ОАО “КИБЕРПЛАТ”

123610, г.Москва, ЦМТ-2, Краснопресненская наб., д.12 подъезд № 7
Телефон: (495) 967-02-20 Факс: (495) 967-02-08

JSC CYBERPLAT

123610, Moscow, WTC-2, Krasnopresnenskaya nab., 12, entrance # 7
Phone: (495) 967-02-20 Fax: (495) 967-02-08

Руководство программиста по криптографической библиотеке IPRIVPG

1. Используемые типы

1.1. Типы криптосредств

Название	Код	Описание
IPRIV_ENGINE_RSAREF	0	Библиотека RSAREF

В данной версии библиотеки есть только одно криптосредство.

1.2. Типы ключей

IPRIV_KEY_TYPE_RSA_SECRET 1

IPRIV_KEY_TYPE_RSA_PUBLIC 2

1.3. Максимальная длина кода покупателя

MAX_USERID_LENGTH 20

1.4. Структура ключа

```
typedef struct
{
    short eng;                  Тип криптосредства
    short type;                 Тип ключа
    unsigned long keyserial;    Серийный номер ключа
    char userid[24];            Код покупателя (минимум MAX_USERID_LENGTH+1)
    void* key;                  Специфические для криптосредства данные
} IPRIV_KEY;
```

1.5. Функция обратного вызова для загрузки открытого ключа по серийному номеру

typedef int (*Crypt_FindPublicKey_t)(unsigned long keyserial, IPRIV_KEY* key, char* info, int info_len);

Должна возвращать 0 в случае успеха или код ошибки.

2. Интерфейс библиотеки

2.1. Инициализация библиотеки

int Crypt_Initialize(void);

Должна выполняться только один раз при запуске приложения (в основном потоке).

Возвращает: 0 - успех или код ошибки

2.2. Деинициализация библиотеки

int Crypt_Done(void);

Должна выполняться только один раз при завершении приложения (в основном потоке).

Возвращает: 0 - успех или код ошибки

2.3. Загрузка закрытого ключа из буфера в памяти (Crypt_OpenSecretKey), из файла (Crypt_OpenSecretKeyFromFile)

или из внутреннего хранилища криптопровайдера (Crypt_OpenSecretKeyFromStore). Crypt_OpenSecretKey2 грузит ключ из буфера без заголовков (чистое тело ключа в base64).

int Crypt_OpenSecretKey(int eng, const char* src, int nsrc,const char* passwd, IPRIV_KEY* key);

int Crypt_OpenSecretKey2(int eng, const char* src, int nsrc,const char* passwd, IPRIV_KEY* key);

int Crypt_OpenSecretKeyFromFile(int eng, const char* path, const char* passwd, IPRIV_KEY* key);

int Crypt_OpenSecretKeyFromStore(int eng, unsigned long keyserial, IPRIV_KEY* key);

eng: входной, тип криптопровайдера (0)

src: входной, буфер с телом закрытого ключа

nsrc: входной, длина буфера, -1 - считается сама (должен быть нуль-терминатор)

path: входной, путь к файлу с закрытым ключом

passwd: входной, кодовая фраза для расшифровки закрытого ключа

keyserial: входной, серийный номер закрытого ключа

key: выходной, закрытый ключ

Возвращает: 0 - успех или код ошибки

2.4. Загрузка открытого ключа из буфера в памяти (Crypt_OpenPublicKey), из файла (Crypt_OpenPublicKeyFromFile)

или из внутреннего хранилища криптопровайдера (Crypt_OpenPublicKeyFromStore). Crypt_OpenPublicKey2 грузит ключ из буфера без заголовков и подписи (чистое тело ключа в base64).

int Crypt_OpenPublicKey(int eng, const char* src,i nt nsrc, unsigned long keyserial, IPRIV_KEY* key, IPRIV_KEY* cakey);

int Crypt_OpenPublicKey2(int eng, const char* src, int nsrc, IPRIV_KEY* key);

int Crypt_OpenPublicKeyFromFile(int eng, const char* path, unsigned long keyserial, IPRIV_KEY* key, IPRIV_KEY* cakey);

int Crypt_OpenPublicKeyFromStore(int eng, unsigned long keyserial, IPRIV_KEY* key);

eng: входной, тип криптопровайдера (0)

src: входной, буфер с телом открытого ключа

nsrc: входной, длина буфера, -1 - считается сама (должен быть нуль-терминатор)

path: входной, путь к файлу с открытыми ключами

keyserial: входной, серийный номер открытого ключа. Если указать 0, то откроется первый ключ в файле

key: выходной, открытый ключ

cakey: входной, открытый ключ для проверки подписи ключа, может быть 0.

Возвращает: 0 - успех или код ошибки

2.5. Закрытие ключа

int Crypt_CloseKey(IPRIV_KEY* key);

key: входной, открытый или закрытый ключ

Возвращает: 0 - успех или код ошибки

Должна вызываться для всех ключей, которые были открыты функциями в пунктах 2.3 и 2.4

2.6. Экспорт закрытого ключа

int Crypt_ExportSecretKey(char* dst, int ndst, const char* passwd, IPRIV_KEY* key);

int Crypt_ExportSecretKeyToFile(const char* path, const char* passwd, IPRIV_KEY* key);

dst: выходной, буфер для приема закрытого ключа

ndst: входной, максимальная длина приемного буфера

path: входной, путь к файлу для закрытого ключа

passwd: входной, кодовая фраза для шифрования закрытого ключа

key: входной, закрытый ключ

Возвращает: длина тела ключа или код ошибки

2.7. Экспорт открытого ключа

int Crypt_ExportPublicKey(char* dst, int ndst, IPRIV_KEY* key, IPRIV_KEY* cakey);

int Crypt_ExportPublicKeyToFile(const char* path, IPRIV_KEY* key, IPRIV_KEY* cakey);

dst: выходной, буфер для приема открытого ключа

ndst: входной, максимальная длина приемного буфера

path: входной, путь к файлу с открытыми ключами

key: входной, открытый ключ

cakey: входной, закрытый ключ для формирования подписи открытого ключа, может быть 0

Возвращает: длина тела ключа или код ошибки

2.8. Формирование подписи сообщения

int Crypt_Sign(const char* src, int nsrc,char* dst, int ndst, IPRIV_KEY* key);

src: входной, буфер с телом сообщения

nsrc: длина сообщения, -1 - считается сама (должен быть нуль-терминатор)

dst: выходной, буфер для приема тела подписанного сообщения

ndst: входной, максимальная длина приемного буфера

key: входной, закрытый ключ

Возвращает: длина тела сообщения или код ошибки

2.9. Формирование отделенной от сообщения подписи

int Crypt_Sign2(const char* src, int nsrc, char* dst, int ndst, IPRIV_KEY* key);

src: входной, буфер с телом сообщения

nsrc: длина сообщения, -1 - считается сама (должен быть нуль-терминатор)

dst: выходной, буфер для приема тела подписи

ndst: входной, максимальная длина приемного буфера

key: входной, закрытый ключ

Возвращает: длина тела сообщения или код ошибки

2.10. Проверка подписи сообщения

int Crypt_Verify(const char* src, int nsrc, const char** pdst, int* pndst, IPRIV_KEY* key);

src: входной, буфер с телом сообщения

nsrc: длина сообщения, -1 - считается сама (должен быть нуль-терминатор)

pdst: выходной, может быть 0, адрес указателя, в который помещается адрес оригинального сообщения (до подписи)

pndst: выходной, может быть 0, адрес переменной, в которую помещается длина оригинального сообщения (до подписи)

key: входной, открытый ключ

Возвращает: 0 - успех или код ошибки

2.11. Проверка подписи сообщения произвольного формата

int Crypt_Verify2(const char* src, int nsrc, Crypt_FindPublicKey_t find_key, char* info, int info_len, unsigned long* pkeyserial);

int Crypt_Verify3(const char* src, int nsrc, const char* sig, int nsig, IPRIV_KEY* key);

src: входной, буфер с телом сообщения

nsrc: длина сообщения, -1 - считается сама (должен быть нуль-терминатор)

find_key: входной, адрес функции обратного вызова для поиска открытого ключа отправителя

pkeyserial: выходной, если не 0, то сюда вернется серийный номер ключа отправителя

info: выходной, если не 0, то сюда вернется описание ключа

info_len: входной, длина буфера info

Возвращает: 0 - успех или код ошибки

2.12. Проверка подписи сообщения произвольной длины

int Crypt_Verify_Detached(const char* src, int nsrc, const char** pdst, int* pndst, IPRIV_KEY* key);

src: входной, буфер с телом сообщения

nsrc: длина сообщения, -1 - считается сама (должен быть нуль-терминатор)

pdst: выходной, может быть 0, адрес указателя, в который помещается адрес оригинального сообщения (до подписи)

pndst: выходной, может быть 0, адрес переменной, в которую помещается длина оригинального сообщения (до подписи)

key: входной, открытый ключ

Возвращает: 0 - успех или код ошибки

2.13. Шифрование открытым ключом

int Crypt_Encrypt(const char* src, int nsrc, char* dst, int ndst, IPRIV_KEY* key);

Длина сообщения не должна превышать длину ключа.

src: входной, буфер с телом сообщения

nsrc: длина сообщения, -1 - считается сама (должен быть нуль-терминатор)

dst: выходной, буфер для приема зашифрованного сообщения

ndst: входной, максимальная длина приемного буфера

Возвращает: длина зашифрованного сообщения или код ошибки

Данная функция имеет ограничения по использованию и не рекомендуется для прикладных программистов.

2.14. Дешифрование закрытым ключом

int Crypt_Decrypt(const char* src, int nsrc, char* dst, int ndst, IPRIV_KEY* key);

src: входной, буфер с зашифрованным сообщением

nsrc: длина зашифрованного сообщения, -1 - считается сама (должен быть нуль-терминатор)

dst: выходной, буфер для приема сообщения

ndst: входной, максимальная длина приемного буфера

Возвращает: длина сообщения или код ошибки

Данная функция имеет ограничения по использованию и не рекомендуется для прикладных программистов.

3. Коды возвращаемых ошибок

Название ошибки	Код	Описание ошибки
CRYPT_ERR_BAD_ARGS	1	Ошибка в аргументах
CRYPT_ERR_OUT_OF_MEMORY	2	Ошибка выделения памяти
CRYPT_ERR_INVALID_FORMAT	3	Неверный формат документа
CRYPT_ERR_NO_DATA_FOUND	4	Документ прочитан не до конца
CRYPT_ERR_INVALID_PACKET_FORMAT	5	Ошибка во внутренней структуре документа
CRYPT_ERR_UNKNOWN_ALG	6	Неизвестный алгоритм шифрования
CRYPT_ERR_INVALID_KEYLEN	7	Длина ключа не соответствует длине подписи
CRYPT_ERR_INVALID_PASSWD	8	Неверная кодовая фраза закрытого ключа
CRYPT_ERR_DOCTYPE	9	Неверный тип документа
CRYPT_ERR_RADIX_DECODE	10	Ошибка ASCII кодирования документа
CRYPT_ERR_RADIX_ENCODE	11	Ошибка ASCII декодирования документа
CRYPT_ERR_INVALID_ENG	12	Неизвестный тип криптосредства
CRYPT_ERR_ENG_NOT_READY	13	Криптосредство не готово
CRYPT_ERR_NOT_SUPPORT	14	Вызов не поддерживается криптосредством
CRYPT_ERR_FILE_NOT_FOUND	15	Файл не найден
CRYPT_ERR_CANT_READ_FILE	16	Ошибка чтения файла
CRYPT_ERR_INVALID_KEY	17	Ключ не может быть использован
CRYPT_ERR_SEC_ENC	18	Ошибка формирования подписи
CRYPT_ERR_PUB_KEY_NOT_FOUND	19	Открытый ключ с таким серийным номером отсутствует
CRYPT_ERR_VERIFY	20	Подпись не соответствует содержимому документа
CRYPT_ERR_CREATE_FILE	21	Ошибка создания файла
CRYPT_ERR_CANT_WRITE_FILE	22	Ошибка записи в файл
CRYPT_ERR_INVALID_KEYCARD	23	Неверный формат карточки ключа
CRYPT_ERR_GENKEY	24	Ошибка генерации ключей

CRYPT_ERR_PUB_ENC	25	Ошибка шифрования
CRYPT_ERR_SEC_DEC	26	Ошибка дешифрации
CRYPT_ERR_UNKNOWN_SENDER	27	Отправитель не определен

4. Пример использования

```
#include <stdio.h>
#include "libipriv.h"

int main(void)
{
	char temp[1024];
	IPRIV_KEY sec;
	IPRIV_KEY pub;

	Crypt_Initialize();

	int rc = Crypt_OpenSecretKeyFromFile(IPRIV_ENGINE_RSAREF, "secret.key", "1111111111", &sec);
	if (!rc) {
		rc = Crypt_OpenPublicKeyFromFile(IPRIV_ENGINE_RSAREF, "pubkeys.key", 17033, &pub, 0);
		if (!rc) {
			rc = Crypt_Sign("Hello world", -1, temp, sizeof(temp), &sec);
			if (rc > 0) {
				puts(temp);
				rc = Crypt_Verify(temp, rc, 0, 0, &pub);
				if (!rc)
					puts("Verify OK");
				else
					puts("Verify failed");
			} else
				puts("Can't sign message");
			Crypt_CloseKey(&pub);
		} else
			puts("Can't open public key");
		Crypt_CloseKey(&sec);
	} else
		puts("Can't open secret key");

	Crypt_Done();
	return 0;
}
```

В данном примере открываются из файла на диске закрытый и открытый ключ, производится подпись сообщения закрытым ключом и проверка подписи тем же открытым ключом.

В реальной жизни будет загружаться банковский открытый ключ для проверки подписи на входящих сообщениях и свой закрытый ключ для подписи своих запросов.

Этот пример может служить проверкой правильности сборки библиотеки iprivpg из исходного кода.