**/** C++ function description:**

name: AEGIS_check

function: performs AEGIS-algorithm Encryption process then compares the result with the RTL output

arguments: the RTL inputs (msglen, adlen, key, IV, AD, plain_text), the RTL encryption outputs (cipher, tag) as strings

arguments_type: string/string array

return: 1/0 (int) depending on the C++ output equals/not_equal the RTL output

**/

```
1049  extern "C" int AEGIS_check(char* msg_len_sv, char* ad_len_sv, char* key_sv, char* IV_sv, char* AD_sv, char** plain_text_sv,
1050                             char** cipher_out_sv, char* tag_out_sv)
1051  {
1052      unsigned long long msglen, adlen, cipherlen;
1053      uint8 key[32], IV[32], AD[16], Tag[16], Tag_sv[16];
1054
1055      // transform string to integer value
1056      msglen=stoi(msg_len_sv);
1057      cipherlen = msglen;
1058      adlen=stoi(ad_len_sv);
1059
1060      int N = msglen / 8; /* to determine the arrays size (N) */
1061      uint8* plaintext = new uint8[N];
1062      uint8* encryptedOutput = new uint8[N];
1063      uint8* decryptedOutput = new uint8[N];
1064      uint8* cipher_sv = new uint8[N];
1065
1066      int msg_or_cipher_blocks_num = msglen/128;
1067      int error_count = 0;
1068      string tmp_str;
```

```
1070      // key, transform from hex-string to hex-value
1071      tmp_str=key_sv;
1072      for (int i = 0; i < 64; i+=2)
1073      {
1074          unsigned int byteValue;
1075          stringstream ss;
1076          ss << std::hex << tmp_str.substr(i,2);
1077          ss >> byteValue;
1078          key[i/2] = static_cast<uint8>(byteValue);
1079      }
1080
1081      // IV, transform from hex-string to hex-value
1082      tmp_str=IV_sv;
1083      for (int i = 0; i < 64; i+=2)
1084      {
1085          unsigned int byteValue;
1086          stringstream ss;
1087          ss << std::hex << tmp_str.substr(i,2);
1088          ss >> byteValue;
1089          IV[i/2] = static_cast<uint8>(byteValue);
1090      }
1091
```

```
1092      // AD, transform from hex-string to hex-value
1093      tmp_str=AD_sv;
1094      for (int i = 0; i < 32; i+=2)
1095      {
1096          unsigned int byteValue;
1097          stringstream ss;
1098          ss << std::hex << tmp_str.substr(i,2);
1099          ss >> byteValue;
1100          AD[i/2] = static_cast<uint8>(byteValue);
1101      }
1102
1103      // SV tag, transform from hex-string to hex-value
1104      tmp_str=tag_out_sv;
1105      for (int i = 0; i < 32; i+=2)
1106      {
1107          unsigned int byteValue;
1108          stringstream ss;
1109          ss << std::hex << tmp_str.substr(i,2);
1110          ss >> byteValue;
1111          Tag_sv[i/2] = static_cast<uint8>(byteValue);
1112      }
1113
```

```
1114      // plain_text, transform from hex-string to hex-value
1115      for (int j = 0; j < msg_or_cipher_blocks_num; j++)
1116      {
1117          tmp_str=plain_text_sv[j];
1118          for (int i = 0; i < 32; i+=2)
1119          {
1120              unsigned int byteValue;
1121              stringstream ss;
1122              ss << std::hex << tmp_str.substr(i,2);
1123              ss >> byteValue;
1124              plaintext[j*16+i/2] = static_cast<uint8>(byteValue);
1125          }
1126      }
1127
1128      // SV cipher, transform from hex-string to hex-value
1129      for (int j = 0; j < msg_or_cipher_blocks_num; j++)
1130      {
1131          tmp_str=cipher_out_sv[j];
1132          for (int i = 0; i < 32; i+=2)
1133          {
1134              unsigned int byteValue;
1135              stringstream ss;
1136              ss << std::hex << tmp_str.substr(i,2);
1137              ss >> byteValue;
1138              cipher_sv[j*16+i/2] = static_cast<uint8>(byteValue);
1139          }
1140      }
1141
```

```
1142      // Call the encrypt function
1143      aegisEncrypt(&msglen, adlen, AD, plaintext, encryptedOutput, Tag, key, IV);
1144
1145      // Comparing the SV cipher and tag with the C++ function cipher (encryptedOutput) and tag (Tag)
1146      for (int i = 0; i < N; i++) {
1147          if(cipher_sv[i] != encryptedOutput[i]) error_count++;
1148      }
1149      for (int i = 0; i < 16; i++) {
1150          if(Tag_sv[i] != Tag[i]) error_count++;
1151      }
1152
1153      return (error_count) ? 0 : 1;
1154  }
```