Тема: Электронная подпись

Содержание

Введение

1 Анализ состояния вопроса

2 Постановка задачи

3 Проектирование программного обеспечения

4 Реализация программного обеспечения

5 Тестирование программного обеспечения

6 Схема алгоритма

Выводы

Список используемых источников

Приложение

Введение

Электро́нная по́дпись (ЭП), Электро́нная цифровая по́дпись (ЭЦП) — реквизит электронного документа, полученный в результате криптографического преобразования информации с использованием закрытого ключа подписи и позволяющий проверить отсутствие искажения информации в электронном документе с момента формирования подписи (целостность), принадлежность подписи владельцу сертификата ключа подписи (авторство), а в случае успешной проверки подтвердить факт подписания электронного документа (неотказуемость).

Электронная подпись предназначена для определения лица, подписавшего электронный документ, и является аналогом собственноручной подписи в случаях, предусмотренных законом. Электронная подпись применяется при совершении гражданско-правовых сделок, оказании государственных и муниципальных услуг, исполнении государственных и муниципальных функций, при совершении иных юридически значимых действий.

Решение поставленных задач отражено в пояснительной записке, которая состоит из восьми разделов и содержит необходимую и достаточную информацию по использованию данного программного средства.

1 Анализ состояния вопроса

В настоящее время существуют множество программных продуктов, использующих проверку целостности документов с помощью цифровой подписи.

Разрабатываемое приложение будет наглядно демонстрировать процесс формирования и проверки ЭЦП по одному из популярных алгоритмов создания ЭЦП - алгоритму RSA.

2 Постановка задачи

Поставленную задачу можно разделить на следующие этапы:

Создание следующих классов:

RSA\_sign, содержащего следующие поля и функций для реализации алгоритма RSA:

* static array<unsigned char>^ MD5hash(array<unsigned char>^ data) – для вычисления MD5 хэша исходных данных
* array<unsigned char>^ CreateSignature(array<unsigned char>^ Text) – для генерации ЭЦП сообщения
* bool VerifySignature(array<unsigned char>^ Text, array<unsigned char>^ Signature, array<unsigned char>^ e, array<unsigned char>^ n) – для проверки цифровой подписи сообщения
* BigInteger^ n – параметр N
* BigInteger^ e - открытый ключ
* BigInteger^ d - закрытый ключ

ConnectHERO - содержащего следующие поля и функций для реализации соединения клиент-сервер между двумя экземплярами приложения:

* String^ ip – ip адрес для соединения в режиме клиента
* int port; - порт для соединения
* bool is\_server – для хранения информации о режиме работы программы
* bool ConnectHERO::ConnectToServer() – для попытки подключения к серверу
* bool ConnectHERO::StartServer() – для работы в режиме сервера
* bool have\_connections – для хранения информации о наличии подключения
* static BackgroundWorker^ bg\_get\_messages –для получения сообщений от собеседника в отдельном потоке
* static BackgroundWorker^ bg\_get\_client – для ожидания подключения собеседника в отдельном потоке
* ConnectHERO::ConnectHERO(System::String^ ip, int port, DataGridView^ output, TextBox^ public\_open\_key\_input, TextBox^ public\_modN\_input) – конструктор класса, для создания экземпляра обьекта
* bg\_get\_messages\_do\_work(Object^ sender, DoWorkEventArgs^ e) – для запуска потока получения сообщений в фоновом режиме
* static void post\_message(String^ username, String^ msg, String^ ecp) – для вывода полученного сообщения в окно приложения
* void ConnectHERO::Publish\_RSA\_data(RSA\_public\_data^ data) – для отправки данных RSA собеседнику
* bool SendSignedMessage(String^ text, String^ ECP) – для отправки сообщения с цифровой подписью
* static void PrintPublicRSAonForm(RSA\_public\_data^ d) – для вывода в окно приложения полученных от собеседника данных RSA
* close\_connections() – для закрытия открытых в приложении подключений

3 Проектирование программного обеспечения

После запуска приложения, пользователю должны быть доступны следующие элементы для работы с приложением:

button\_start – для запуска сервера / подключения к серверу

radioButton\_server – для выбора режима сервера

radioButton\_client – для выбора режима клиента

textBox\_ip – для указания IP адреса для подключения

txtBox\_crypto\_n – для крипто-параметра N

txtBox\_crypto\_e – для крипто-параметра E (открытый ключ)

txtBox\_crypto\_d – для крипто-параметра D (закрытый ключ)

btnGen\_A – для генерации крипто-параметров

btn\_Public\_RSA\_data – для публикации крипто-параметров

txtBox\_crypto\_Public\_n – для крипто-параметра N собеседника

txtBox\_crypto\_Public\_e – для крипто-параметра E собеседника

dataGridView\_output – для вывода сообщений

textBox\_input – для ввода сообщения

txtBoxSIGN\_A – для цифровой подписи сообщения

checkBox\_calc\_ECP – для указания вычислять ли ЭЦП для сообщения

button\_send\_message – для отправки сообщения

4 Реализация программного обеспечения

За соответствующими элементами управления закреплены следующие обработчики событий:

//при загрузке приложение пытается подключиться к серверу с настройками по умолчанию

//если подключиться к серверу не удалось, приложение запускается в режиме сервера и ожидает подключение собеседника

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

ConnectHERO^ connectHERO = gcnew ConnectHERO("127.0.0.1", 3005, dataGridView\_output, txtBox\_crypto\_Public\_n, txtBox\_crypto\_Public\_e);

if (connectHERO->is\_server){

connectHERO->post\_message("Вы", "Не удалось подключиться к серверу", "");

radioButton\_server->Checked = true;

if (connectHERO->StartServer())

connectHERO->post\_message("Вы", "Запущен сервер на " + connectHERO->ip, "");

}

else

{

radioButton\_server->Checked = false;

connectHERO->post\_message("Вы", String::Format("Соединение с {0}:{1} установлено!", connectHERO->ip, connectHERO->port), "");

}

enc = Encoding::Default;

}

//при вводе текста сгенерировать новую ЭЦП

private: System::Void txtBoxSIGN\_A\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

if (!checkBox\_calc\_ECP->Checked) return;

if (!CheckCryptoInputs())

{

txtBoxSIGN\_A->Text = "Сгенерируйте криптографические параметры!";

return;

}

if (!CheckMessageInput())

{

txtBoxSIGN\_A->Text = "Введите текст сообщения!";

return;

}

array<unsigned char>^ bText = enc->GetBytes(textBox\_input->Text);

txtBoxSIGN\_A->Text = BigInteger(rsa\_sign->CreateSignature(bText)).ToString();

}

//подключиться к серверу или запустить сервер

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

if (connectHERO == nullptr || connectHERO->have\_connections == false)

{

if (radioButton\_server ->Checked)

{

connectHERO->StartServer();

}

if (radioButton\_client->Checked)

{

connectHERO->ConnectToServer();

}

}

else

{

MessageBox::Show("Подклюяение с собеседником уже установлено!");

}

}

// сгенерировать крипотпараметры

private: System::Void btnGen\_A\_Click(System::Object^ sender, System::EventArgs^ e) {

rsa\_sign = gcnew RSA\_sign();

txtBox\_crypto\_e->Text = rsa\_sign->E();

txtBox\_crypto\_d->Text = rsa\_sign->D();

txtBox\_crypto\_n->Text = rsa\_sign->N();

}

//отправить сообщение

private: System::Void button1\_Click\_1(System::Object^ sender, System::EventArgs^ e) {

MessageToChat();

}

//контекстное мею - кнопка проверить ЭЦП

private: System::Void проверитьЭЦПToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e) {

int i;

try

{

i = dataGridView\_output->SelectedRows[0]->Index;

}

catch (Exception^ ex)

{

MessageBox::Show("Выберите сообщение!");

return;

}

if (dataGridView\_output->Rows[i]->Cells[2]->Value->ToString() == "")

{

MessageBox::Show("Сообщение было отправлено без ЭЦП");

return;

}

if (txtBox\_crypto\_Public\_n->Text == "" || txtBox\_crypto\_Public\_e->Text == "")

{

MessageBox::Show("Не опубликованы криптографические параметры Абонента В");

return;

}

if (dataGridView\_output->Rows[i]->Cells[1]->Value->ToString() == "")

{

MessageBox::Show("Сообщение отсутствует");

return;

}

String^ sign = dataGridView\_output->Rows[i]->Cells[2]->Value->ToString();

try

{

BigInteger^ S = gcnew BigInteger(sign, 10);

BigInteger^ E = gcnew BigInteger(txtBox\_crypto\_Public\_e->Text, 10);

BigInteger^ n = gcnew BigInteger(txtBox\_crypto\_Public\_n->Text, 10);

String^ msg = dataGridView\_output->Rows[i]->Cells[1]->Value->ToString();

if (rsa\_sign->VerifySignature(enc->GetBytes(msg), S->getBytes(), E->getBytes(),

n->getBytes()))

{

MessageBox::Show("Цифровая подпись верна");

return;

}

}

catch (Exception^ e)

{

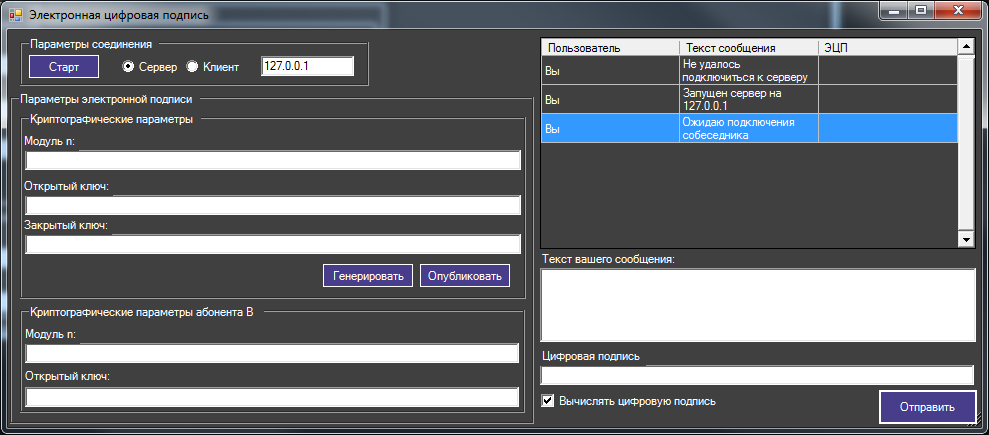
MessageBox::Show("Цифровая подпись не верна\n"+ e->Message);

}

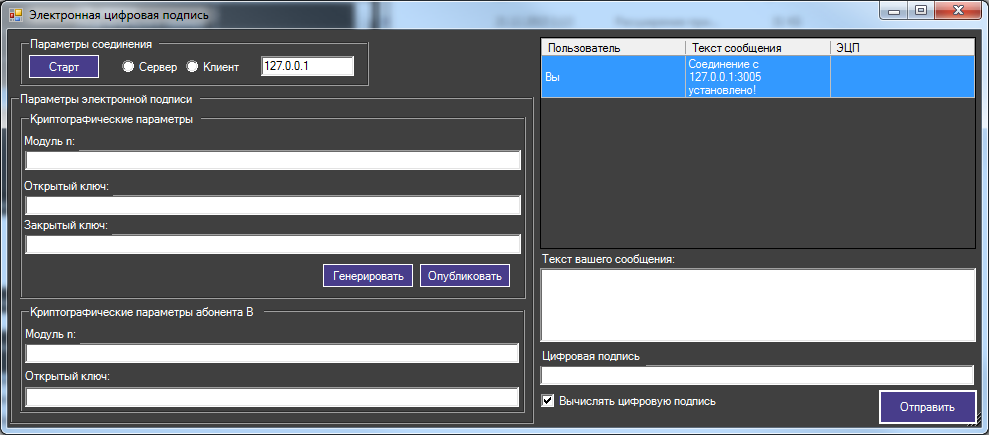
}

5 Тестирование программного обеспечения

После запуска программы на экране появится окно, в котором доступны все функции приложения. Программа произведет попытку подключения к серверу, если произойдет ошибка подключения – приложении будет запущено в режиме сервера, в элементе для вывода информации появятся соответствующие сообщения:



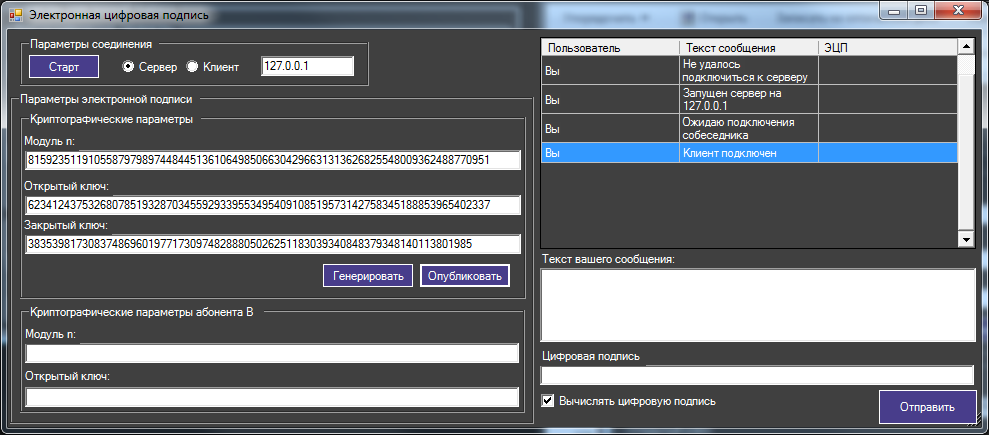
Запустим второй экземпляр приложения:



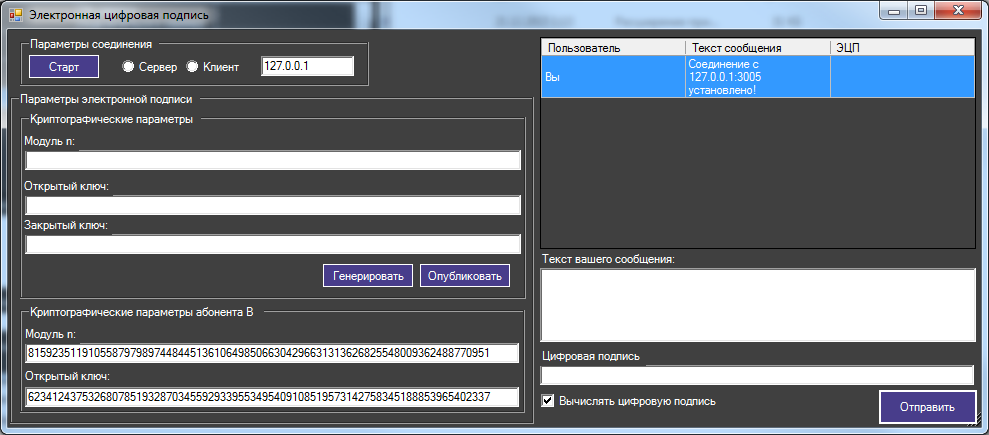
Второй экземпляр успешно подключился к первому, на экран выведено соответствующее сообщение.

Для создания ЭЦП необходимо сгенерировать криптографические параметры и обменяться ими. Нажмем на кнопку «Генерировать», затем на кнопку «Опубликовать»

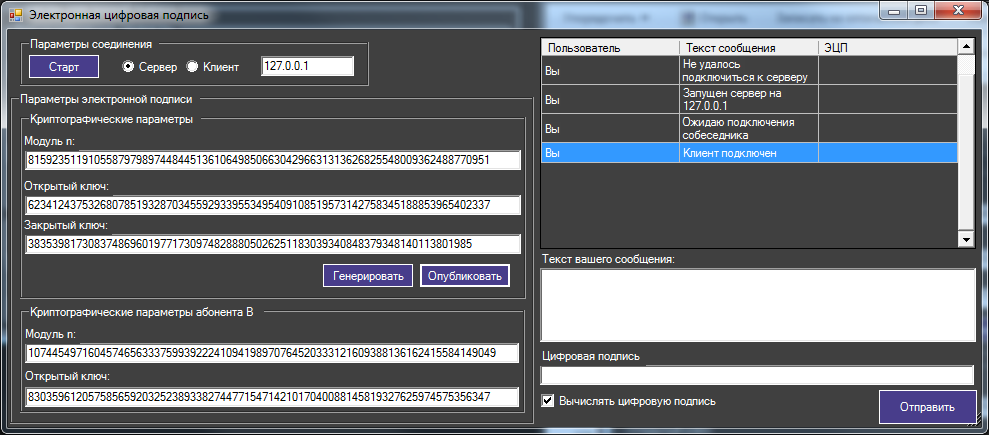
Экземпляр приложения 1



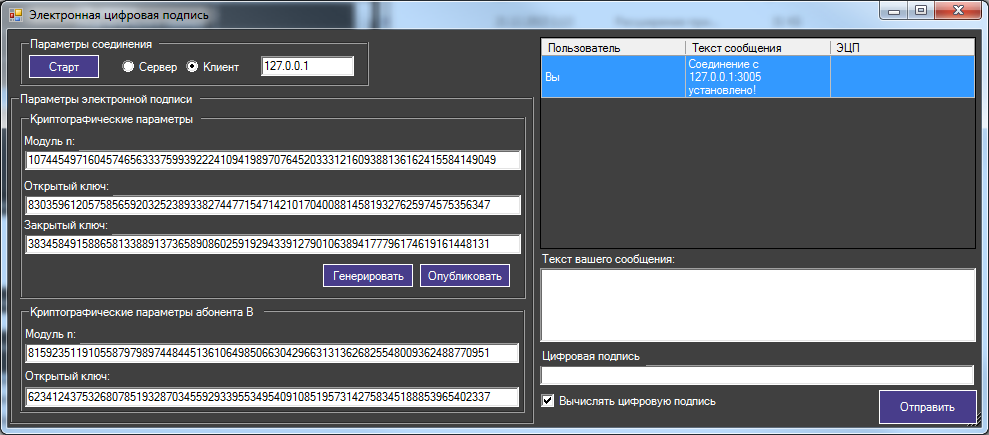
Экземпляр приложения 2



Затем сделаем то же самое во втором экземпляре:

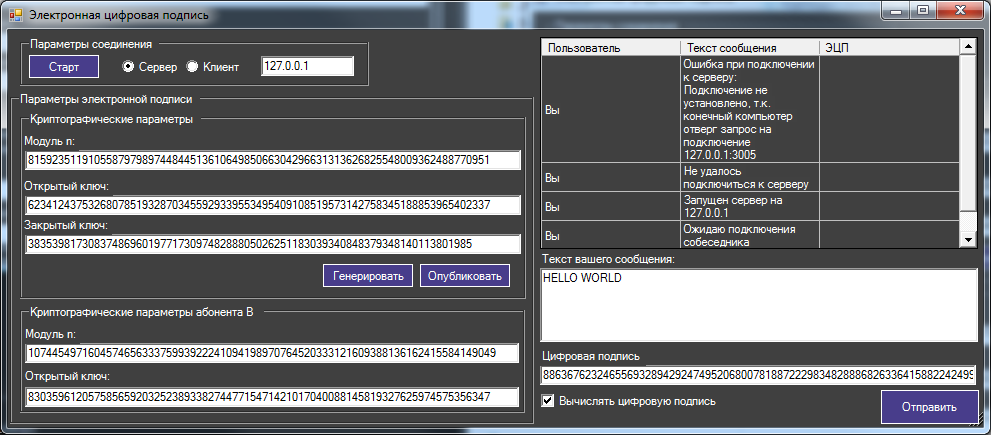
Экземпляр приложения 1 

Экземпляр приложения 2



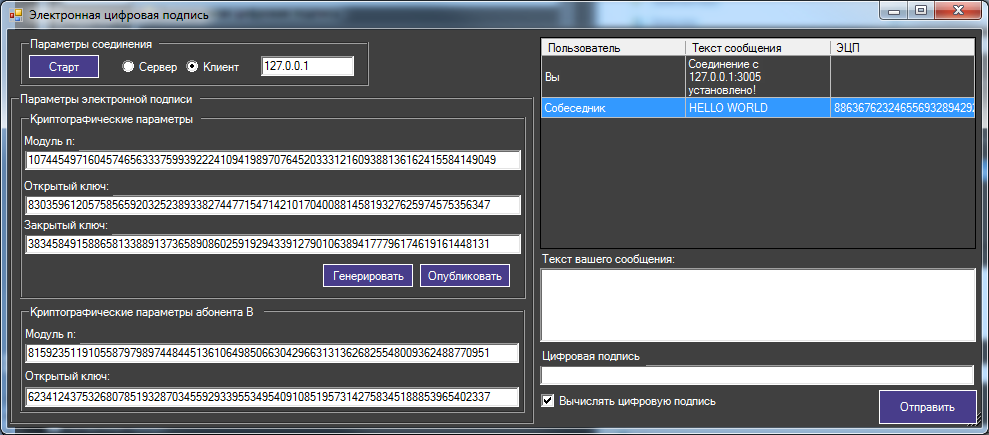
После генерации ключей и обмена ими можно приступать к обмену личными сообщениями между пользователями.

Введем в поле ввода сообщение и нажмем на кнопку отправить:



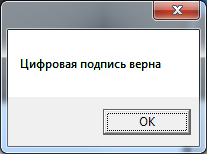
Под сообщением появится сгенерированная ЭЦП, а в поле вывода введенное сообщение.

В окне второго приложения появится текст сообщения, а в соседнем столбце значение ЭЦП:



Нажмем правой кнопкой мыши на появившемся сообщении и выберем «Проверить ЭЦП»:

На экране появится сообщение



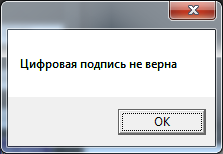
Попробуем изменить в таблице значение ЭЦП на один символ, после чего снова проверить ЭЦП выбранного сообщения:

57783592165773142203423942015133496805121998405591674276744600075356612450349

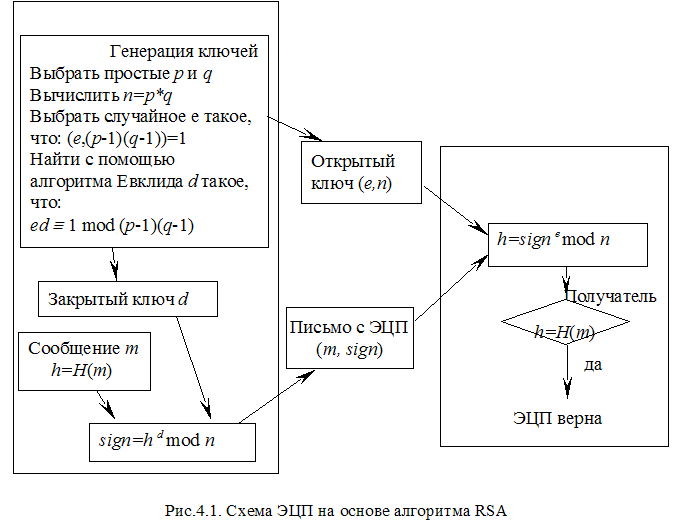
=>

57783592165773142203423942015133496805121998405591674276744600075356612450340

На экране появится следующее сообщение:



6 Схема алгоритма ЭЦП RSA



Список используемых источников

Описание алгоритма: <https://ru.wikipedia.org/wiki/RSASSA-PSS>

Электронная цифровая подпись для чайников: <http://habrahabr.ru/post/97066/>

Герберт Шилдт «C++. Базовый курс» 2014

Методы и средства защиты компьютерной информации <http://www.volpi.ru/umkd/zki/index.php?man=1&page=28>