http://www.cs.unc.edu/ stotts/COMP145/homes/crypt/).
- The Cryptology Matrix – A collection of resources, including CryptAid, for creating and analyzing historical ciphers from New Mexico State University (`http://www.math.nmsu.edu/crypto/public_html/`).
- Cryptlib – A security toolkit that provides strong encryption and authentication services (http://www.cs.auckland.ac.nz/ pgut001/cryptlib/).
- Pretty Good Privacy (PGP) – A security toolkit that provides strong encryption and authentication services (http://web.mit.edu/network/pgp.html).

Most of these are commerical products that provide strong encryption and authentication services. These are not suited at all for historical cryptography.

Among those that are specifically designed for historical ciphers (CryptAid, Cypher), none provides the extensible, integrated platform available in CryptoToolJ. And none appear to be designed to support an ongoing research program in computer-based historical cryptography.

6 Plans for the Future

CryptoToolJ is a very much a work in progress. We have used CryptoToolJ as the basis for a project in a software engineering course ([5]). At Trinity College it will continue to serve as a computational platform for research and teaching in cryptography and cryptology. We are currently working on a number of new cipher engines, including an implementation of the Enigma machine, and a number of new analyzers, including a genetic analysis program.

The current version of CryptoToolJ (v1.3) demonstrates the feasibility of providing an integrated and extensible software platform to support the creation and analysis of historical ciphers. The system's overall design, particularly its plugin interface, will allow it to grow into a more useful cryptology tool as new cipher engines and analyzers are developed.

Future releases of the tool will incorporate new resources. In between major releases, the CryptoToolJ website will serve as a repository for new resources as they become available. It is our hope that amateur cryptographers, researchers and students will use CryptoToolJ and contribute to its extension, thereby making it a generally useful resource in the ongoing study of cryptology.

7 References

1. http://starbase.trincoll.edu/ crypto/hcryptoj/
2. http://www.unicode.org/
3. R. Morelli and R. Walde. HcryptoJ: A Java Toolkit for Creating and Analyzing Historical Ciphers. Unpublished manuscript, 2002.
4. Java Cryptography Extension. URL: http://java.sun.com/products/jce/.
5. Ralph Morelli, Ralph Walde, and Gregg Marcuccio. A Java API for Historical Ciphers. *Proceedings of the Thirty Second SIGCSE Technical Symposium on Computer Science Education*, pp. 307-311, 2001.