# PureCrypt v1.0a

## License

In accordance with the Libtomcrypt license, since the majority of this code is from Libtomcrypt anyway (all I've done is hit it with a hammer till it's PB shaped), this software is hereby placed in the public domain.

Please note that in some countries, such as the United States, there are particular laws in place governing the use of, especially regarding importing or exporting, cryptographic libraries. I sincerely suggest you check on such legalities before you use this library or make any product that uses this library publicly available. I am not a lawyer, so please do not ask me questions about legal issues.

Libtomcrypt: The original library
(c) 2005, Tom St Denis
All rights reserved.

PureCrypt: The kludgey derived work by a part-time village idiot
(c) 2005, Jarratt Davis
All rights belong to Tom St Denis.

## Disclaimer

## Thanks

My thanks go out to Tom St. Denis for making Libtomcrypt, and thereby making my life much, much easier. Before I found his library I was writing my own from scratch and spending far more time on the project than I had allocated - whoops time buffer overflow error! Also I'd like the thank the following people (in no particular order) from the PureBASIC forum who took the time to assist a PB newbie: Pupil, MushroomHead, FloHimself, Traumatic and Fred (right said Fred)!

## What is it?

PureCrypt is essentially a rearrangement and a mangling of parts of Libtomcrypt v1.02 for use with PureBASIC as a native library. As it is derived from the Libtomcrypt library, rather than being the actual library itself, only certain hashes, ciphers etc. will be available. I will gradually add to PureCrypts functionality as needed.

This library exists purely because I needed some cryptographic functionality that is currently unavailable within PureBASIC. If you need a cryptographic library for a programming language other than PureBASIC, I suggest you go straight to the source and download the latest version of Libtomcrypt.

Currently PureCrypt supports the following hashes, ciphers etc.:
- SHA-1 hash.
- SHA-256 hash.
- SHA-512 hash.
- Blowfish cipher in ECB (Electronic Code Book) mode only.

Note: SHA-1 shouldn't be used in any situation where security is a high priority.

## What it isn't

A substitute for yak's milk, a golfing umbrella, or a mafia midget in a blue Armani suit.

## Installation

1. Find the PureBASIC "Library SDK" directory and run "LibraryMaker.exe".
2. For "Objects path", select the directory that contains both "PureCrypt.lib" and "PureCrypt.Desc".
3. Click on the "Start" button.
4. Select "PureCrypt.Desc" and click on the "Ok" button.
5. All done. Hopefully you didn't receive any error messages.
6. Now run the example/test programs and make sure you are getting the right output. If not, please send me a copy of the output and the specs of the machine you ran the examples on.

## To do

The library:
Add and enable library test mode. Or maybe remove it and just include test functions along the lines of the original Libtomcrypt.

Also include public/private key systems: Diffie-Hellman and RSA 64-bit as examples (or maybe even as part of the library).

Hashes:
MD4
MD5
Tiger
Whirlpool
Haval

Ciphers:
Cast5
DES
3DES
RC4
RC5
RC6
Skipjack
Safer
AES/Rijndael

Twofish
Gost

## PureBASIC API

### Hashes

Hash functions are used in many different ways.  Some of the more common uses include password systems and secure CRC style checking of files - I might write some simple examples of these if the need arises.  All of the hash functions allocate an "unsigned char" buffer of a particular size, depending on the hash algorithm, and return a pointer to that buffer.

*result = SHA1Hash(input, length)
Given input and the length of the input in bytes, this function returns a 20 byte encrypted hash.
eg.

```
Structure Sha1Struct
  StructureUnion
    ByteArray.b[20]
    WordArray.w[10]
    LongArray.l[5]
  EndStructureUnion
EndStructure
*result.Sha1Struct = SHA1Hash("abc", 3)
```

*result = SHA256Hash(input, length)
Given input and the length of the input in bytes, this function returns a 32 byte encrypted hash.
eg.

```
Structure Sha256Struct
  StructureUnion
    ByteArray.b[32]
    WordArray.w[16]
    LongArray.l[8]
  EndStructureUnion
EndStructure
*result.Sha256Struct = SHA256Hash("abc", 3)
```

*result = SHA512Hash(input, length)
Given input and the length of the input in bytes, this function returns a 64 byte encrypted hash.
eg.

```
Structure Sha512Struct
  StructureUnion
    ByteArray.b[64]
    WordArray.w[32]
    LongArray.l[16]
  EndStructureUnion
EndStructure
*result.Sha512Struct = SHA512Hash("abc", 3)
```

### Ciphers

Some of the more common uses of ciphers includes encrypting files, securing network-based communications etc.  Each cipher function allocates an "unsigned char" buffer of a particular size, depending on the ciphers requirements, and returns a pointer to that buffer.

```
*result = BlowfishEncode(password, length, input, rounds)
*result = BlowfishDecode(password, length, input, rounds)
```

Given the password, password length, input and number of rounds, this function takes an 8 plain/encrypted byte block and returns an 8 byte encrypted/decrypted block. The password has to be 8 or more bytes in length, and less than 56 bytes otherwise you will get a null result. The input given has to be 8 bytes in length. The rounds setting is either 0 (standard number of rounds), or 1 (16 rounds). The former is faster, the latter is more secure. The reason why this function doesn't take more than 8 bytes input is that I thought it best that the programmer decides what to do when the data they have for input doesn't fit perfectly into 8 byte blocks - especially as the Blowfish specification does not include how to handle that situation. It might also have something to do with the fact that is the way Tom dealt with it. Anyway I think it's better from the programmers point of view to have low level access to Blowfish as opposed to being limited by a higher level API.

eg.

```
Structure BFStruct
  StructureUnion
    ByteArray.b[8]
    WordArray.w[4]
    LongArray.l[2]
  EndStructureUnion
EndStructure
*result.BFStruct = BlowfishEncode("12345678", 8, "abcdefgh", 0)
```

## Possible bugs

Libtomcrypt can handle pretty much any type of "endian" machine you can throw at it. PureCrypt, on the other hand, currently cannot. PureCrypt is known to compile and run on "little endian" machines running MS Windows of some variant. Win2k Pro is the only version of windows this has been tested under so far. I will fix this as needed. I probably should have just written a wrapper for Libtomcrypt, however for some reason I was unable to convince Pelles C to compile Libtomcrypt properly. Probably because both the library and the compiler are too smart for me. PureCrypt isn't as smart, but it works. For me at least.

Also I might alter how the ciphers work so that there isn't an 8 byte block being allocated every time one of those functions is called. Actually I'm not convinced that I understand how PureBASIC's memory management works so there is a possibility that the hash functions are leaking as well. I need more information before I can determine whether or not this is the case. If I want to use leak detection software, I'm probably going to have to turn this into a console program and compile it first.

If you discover any bugs, or problems with this library, feel free to contact me via the address listed below. You will need to include relevant details such as what you did when the bug appeared, examples of code used (if needed), operating system etc.

## Tools used

Pelles C, a decent "free" C compiler for MS Windows operating systems. It can also cross-compile to WinCE for certain types of PDA processor.

PureBASIC's Library Maker was used to rip apart the compiled PureCrypt library and turn it into a PureBASIC native library.

## Where can I get the latest version?

The latest version of Libtomcrypt can be found at:
http://libtomcrypt.org

The latest version of PureCrypt can be found at:
http://jazz.hal9000.net.au

You do not need Libtomcrypt to compile PureCrypt.  All code required to compile PureCrypt has been included in this archive.

## How can I contact you?

I can be contacted via:
jazz@hal9000.net.au

Please make sure that your subject line contains: "PureCrypt", otherwise there is a strong chance I will lose your email amongst the flood of spam that account currently receives.  Please do not contact me with questions regarding Libtomcrypt, and do not contact Tom regarding PureCrypt.  I am sure he is busy enough without having to answer questions about other peoples kludgey hacks.