1: LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
Const
PROV_RSA_FULL = 1;
PROV_RSA_AES = 24;
CRYPT_VERIFYCONTEXT = $F0000000;
CRYPT_NEWKEYSET = $00000008;
// use with PROV_RSA_AES To get SHA-2 values.
http://www.tek-tips.com/faqs.cfm?id=7423
CALG_SHA256 = $0000000C;
CALG_SHA384 = $0000000D;
CALG_SHA512 = $0000000E;
HP_HASHVAL = $0002;
CRYPT32 = 'crypt32.dll';
MS_ENHANCED_PROV = 'Microsoft Enhanced Cryptographic Provider v1.0';
HASH256TEST= 'The quick brown fox jumps over the lazy dog';

By the way with OpenSSL and the well known libeay32.dll a further solution exists (just a type extract below) but in this article we focus on Win DLL.

type
SHA_CTX2 = Record
Unknown: Array[0..5] of LongWord;
State: Array[0..4] of LongWord;
Count: UInt64;
Buffer: Array[0..63] of Byte;
End;

function SHA256_CTX(nameform: DWord; namebuffer: array of char;
var nsize: DWord): boolean;
external 'SHA256@libeay32.dll stdcall';

function libeay32version: pchar;
external 'SSLeay_version@libeay32.dll stdcall';

procedure SHA256Init(var Context: SHA_CTX2);
external 'SHA256_Init@libeay32.dll stdcall';

Probably the best way to get started with this sort of thing is to create a small test DLL, create a few functions with known parameters and call it. In our case we need 6 functions to declare:

function CryptAcquireContext(out phProv: TCryptProv; szContainer:
PChar; szProvider: PChar; dwProvType: DWord;
dwFlags: DWord): boolean; //stdcall;
External 'CryptAcquireContextA@advapi32.dll stdcall';

function CryptCreateHash(phProv: TCryptProv; AlgId: TAlgID; hKey:
TCryptKey; dwFlags: DWord; out phHash: TCryptHash): boolean;
External 'CryptCreateHash@advapi32.dll stdcall';

function CryptHashData(phHash: TCryptHash; aRes: PChar; dwDataLen:
DWord; dwFlags: DWord): boolean; //stdcall;
External 'CryptHashData@advapi32.dll stdcall';

function CryptGetHashParam(phHash: TCryptHash; dwParam: Dword;
out pbdata: TSHA_RES3;
var dwDataLen: DWord; dwFlags: DWord): Boolean; //stdcall;
External 'CryptGetHashParam@advapi32.dll stdcall';
The quality of a DLL function is the parameter documentation. So much the better you find a well based documentation concerning view the parameter and return types of a function!

The Win module file format only provides a single text string to identify each function. There is no structured way to list the number of parameters, the parameter types, or the return type. However, some languages do something called function "decoration" or "mangling", which is the process of encoding information into the text string.

Our first and important call is CryptAcquireContext(): The CryptAcquireContext function is used to acquire a handle to a particular key container within a particular cryptographic service provider (CSP). A CSP is an independent module that performs all cryptographic operations.

At least one CSP is required with each application that uses cryptographic functions. A single application can occasionally use more than one CSP. This returned handle is used in calls to CryptoAPI functions that use the selected CSP, so the first 2 calls are:

The following code assumes that the handle of a cryptographic context has been acquired and that a hash object has been created and its handle (hHash) is available. So we dont need any pointers and I can script it in maxBox, Python or Powershell with call by references and a strict PChar with the ByteArray

TSHA_RES3 = Array[1..32] of Byte;

The CryptCreateHash() function initiates the hashing of a stream of data. This handle is used in subsequent calls to CryptHashData and CryptHashSessionKey to hash session keys and other streams of data that we get we a fileToString();

The last step is to get the hash with CryptGetHashParam:

I do always evaluate on each function the boolean return value to make sure. When was the last time you saw the return value for a function checked?

The CryptGetHashParam function retrieves data that governs the operations of a hash object. The actual hash value can be retrieved by using this function. Don't forget to free handles and structure:
141: println('Destroy hash-hnd: '+botostr(CryptDestroyHash(hhash)));
142: println('Crypt_ReleaseContext: '+botostr(CryptReleaseContext(hProv, 0)));
143: println('Crypto dll: SHA256: '+binToHEX_Str(shares3));
144: println('Test length: '+itoa(length(bin)));
145: writeln('test length: '+itoa(length(bin)));
146: writeln('test length: '+itoa(length(bin)));
147: writeln('test length: '+itoa(length(bin)));
148: writeln('test length: '+itoa(length(bin)));
149: writeln('test length: '+itoa(length(bin)));
150: writeln('test length: '+itoa(length(bin)));
151: writeln('test length: '+itoa(length(bin)));
152: writeln('test length: '+itoa(length(bin)));
153: writeln('test length: '+itoa(length(bin)));
154: writeln('test length: '+itoa(length(bin)));
155: writeln('test length: '+itoa(length(bin)));
156: writeln('test length: '+itoa(length(bin)));
157: writeln('test length: '+itoa(length(bin)));
158: writeln('test length: '+itoa(length(bin)));
159: writeln('test length: '+itoa(length(bin)));
160: writeln('test length: '+itoa(length(bin)));
161: writeln('test length: '+itoa(length(bin)));
162: writeln('test length: '+itoa(length(bin)));
163: writeln('test length: '+itoa(length(bin)));
164: writeln('test length: '+itoa(length(bin)));
165: writeln('test length: '+itoa(length(bin)));
166: writeln('test length: '+itoa(length(bin)));
167: writeln('test length: '+itoa(length(bin)));
168: writeln('test length: '+itoa(length(bin)));
169: writeln('test length: '+itoa(length(bin)));
170: writeln('test length: '+itoa(length(bin)));
171: writeln('test length: '+itoa(length(bin)));
172: writeln('test length: '+itoa(length(bin)));
173: writeln('test length: '+itoa(length(bin)));
174: writeln('test length: '+itoa(length(bin)));
175: writeln('test length: '+itoa(length(bin)));
176: writeln('test length: '+itoa(length(bin)));
177: writeln('test length: '+itoa(length(bin)));
178: writeln('test length: '+itoa(length(bin)));
179: writeln('test length: '+itoa(length(bin)));
180: writeln('test length: '+itoa(length(bin)));
181: writeln('test length: '+itoa(length(bin)));
182: writeln('test length: '+itoa(length(bin)));
183: writeln('test length: '+itoa(length(bin)));
184: writeln('test length: '+itoa(length(bin)));
185: writeln('test length: '+itoa(length(bin)));
186: writeln('test length: '+itoa(length(bin)));
187: writeln('test length: '+itoa(length(bin)));
188: writeln('test length: '+itoa(length(bin)));
189: writeln('test length: '+itoa(length(bin)));
190: writeln('test length: '+itoa(length(bin)));
191: writeln('test length: '+itoa(length(bin)));
192: writeln('test length: '+itoa(length(bin)));
193: writeln('test length: '+itoa(length(bin)));
194: writeln('test length: '+itoa(length(bin)));
195: writeln('test length: '+itoa(length(bin)));
196: writeln('test length: '+itoa(length(bin)));
197: writeln('test length: '+itoa(length(bin)));
198: writeln('test length: '+itoa(length(bin)));
199: writeln('test length: '+itoa(length(bin)));
200: writeln('test length: '+itoa(length(bin)));
201: Next we step to the double SHA256 called SHA256D and block generation. Its important to realize that block generation is not a long, set problem (like doing a million hashes), but more like a lottery. Each hash basically gives you a random number between 0 and the maximum value of a 256-bit number (which is huge). If your hash is below the target, then you win. If not, you increment the nonce (completely changing hash) and try again to mine. With the SHA256 lib of PascalCoin the function is simpler to use in comparison to the DLL:

204: Example:

205: sr:= filetoString(Exepath+'maxbox4.exe')

206: writeln(SHA256ToStr(CalcSHA256(sr)))

207: or more simpler with an alias in maxbox:

208: writeln(GetSHA256(sr))

209: function GetSHA256(Msg: AnsiString): string; //overload;

210: var Stream: TMemoryStream;

211: begin

212: Stream:= TMemoryStream.Create;

213: try

214: Stream.WriteByte(PAnsiChar(Msg)^,Length(Msg));

215: Stream.Position:= 0;

216: Result:= SHA256ToStr(CalcSHA256(Stream));

217: finally

218: Stream.Free;

219: end;

220: end;

221: end;

222: Imagine now the double hash. It is also a crypto hash function, mainly used to ensure integrity of the encrypted message of the block, i.e. if you manipulate the message it will be visible, because the hash will also change. It also guarantees the uniqueness of a message or block of data.

226: In terms of Bitcoin or PascalCoin, it guarantees the uniqueness of each coin. So you cannot just copy the same set of data over and over again. The function is

228: Function CalcDoubleSHA256(Msg : AnsiString) : TSHA256HASH;

230: Function SHA256ToStr( Hash : TSHA256HASH) : String;

232: sr:= filetoString(Exepath+'maxbox4.exe')

233: writeln(SHA256ToStr(CalcDoubleSHA256(sr)))

234: >>> 7DECBAE2 2C539395 8C3707E9 080281CE 06F45779 BFBBB81F 9954E031 982A505E

236: It appears to be double SHA256. In other words:

238: SHA256D(x) =SHA256(SHA256(x)).

240: SHA256 (and thus SHA256D) is a cryptographic hash function (it performs a 1-way transformation on an input value) that forms the proof-of-work algorithm used when adding blocks to the blockchain in bitcoin. You are hashing the hexadecimal representation of the first hash. You need to hash the actual hash, the binary data that the hex represents.

242: Just semantics, but to avoid a common misunderstanding: SHA256 and others does hashing, not encoding. Encoding is something entirely different. For one it implies it can be decoded, whereas hashing is strictly a one-way (and destructive) operation!
244: There’s no guarantee that every single value in a hash function is reachable, depending on the hash algorithm. For some cryptographic algorithms, it is likely that less than half of the output keyspace is reachable for any given input. However, this may not hold true for every single cryptographic hash algorithm, and it is computationally unfeasible to verify.

245: There is also no proof that every output of common has functions is reachable for some input, but it is expected to be true. No method better than brute force is known to check this, and brute force is entirely impractical.

246: Ref:

247: http://www.pascalcoin.org/

248: https://en.bitcoin.it/wiki/Target

250: https://bitcoinwisdom.com/

251: https://maxbox4.wordpress.com

252: http://www.xorbin.com/tools/sha256-hash-calculator

253: http://www.softwareschule.ch/examples/sha256.txt

254:


257:

258: https://maxbox4.wordpress.com/2017/08/23/five-steps-to-get-sha256-or-other-ciphers/

259:

260: Doc: SHA256 Lib Interface:

262: procedure SIRegister_USha256(CL: TPSPascalCompiler);

263: begin

264: type TSHA256HASH, 'array[0..7] of Cardinal';

265: type TSHAChunk, 'array[0..7] of Cardinal';

266: //TSHA256HASH = array[0..7] of Cardinal;

267: Function CalcDoubleSHA256( Msg : AnsiString ) : TSHA256HASH;

268: Function CalcSHA256( Msg : AnsiString ) : TSHA256HASH;

269: Function CalcSHA2561( Stream : TStream ) : TSHA256HASH;

270: Function SHA256ToStr( Hash : TSHA256HASH ) : String;

271: Function CanBeModifiedOnLastChunk( MessageTotalLength: Int64; var startBytePos : integer) : Boolean';

272: Procedure PascalCoinPrepareLastChunk( const messageToHash : AnsiString; var stateForLastChunk : TSHA256HASH; var bufferForLastChunk : TSHAChunk);

273: Procedure ExecuteLastChunk( const stateForLastChunk: TSHA256HASH; const bufferForLastChunk: TSHAChunk; nPos : Integer; nOnce, Timestamp : Cardinal ) : TSHA256HASH;

274: Procedure ExecuteLastChunkAndDoSha256( const stateForLastChunk: TSHA256HASH; const bufferForLastChunk: TSHAChunk; nPos : Integer; nOnce, Timestamp : Cardinal ) : TSHA256HASH;

275: Procedure PascalCoinExecuteLastChunkAndDoSha256( const stateForLastChunk : TSHA256HASH; const bufferForLastChunk: TSHAChunk; nPos : Integer; nOnce, Timestamp : Cardinal; var ResultSha256 : AnsiString);

276: Function Sha256HashToRaw( const hash : TSHA256HASH ) : AnsiString;

277: Function GetSHA256( Msg : AnsiString ) : string;

278: function GetDriveNumber(const Drive: string): Integer;

279: function HardDiskSerial(const Drive: string): DWORD;

280: function IsDriveReady2(const Drive: string): Boolean;

281: function Touchfile(const FileName: string): Boolean;

282: function URLFromShortcut(const Shortcut: string): string;