CryptoBench User's Guide

Disclaimer

The author makes no warranty or representation that the operation of CryptoBench is or will be error-free, and the author is under no obligation to provide any services, by way of maintenance, update, or otherwise.

CRYPTOBENCH AND ANY DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL THE AUTHOR BE LIABLE FOR DIRECT, INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CryptoBench implements strong cryptography which is illegal in some countries. You are strongly advised to verify the import, export and usage laws and regulations applicable in your country with respect to cryptographic products. The author is not liable for any violations you make.

A good place to start your research is the Electronic Privacy Information Center (EPIC) report Cryptography and Liberty 1999. An International Survey of Encryption Policy.

The use of CryptoBench in the following countries is expressly forbidden: Afghanistan, Angola, Cuba, Iran, Iraq, Libya, North Korea, Montenegro, Rwanda, Serbia, Sudan, and Syria.

Foreword

There are several very good cryptographic libraries available on the internet. Some are offered with little or no licensing restrictions on their use, while others impose regulations that range from non-commercial use only, to the payment of royalty fees. CryptoBench uses a library that falls in the former group: Crypto++.

Developed and maintained by Wei Dai, Crypto++ is one of the best open source cryptographic C++ libraries available. Offering a comprehensive set of algorithms, it is portable across multiple operative systems and compilers. The library's quality, stability and performance are outstanding. Version 5.0.4 was validated by NIST and CSE for FIPS 140-2 level 1 compliance.

While there are other open -or quasi open- source libraries like Peter Gutmann's cryplib, Jack Lloyd's Botan or GNU Libgcrypt, I decided on Crypto++ mainly for two reasons; first, the library is copyrighted as a compilation, but the individual files are not (with very few exceptions) and second, I like its design. Elegant and based on sound, almost academic principles of object oriented software development.

Introduction

Simply put, CryptoBench provides a source of strong cryptographic transformations to help in the cryptanalysis process of common cryptographic schemes.

If the previous sentence doesn't make much sense, you'll probably find the software an odd curiosity. Something worth tinkering with for a few moments and deleting later. However, in doing so, I hope to spark enough interest in the subject of cryptography so that you'll continue experimenting. There are many websites with valuable information, and the search engine of your preference is likely to yield hundreds of thousands (if not millions) of links. Nevertheless a couple of good places to start are PGP's Introduction to Cryptography and the RSA Labs FAQ.

Cryptanalysis seeks to discover a method to translate encrypted information back to its unencrypted state or to "break" the security of cryptographic algorithms and protocols. However as Bruce Schneier puts it in A Self Study Course in Block-Cipher Cryptanalysis:

"…Breaking a cipher doesn't necessarily mean finding a practical way for an eavesdropper to recover the plaintext from just the ciphertext. In academic cryptography, the rules are relaxed considerably. Breaking a cipher simply means finding a weakness in the cipher that can be exploited with a complexity less than brute-force…".

For the most part modern cryptography is the domain of the numerical sciences, notably Discrete Mathematics, Statistics and Information Theory. Anyone seriously interested in this topic will undoubtedly benefit from academic training in those areas but, I think, their mastery isn't a *sine qua non* condition to get involved in this fascinating subject.

Overall, the process of cryptanalysis entails three steps: 1) identification of the cryptographic algorithm, 2) recovery of encryption keys and 3) reconstruction of the plaintext. Most of the available literature centers around the second step but, to my knowledge, there is very little material regarding fingerprinting an encryption scheme based on ciphertext only. That's the subject of my (hobby) research and that's what ultimately led me to develop CryptoBench, since I couldn't find a freely available software that will provide multiple encryption algorithms and allow for low level control of the encryption parameters.

Developing good cryptographic software is hard -very hard- so, if CryptoBench works it's because of Wei Dai's Crypto++ library. When it doesn't, it's most likely my fault.

CryptoBench assumes you have a basic knowledge of cryptography and that you are familiar with terms like *hash*, *initialization vector*, *cipher-mode*, *asymmetric cryptography*, etc. and while the program is as intuitive and user-friendly as I could make it (considering I wasn't planning on releasing it to the public), it won't stop you from shooting yourself in the foot, so use it only on files you don't care to lose.

Installation

The CryptoBench software package is composed by this user's guide and the main executable `CrytoBench.exe`.

To install, just copy the files into an empty directory.

The software is certified to work on Microsoft Windows 2000, 2003 Server and XP (Home and Pro)

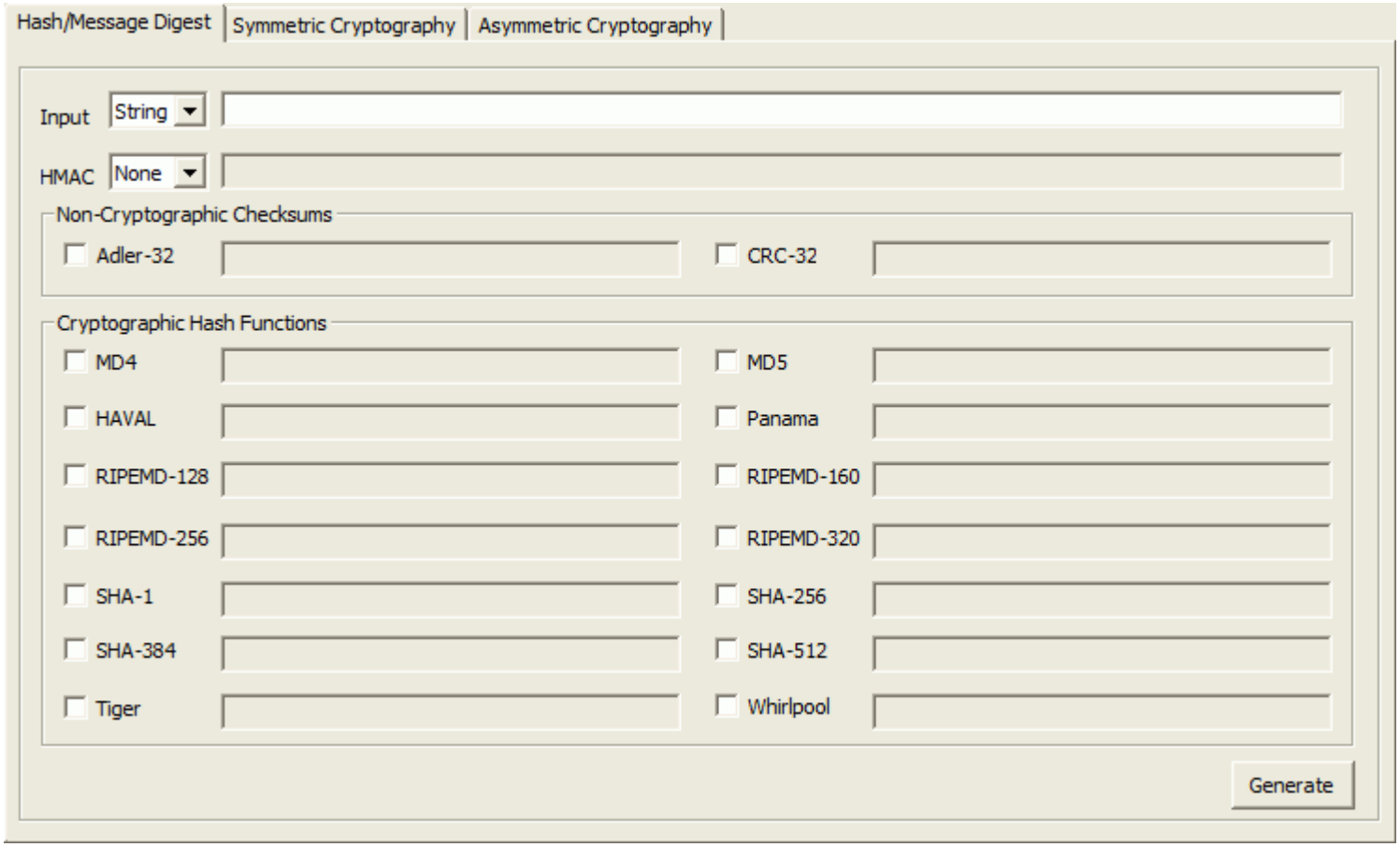
CryptoBench.exe v1.0 SHA-1: 61E8107A0E642F0756A186AD448AA4B40830C3B5

Usage

Hash/Message Digest Tab

14 cryptographic hashes and 2 non-cryptographic checksums can be generated for ASCII **String**s, **Hex**adecimal strings and **File**s by selecting the appropriate checkbox and clicking on **Generate**.

If you would like to produce HMACs for cryptographic hashes (not available for checksums), select the HMAC type, enter the authentication code you wish to use and click on Generate.



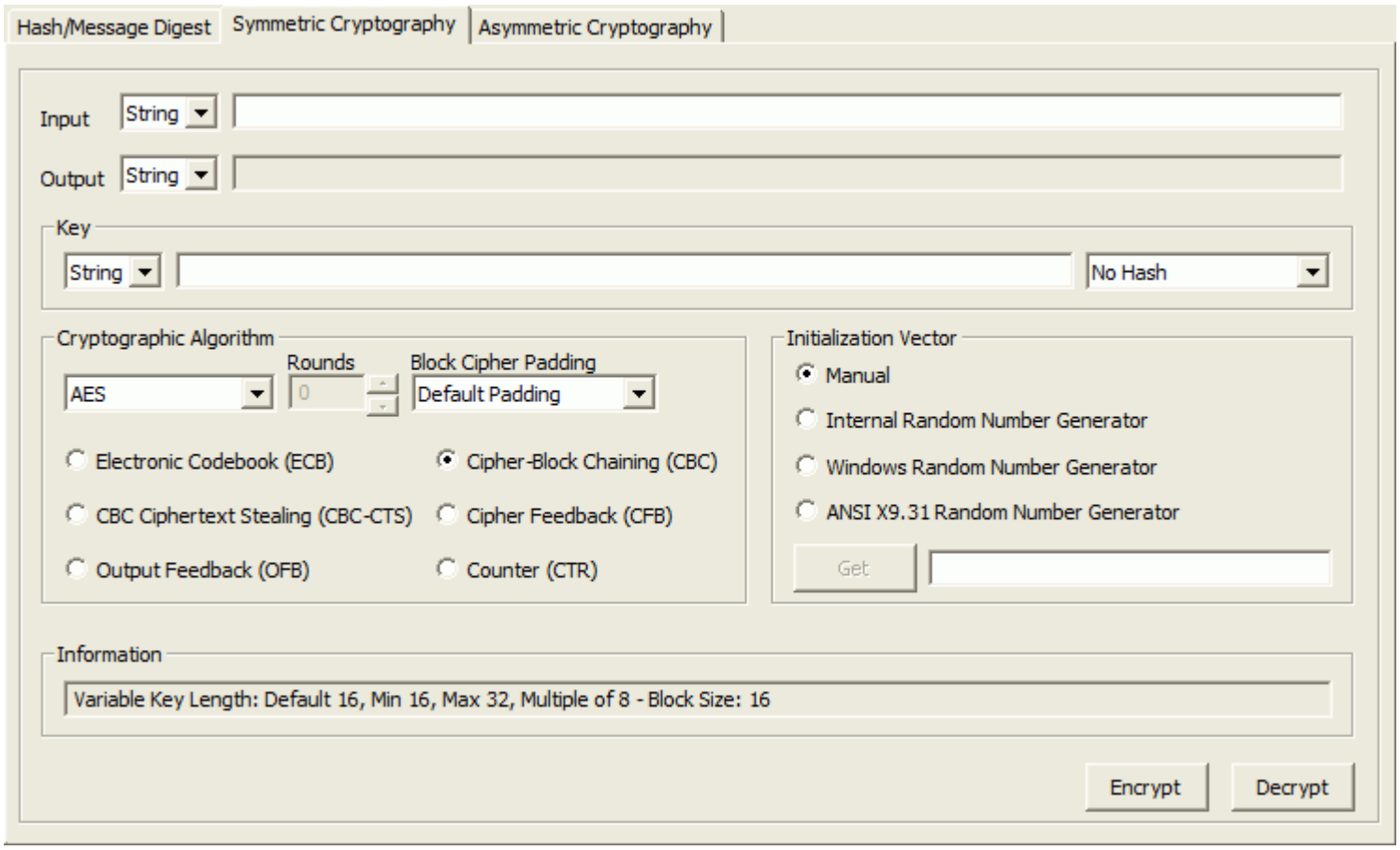
Symmetric Cryptography Tab

On this tab you can perform **Encrypt** and **Decrypt** operations with 29 different secret key schemes. The **Key** (password or passphrase) can be ASCII or hexadecimal and if you would like to use its hash instead of the raw key, select the appropriate algorithm from the drop list. Only hashes that produce a valid key size are displayed.

Each scheme has specific requirements regarding key size, padding and number of **Rounds** (enabled when the algorithm supports variable rounds) as shown in the **Information** field (length values are in bytes).

You also have the option to select the cipher-mode supported by the algorithm. If the cipher-mode calls for an **Initialization Vector**, your choices are:

- **Manual** to enter a hexadecimal value of size equal to the algorithm's **Block Size**,
- **Internal Random Number Generator** to use CryptoBench's internal algorithm,
- **Windows Random Number Generator** to use MS Windows algorithm (remember the infamous NSAKEY?) or
- **ANSI X9.31 Random Number Generator** to obtain a number that complies with ANSI's X9.31 standard.

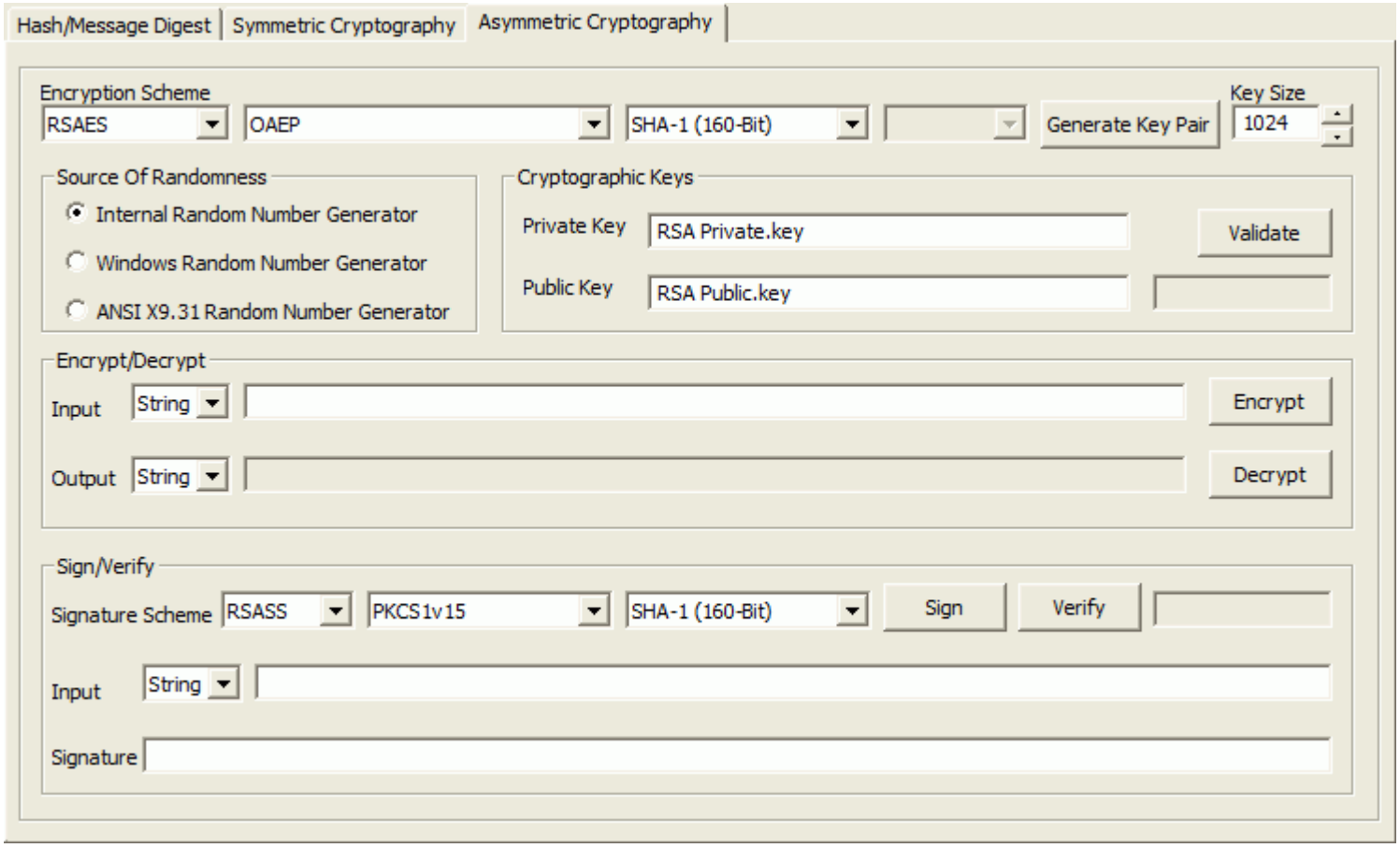


Asymmetric Cryptography Tab

6 asymmetric or public key algorithms are available. You can generate the keys by selecting the desired scheme, available parameters and **Key Size** and then clicking on the **Generate Key Pair**. Remember that the bigger the key, the longer it takes to calculate. The **Validate** button will verify the keys.

The **Encrypt** process will transform a string using the public key. **Decrypt** will reverse the process using the private key.

You can **Sign** and **Verify** strings and files by selecting the appropriate signature scheme that matches the public and private keys.



Known Issues

Here's the list of known problems that I'd like to resolve for the next release (if there ever is one)

- There are a few memory leaks here and there. Just a few Kb, nothing horrible.
- Filename validation and file manipulation is weak.
- The error messages aren't very clear.
- The user's guide could be much more complete.

Future Enhancements

New functionality and improvements I would like to add for the next versions (same caveat)

- Improve error handling.
- Optimize the code to improve speed and reduce the executable's size.
- Control the number of rounds for every symmetric algorithm.
- Secret sharing algorithms
- LUC public key cryptosystem and LUC signature system.
- Compression operations.
- Binary ⇔ Text encoding/decoding.
- Stream ciphers ARC4, SEAL, WAKE and Panama.

Bug Reports & Questions

E-mails are welcome at CryptoBench but I don't guarantee that I will read, much less reply, every message. I do this on my spare time and sometimes there's none.

The latest version is available at http://www.addario.org/cryptobench/

Acknowledgements and Trademarks

CryptoBench uses Dr. Brian R. Gladman's implementation of the MARS algorithm.

RSA, DESX, RC2, RC4, RC5, RC6, MD2, MD4, and MD5 are registered trademarks of RSA Security, Inc.

CAST-128 and CAST-256 are registered trademarks of Entrust Technologies, Inc.

Diamond2 Block Cipher is a trademark of Michael Paul Johnson.

IDEA is a registered trademark of Mediacrypt.

Other brand, product, and algorithm names may be trademarks or registered trademarks of their respective holders.

About my Research Project

Yes, I'm aware the whole purpose of cryptography is to provide an output that's statistically random and therefore without patterns or structure, and that it's unlikely I'll ever find a solution but in the end, I'm really more interested in the journey itself rather than the final destination.