

上海黄金交易所

数据库存储加密 C 接口设计

V1.02

杭州信雅达科技有限公司

编制日期：2025 年 7 月 18 日

目录

1、接口说明	3
1.1.设置日志配置	3
1.2.建立与设备的连接	3
1.3.断开与设备的连接	4
1.4.短数据 hash 接口	4
1.5.长数据 hash 接口	5
1.6.短数据加解密接口	6
1.7.批量数据加解密接口	6
1.8.长数据加解密接口	7
1.9.批量数据 MAC 接口	9
1.10.长数据 MAC 接口	9
1.11.短数据 MAC 接口	11
1.12.获取版本信息	11
1.13.关闭句柄	12
1.14.短数据加密并输出密文校验值	12
1.15.短数据解密并对密文校验值进行核对	12
1.16.批量数据加密并输出密文校验值	13
1.17.批量数据解密并对密文校验值进行核对	14
2、错误码	14

1、接口说明

1.1.设置日志配置

功能：

设置日志相应的配置。

函数原型：

```
int SYD_SetLogConfig (  
    char*   pcLogPath,  
    int      iLogPathLen,  
    int      iLogLevel);
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

pcLogPath: 输入。日志存储路径

iLogPathLen: 输入。日志路径长度。

iLogLevel: 输入。日志级别设置

0: 不记录日志

1: 记录网络层错误日志

2: 记录接口错误日志及网络层错误日志

3: 记录正确日志及错误日志

1.2.建立与设备的连接

功能：

建立到指定地址和端口号的设备的 TCP/IP 连接，返回 TCP/IP 连接句柄。如果采用短连接，交易结束，要调用 SYD_Disconnect_Ex 进行释放连接。

函数原型：

```
int SYD_Connect_Ex(  
    char*   pcIpList[],  
    int      iPortList[],  
    int      iSize,  
    int      iConnectTimeOut,  
    int      iDealTimeOut,  
    int      iThreadCount,  
    void**   phHandle);
```

返回值：0 表示成功；非 0 表示失败；（只要有设备可用则返回 0）

参数说明：

pcIpList: 输入。设备的 IP 地址，（可输入多个地址，用于主备，高可用）

iPortList: 输入。设备的端口号，（可输入多个端口号，与 pcIpList 参数一一对应，用于主备，高可用）。

iSize: 输入。表示 IP 地址及端口号的个数。(默认将第一个 IP 地址及端口号当主加密设, 其余为备用设备)

iConnectTimeOut: 输入。连接超时, 单位毫秒。

iDealTimeOut: 输入。交易接口超时设定, 单位毫秒(不超过 10ms)。

iThreadCount: 输入。线程池中的线程个数。
 若 iThreadCount=0, 表示使用短连接
 若 iThreadCount>0, 表示使用连接池 (适用于长连接, 最大不超过 16 个线程)

phHandle: 输出。与设备建立的连接句柄。
 (该句柄支持数据重发, 降低丢包概率; 支持主机可选; 支持超时时间设置, 支持高可用, 用于新增接口调用)

1.3.断开与设备的连接

功能:

关闭与设备的 TCP/IP 连接

函数原型:

```
int SYD_DisConnect_Ex(
    void* phHandle);
```

返回值: 0 表示成功; 非 0 表示失败;

参数说明:

phHandle: 输入。 与设备建立的连接句柄。

1.4.短数据 hash 接口

功能:

对指定数据进行 SM3 哈希处理。(数据内容 1K 内则使用短数据接口)。

函数原型:

```
int SYD_SM3_Hash_ShortData(
    void*   phHandle,
    char*   pcData,
    int     iDataLen,
    char*   pcHash);
```

返回值: 0 表示成功; 非 0 表示失败;

参数说明:

phHandle: 输入。 与设备建立的连接句柄。

pcData: 输入。 要加密的数据。

iDataLen: 输入。 要加密的数据长度 (长度必须大于 0 且最大不超过 1K)。

pcHash: 输出。 哈希值 固定长度 32 字节。

1.5.长数据 hash 接口

1.5.1 初始化 hash

功能：

初始化 SM3 哈希。

函数原型：

```
int SYD_SM3_Hash_Init(  
    void** phHash);
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

phHash： 输出。哈希过程数据句柄。

（该句柄内存由 SYD_CloseHandle 进行释放）

1.5.2 多组数据 hash

功能：

对多个分组的数据进行 SM3 哈希运算。

函数原型：

```
int SYD_SM3_Hash_Update(  
    void* phHandle,  
    char* pcData,  
    int iDataLen,  
    void* phHash);
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

phHandle： 输入。与设备建立的连接句柄。

pcData： 输入。要哈希的数据。

iDataLen： 输入。要哈希的数据长度[0, 1k]。

phHash： 输入/输出。哈希过程数据句柄。

（该句柄内存由 SYD_CloseHandle 进行释放）

1.5.3 结束 hash

功能：

结束多个分组的 SM3 哈希

函数原型：

```
int SYD_SM3_Hash_Final(  
    void* phHandle,  
    char* pcHash,  
    void* phHash);
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

phHandle： 输入。与设备建立的连接句柄。

pcHash： 输出。哈希值（固定长度 32 字节）

phHash： 输入。哈希过程数据句柄。

（该句柄内存由 SYD_CloseHandle 进行释放）

1.6.短数据加解密接口

功能：

对指定数据进行 SM4 加解密处理（数据内容 1K 内则使用短数据接口）

函数原型：

```
int SYD_SM4_ShortData(  
    void* phHandle,  
    int iKeyIndex,  
    char* pcData,  
    int iDataLen,  
    int iFlag,  
    char* pcOutData,  
    int* piOutDataLen);
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

phHandle： 输入。与设备建立的连接句柄。

iKeyIndex： 输入。会话密钥索引值。

pcData： 输入。报文密文或明文。

iDataLen： 输入。报文输入长度，以字节为单位。

（作为明文输入时，长度必须大于 0 且最大不超过 1K，作为密文长度输入时，长度必须大于 0，且最大不超过 1K+16，必须为 16 的整倍数）

iFlag： 输入。加解密标识。0:表示 ECB 解密；1:表示 ECB 加密；

pcOutData： 输出。报文明文或密文。

piOutDataLen： 输入/输出。报文长度。（作为输入时表示 pOutData 缓存的最大长度；作为输出时加密结果 16 的整数倍，解密会去掉 PKCS7 填充）。

1.7.批量数据加解密接口

功能：

对指定批量数据进行 SM4 加解密处理。

函数原型：

```
int SYD_SM4_BatchData(  
    void* phHandle,  
    int iKeyIndex[],  
    char* pcData[],  
    int iDataLen[],
```

```

        int    iSize,
        int    iFlag,
        char*  pcOutData[],
        int    iOutDataLen[]);

```

返回值：0 表示成功；非 0 表示失败；（有一条数据失败则报错）

参数说明：

phHandle： 输入。 与设备建立的连接句柄。

iKeyIndex： 输入。 会话密钥索引值数组。

pcData： 输入。 要加密的报文明文或密文数组。

iDataLen： 输入。 报文长度数组，以字节为单位。

（作为明文输入时，长度必须大于 0 且最大不超过 1K， 作为密文长度输入时，长度必须大于 0，且最大不超过 1K+16，必须为 16 的整倍数）

iSize： 输入。 表示数组的大小。

iFlag： 输入。 加解密标识，0:表示 ECB 解密；1:表示 ECB 加密；

pcOutData： 输出。 输出的报文密文或明文数组。

iOutDataLen： 输入/输出。 报文长度数组。（作为输入时表示 pcOutData 缓存的最大长度；作为输出时加密结果 16 的整数倍，解密会去掉 PKCS7 填充）

1.8.长数据加解密接口

1.8.1 初始化 SM4 加解密

功能：

数据加解密初始化。

函数原型：

```

int SYD_SM4_LongDataInit(
    int    iKeyIndex,
    int    iFlag,
    void** phProcHandle);

```

返回值：0 表示成功；非 0 表示失败；

参数说明：

iKeyIndex： 输入。会话密钥索引值。

iFlag： 输入。加解密标识，0:表示 ECB 解密；1:表示 ECB 加密；

phProcHandle： 输出。过程数据句柄。

（该句柄内存由 SYD_CloseHandle 进行释放）

1.8.2 多组数据 SM4 加解密

功能：

多个分组数据的加解密操作

函数原型：

```

int SYD_SM4_LongDataUpdate(

```

```

void* phHandle,
char* pcData,
int iDataLen,
char* pcOutData,
int* piOutDataLen,
void* phProcHandle);

```

返回值：0 表示成功；非 0 表示失败；

参数说明：

phHandle: 输入。与设备建立的连接句柄。

pcData: 输入。要加密的数据。

iDataLen: 输入。要加密的数据长度。
 (加密长度: [0, 1k])
 (解密长度: [0, 1k + 16])
 注意：作为解密时，输入数据累积的总长度不能为 0)

pcOutData: 输出。作为加密时输出密文数据，作为解密时输出明文数据。

piOutDataLen: 输入/输出。
 作为输入时表示 pcOutData 缓冲区大小。
 作为输出时表示明文或密文数据长度。

phProcHandle: 输入/输出。过程数据句柄。
 (该句柄内存由 SYD_CloseHandle 进行释放)

1.8.3 结束 SM4 加解密

功能：

结束多个分组数据的加解密操作

函数原型：

```

int SYD_SM4_LongDataFinal(
void* phHandle,
char* pcOutData,
int* piOutDataLen,
void* phProcHandle);

```

返回值：0 表示成功；非 0 表示失败；

参数说明：

phHandle: 输入。与设备建立的连接句柄。

pcOutData: 输出。作为加密时输出密文数据，作为解密时输出明文数据。

piOutDataLen: 输入/输出。
 作为输入时表示 pcOutData 缓冲区大小。
 作为输出时表示明文或密文数据长度。

phProcHandle: 输入。过程数据句柄。
 (该句柄内存由 SYD_CloseHandle 进行释放)

1.9.批量数据 MAC 接口

功能：

对批量数据进行 SM4 MAC 计算或校验处理。

函数原型：

```
int SYD_SM4Mac_BatchData(  
    void*   phHandle,  
    int     iKeyIndex[],  
    char*   pcData[],  
    int     iDataLen[],  
    int     iSize,  
    char*   pcMac[],  
    int     iCheckRet[]);
```

返回值：0 表示成功；非 0 表示失败；（有一条数据失败则报错）

参数说明：

phHandle: 输入。 与设备建立的连接句柄。
iKeyIndex: 输入。 计算 MAC 所需要的密钥索引值数组。
pcData: 输入。 要做 MAC 的数据。
iDataLen: 输入。 要做 MAC 的数据长度[0, 1k]。
iSize: 输入。 表示数组的大小。
pcMac: 输出/输入。输出/输入的 mac 值数组，固定长度 16 字节。
iCheckRet: 输出。 若 iCheckRet 为 NULL, 则输出真实的 MAC 值，若 iCheckRet 不为 NULL, 则会将 Mac 值作为输入校验 Mac, 校验的结果放到 iCheckRet 中，为 0 表示校验成功，非 0 表示校验失败。（这里 iCheckRet 表示的是一个 int 类型的数组）

1.10.长数据 MAC 接口

1.10.1 初始化 MAC

功能：

初始化 SM4 MAC

函数原型：

```
int SYD_SM4Mac_LongDataInit(  
    int     iKeyIndex,  
    void**  phMac);
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

iKeyIndex: 输入。 会话密钥索引值。
phMac: 输出。 MAC 过程数据句柄。
（该句柄内存由 SYD_CloseHandle 进行释放）

1.10.2 多组数据 MAC

功能:

多组数据 SM4 MAC

函数原型:

```
int SYD_SM4Mac_LongDataUpdate(  
    void* phHandle,  
    char* pcData,  
    int iDataLen,  
    void* phMac);
```

返回值: 0 表示成功; 非 0 表示失败;

参数说明:

phHandle: 输入。与设备建立的连接句柄。
pcData: 输入。要做 MAC 的数据。
iDataLen: 输入。要做 MAC 的数据长度[0, 1k]。
phMac: 输入/输出。MAC 过程数据句柄。
(该句柄内存由 SYD_CloseHandle 进行释放)

1.10.3 结束数据 MAC

功能:

结束多组数据 SM4 MAC

函数原型:

```
int SYD_SM4Mac_LongDataFinal(  
    void* phHandle,  
    char* pcMac,  
    int* piCheckRet,  
    void* phMac);
```

返回值: 0 表示成功; 非 0 表示失败;

参数说明:

phHandle: 输入。与设备建立的连接句柄。
pcMac: 输出。MAC 值固定长度 16 字节。
piCheckRet: 输出。如果 piCheckRet 为 NULL, 则输出真实的 MAC 值, 如果 piCheckRet 不为 NULL, 则会将 Mac 值作为输入校验 Mac, 校验的结果放到 piCheckRet 中, 为 0 表示校验成功, 非 0 表示校验失败。(这里 piCheckRet 表示的是一个 int 类型的指针对象)
phMac: 输入。MAC 过程数据句柄。(该句柄内存由 SYD_CloseHandle 进行释放)

1.11.短数据 MAC 接口

功能:

对数据进行 SM4 MAC 计算或校验处理。

函数原型:

```
int SYD_SM4Mac_ShortData(  
    void* phHandle,  
    int    iKeyIndex,  
    char* pcData,  
    int    iDataLen,  
    char* pcMac,  
    int*   piCheckRet);
```

返回值: 0 表示成功; 非 0 表示失败;

参数说明:

phHandle:	输入。与设备建立的连接句柄。
iKeyIndex:	输入。计算 mac 所需要的密钥索引值
pcData:	输入。计算 mac 所需要的数据。
iDataLen	输入。数据长度, 以字节为单位。 [0, 1k]
pcMac:	输出/输入。输出/输入的 mac 值, 固定长度 16 字节
piCheckRet	输出。若 piCheckRet 为 NULL, 则只计算 MAC 值, 若 piCheckRet 不为 NULL, 则会将 Mac 值作为输入校验 Mac, 校验的结果放到 piCheckRet 中, 为 0 表示校验成功, 非 0 表示校验失败。(这里 piCheckRet 表示的是一个 int 类型的指针对象)

1.12.获取版本信息

功能:

获取版本信息 (server:x.xx\nlibsunyard.so:x.xx.xx)

函数原型:

```
int SYD_GetVersionInfo(  
    void* phHandle,  
    char* pcVersion,  
    int*   piVersionLen)
```

返回值: 0 表示成功; 非 0 表示失败;

phHandle:	输入。与设备建立的连接句柄。
pcVersion:	输入。版本信息 (服务端版本号及 so 版本号)
piVersionLen:	输入/输出。版本信息长度。 (作为输入时表示 pcVersion 缓存的最大长度; 作为输出时为 pcVersion 真实大小)

1.13.关闭句柄

功能：

关闭哈希过程数据句柄，SM4 加解密过程数据句柄，MAC 过程数据句柄

函数原型：

```
int SYD_CloseHandle(  
    void *hHandle);
```

参数说明：

hHandle： 输入。过程数据句柄

1.14.短数据加密并输出密文校验值

功能：

短数据加密并计算密文的 MAC 值

函数原型：

```
int SYD_SM4_Encrypt_Mac_ShortData(  
    void*   phHandle,  
    int     iEncKeyIndex,  
    int     iCheckKeyIndex,  
    char*   pcData,  
    int     iDataLen,  
    char*   pcEncData,  
    int*    piEncDataLen,  
    char*   pcMac);
```

返回值：0 表示成功；非 0 表示失败；

参数说明：

phHandle： 输入。 与设备建立的连接句柄。

iEncKeyIndex： 输入。 会话密钥索引值。

iCheckKeyIndex： 输入。 校验密钥索引值。

pcData： 输入。 要加密的数据。

iDataLen： 输入。 要加密的数据长度[1, 1024]。

pcEncData： 输出。 密文数据。

piEncDataLen： 输入/输出。 输入 pcEncData 的缓冲区大小，输出真实的密文数据长度，长度为 16 的整数倍。

pcMac： 输出。 输出密文校验值（算法为 SM4-MAC），固定长度为 16 字节。

1.15.短数据解密并对密文校验值进行核对

功能：

短数据解密并对输入的密文校验值进行核对

函数原型：

```
int SYD_SM4_Decrypt_Mac_ShortData(  
    void*   phHandle,  
    int     iDecKeyIndex,  
    int     iCheckKeyIndex,  
    char*   pcEncData,  
    int     iEncDataLen,  
    char*   pcData,  
    int*    piDataLen,  
    char*   pcMac);
```

```

void*   phHandle,
int     iEncKeyIndex,
int     iCheckKeyIndex,
char*   pcEncData,
int     iEncDataLen,
char*   pcMac,
char*   pcOutData,
int*    piOutDataLen);

```

返回值：0 表示成功；非 0 表示失败；

参数说明：

phHandle: 输入。 与设备建立的连接句柄。
iEncKeyIndex: 输入。 会话密钥索引值。
iCheckKeyIndex: 输入。 校验密钥索引值。
pcEncData: 输入。 密文数据。
iEncDataLen: 输入。 密文数据长度[1, 1024 + 16]。
pcMac: 输入。 密文校验值，固定长度为 16 字节。
pcOutData: 输出。 明文数据。
piOutDataLen: 输入/输出。 输入 pcOutData 的缓冲区大小，输出真实的明文数据长度，长度为 16 的整数倍。

1.16.批量数据加密并输出密文校验值

功能：

批量数据加密并计算密文的 MAC 值

函数原型：

```

int SYD_SM4_Encrypt_Mac_BatchData(
    void*   phHandle,
    int     iEncKeyIndex[],
    int     iCheckKeyIndex[],
    char*   pcData[],
    int     iDataLen[],
    int     iSize,
    char*   pcEncData[],
    int     iEncDataLen[],
    char*   pcMac[]);

```

返回值：0 表示成功；非 0 表示失败；（有一条数据失败则报错）

参数说明：

phHandle: 输入。 与设备建立的连接句柄。
iEncKeyIndex: 输入。 会话密钥索引值数组。
iCheckKeyIndex: 输入。 校验密钥索引值数组。
pcData: 输入。 要加密数据的数组。
iDataLen: 输入。 要加密数据的数据长度[1, 1024]
iSize: 输入。 表示数组的大小。
pcEncData: 输出。 密文数据数组。

iEncDataLen: 输入/输出。 输入 pcEncData 的缓冲区大小，输出真实的密文数据长度，长度为 16 的整数倍。

pcMac: 输出。 输出密文校验值（算法为 SM4-MAC），固定长度为 16 字节。

1.17.批量数据解密并对密文校验值进行核对

功能：
批量数据解密并对输入的密文校验值进行核对

函数原型：

```
int SYD_SM4_Decrypt_Mac_BatchData(  
    void*   phHandle,  
    int     iEncKeyIndex[],  
    int     iCheckKeyIndex[],  
    char*   pcEncData[],  
    int     iEncDataLen[],  
    char*   pcMac[],  
    int     iSize,  
    char*   pcOutData[],  
    int     iOutDataLen[],  
    int     iCheckRet[]);
```

返回值：0 表示成功；非 0 表示失败；（有一条数据失败则报错）

参数说明：

phHandle: 输入。 与设备建立的连接句柄。

iEncKeyIndex: 输入。 会话密钥索引值数组。

iCheckKeyIndex: 输入。 校验密钥索引值数组。

pcEncData: 输入。 密文数据数组。

iEncDataLen: 输入。 密文数据长度[1, 1024 + 16]。

pcMac: 输入。 密文校验值，固定长度为 16 字节。

iSize: 输入。 表示数组的大小。

pcOutData: 输出。 明文数据数组。

piOutDataLen: 输入/输出。 输入 pcOutData 的缓冲区大小，输出真实的明文数据长度，长度为 16 的整数倍。

iCheckRet: 输出。 iCheckRet 不能为 NULL，pcMac 值作为输入来进行 Mac 值的校验，校验的结果放到 iCheckRet 中，为 0 表示校验成功，非 0 表示校验失败。(这里 iCheckRet 表示的是一个 int 类型的数组)

2、错误码

错误代码 (10 进制)	错误代码 (16 进制)	错误含义
-----------------	-----------------	------

-1	0xFFFFFFFF	网络连接失败（请检查加密机 IP 和端口）
-2	0xFFFFFFFFE	网络发送失败（请检查网络连接状态）
-3	0xFFFFFFFFD	网络接收失败（请检查网络连接状态）
-4	0xFFFFFFFFC	网络参数错误（接口内部错误，请联系厂家）
-5	0xFFFFFFFFB	发送或接收的数据大小超过最大缓冲区大小
-6	0xFFFFFFFFA	网络返回包命令码错误（接口内部错误，请联系厂家）
-7	0xFFFFFFFF9	网络返回包序列号错误（单连接不支持并发调用，请线程间加锁或调整为一个线程一个连接使用）
-8	0xFFFFFFFF8	数据处理超时（请排查网络连接是否正常，如果连接无误，可以将超时调大或重试）
-11	0xFFFFFFFF5	对端关闭连接（加密机连接断开，请重新建立连接）
16777217	0x01000001	参数错误（入参不符合接口条件）
16777218	0x01000002	指针错误（指针为 NULL）
16777219	0x01000003	内存错误（内存不足）
16777220	0x01000004	设备连接数错误
16777221	0x01000005	设备端口错误（将检查端口，交易端口为 8889）
16777222	0x01000006	设备超时设置错误（连接超时或交易超时必须大于 0，单位是 ms）
16777223	0x01000007	无效的句柄（句柄内存无效）
16777224	0x01000008	输出缓冲区大小不足
16777225	0x01000009	队列句柄未初始化（代码内部连接池错误）
16777226	0x0100000A	队列已满（代码内部连接池错误）
16777227	0x0100000B	队列为空（代码内部连接池错误）
16777228	0x0100000C	线程创建失败（代码内部连接池错误）
16777229	0x0100000D	线程池已关闭（代码内部连接池错误）
16777230	0x0100000E	线程句柄无效（代码内部连接池错误）
16777231	0x0100000F	线程销毁超时（代码内部连接池错误）
16777232	0x01000010	MAC 校验错误
16777233	0x01000011	线程数错误（线程数不能为 0 且最大不能超过 16）
16777234	0x01000012	句柄类型错误
16777235	0x01000013	长数据的过程句柄没有执行初始化操作（init, update, final 必须先执行 init 操作）
16777236	0x01000014	解密传入的数据长度错误（不能为 0 且必须满足 16 的倍数）
16777237	0x01000015	批量数据数组大小错误（短连接最多只能支持 4

		个，长连接不限制)
33	0x21	密钥索引不存在
21	0x15	服务端 TCP 报文协议解析错误
68	0x44	解密失败（数据填充错误）