МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра математического моделирования и анализа данных

БЖЕЗИНСКИЙ ТАРАС АЛЕКСАНДРОВИЧ

ВСТРАИВАНИЕ КРИПТОГРАФИЧЕСКИХ МЕХАНИЗМОВ В PDF-ФАЙЛЫ

Дипломная работа
студента 5 курса 9 группы

"Допустить к защите"
с предварительной оценкой ____

Руководитель работы

_____
«__»_____ 2013 г

Руководитель
Агиевич Сергей Валерьевич
заведующий НИЛ ПБИТ НИИ ППМИ,
кандидат физ.-мат. наук

"Допустить к защите"
с предварительной оценкой ____

Рецензент работы

_____
«__»_____ 2013 г

Рецензент
Мельникова Елена Николаевна
С.н.с. НИЛ СТАМ НИИ ППМИ,
кандидат физ.-мат. наук

Минск 2013

АННОТАЦИЯ

В настоящей дипломной работе рассматриваются вопросы перехода на электронный документооборот и единый формат электронного документа; изучаются наиболее необходимые криптографические методы, обеспечивающие защиту данных в электронном документе; а также предложены попытки внедрения криптографических стандартов СТБ в электронный документ. В качестве исследуемых криптографических методов выступают электронно-цифровая подпись (ЭЦП) и широковещательное шифрование (BE).

ANNOTATION

This diploma thesis deals with questions of transition to electronic document and common format for electronic document. The most necessary methods of electronic document protection are examined, also considering the introduction of cryptographic standards STB for protection as well. The attention is paid to Digital Signature and Broadcast Encryption methods.

Содержание

1 Введение

В настоящее время начался процесс перехода на электронный документооборот. Подтверждением этому могут служить многочисленные законодательные акты, в том числе и Закон Республики Беларусь №113-З "Об электронном документе и электронной цифровой подписи"от 28 декабря 2009 года.

В связи с этим возникают вопросы об унификации формата электронного документа, его защите и адаптации данного формата под существующие национальные стандарты. Приняв решение данных вопросов за некую базу, становится возможным построение аппарата дополнения и расширения данной базы, а также адаптация новых и нестандартных методов защиты и передачи электронных документов.

Данная работа ставила перед собой следующие задачи:

1. Поиск формата электронного документа, который бы отвечал следующим требованиям:
   - Кроссплатформенность
   - Поддержка различных типов внутреннего содержимого, таких как:
     (a) Обычный текст (plaintext)
     (b) Мультимедийное содержание
     (c) Интерактивное содержание
   - Наличие/разработка стандарта формата
   - Наличие возможностей встраивания криптозащиты, а именно ЭЦП и BE.
2. Исследование данного формата с точки зрения возможностей криптографического содержимого
3. Нахождение способов встраивания криптозащиты в документы и их сравнительная характеристика
4. Выбор одного из способов с обоснованием, а также попытка внедрения СТБ цифровой подписи и широковещательного шифрования.

В работе предприняты попытки решения данных задач. В качестве формата электронного документа был принят PDF, по следующим причинам:

- Наличие стандарта
- Богатый набор средств хранения содержимого
- Постоянно развивающиеся и поддерживаемые компанией-разработчиком средства просмотра и редактирования, а также компоненты для программной работы с форматом.

Таким образом, в последующих разделах рассматриваются следующие вопросы:

1. Формат PDF. Под этим будем понимать внутреннюю структуру документа, описание объектов PDF, наличие стандартов.
2. Обзор необходимых средств криптозащиты.

3. Встраивание криптографических алгоритмов в документы формата PDF. Под этим будем понимать:

    - Исследование возможности встраивания алгоритмов, различные методы достижения цели, а также их сравнительную характеристику
    - Разработка архитектуры и непосредственная реализация встраивания.

2 Формат PDF

На сегодняшний день единственным форматом, который отображается одинаково на всех устройствах, работающих под управлением всех без исключения операционных систем является `PDF - Portable Document Format`, разработанный компанией `Adobe Systems`. PDF является кроссплатформенным и использует ряд возможностей языка `PostScript`, также является открытым стандартом ISO 32000 с 01.07.2008. На практике чаще всего PDF-файл является комбинацией текста с растровой и векторной графикой, однако возможны также комбинации текста и форм, JavaScript, 3D-графикой.

2.1 История PDF

История формата берет свое начало в 1993 году, когда была выпущена первая версия документа PDF. Первое время формат поддерживал мало возможностей, имел больший размер по сравнению с обычными текстовыми файлами, а также поддерживался лишь платным программным обеспечением, что не снискало ему популярности. Однако, с ходом времени, были улучшены алгоритмы сжатия размеров файла, появились бесплатные программы-просмотрщики, добавились возможности, в том числе и поддержка гиперссылок, и формат стал завоевывать популярность. В настоящее время он является наиболее широко распространенным форматом для файлов справок, технической документации, научных публикаций и многого другого. Ниже приведена таблица, где в хронологическом порядке представлены версии формата и их отличительные особенности. Версия pdf-файла задается его первыми 8 байтами.

версия	издание	год	новые возможности	версия ПО
1.0	1-ое	1993		Carousel
1.1	1-ое (переиздано)	1996	Пароли, криптография (MD5, RC4 40 бит), аппаратно-независимые цвета, темы и ссылки	2.0
1.2	1-ое (переиздано)	1996	Интерактивные элементы страницы (радиокнопки, флажки), интерактивные формы (AcroForm); Forms Data Format (FDF) для интерактивной формы данных, которые можно импортировать, экспортировать, передавать и получить из Интернета, события мыши, внешнее воспроизведение видео, внешнее или встроенное воспроизведение аудио; Zlib//deflate сжатие текстовых или двоичных данных; Unicode; расширенные возможности цвета и изображения	3.0
1.3	2-ое	2000	Цифровые подписи; пространства цветов ICC и DeviceN; Javascript; встраивание файловых потоков любого типа; новые возможности PostScript 3; Web Caputre; поддержка CIDFonts; структуры данных для построения отображений чисел и строк в объекты PDF; параметризированные функции	4.0

1.4	3-е	2001	JBGI2; прозрачность; поддержка 128-битного шифрования RC4; XML формы; встроенные PDF4; XMP; OCR; функциональность для различных групп пользователей; поддержка меток принтера; CMaps; импорт содержимого между PDF документами; стандартизировано хранение встроены данных - EmbeddedFiles	5.0
1.5	4-е	2003	JPEG 2000; расширена поддержка встраивания и воспроизведения мультимедийных данных; потоки объектов; кроссссылочные потоки; XML Forms Data Format для интерактивных форм; поддержка криптографии с открытым ключом на основе PKCS#7; подписи уровней доступа; поддержка RSA и SHA-1 длиной до 4096 бит; поддержка внешних алгоиртмов шифрования; слои; многоязычные документы	6.0
1.6	5-е	2004	3D содержимое, в том числе поддержка Universal 3D; внедрение шрифтов OpenType; поддержка XFA 2.2 rich text; AES; расширение SHA256, DSA до 4096 бит (используется PKCS#7); пространство цветов NCHannel; дополнительная поддержка для внедренных данных, включая кросс-документные ссылки к и от внедренным файлам; улучшения к цифровых подписям, связанные с правами пользователя и дополнительной защитой от изменения подписи	7.0
1.7 (ISO 32000-1:2008)	6-е	2006	расширен функционал, связанный с 3D содержимым;XFA 2.4 rich text; портируемые коллекци вложенных файлов; новые типы строк: PDFDocEncoded, ASCII, byte; поддержка SHA384, SHA512, RIPEMD160 на основе PKCS#7	8
1.7 Extension Level 3		2008	256-битное шифрование AES; включение XFA Datasets в PDF/A-2; улучшены вложения Flash-содержимого (к примеру, Flash-видео с H.264); двухстороннее скрипт-API для собственных и Flash приложений; XFA 2.5 & 2.6	9
1.7 Extension Level 5		2009	XFA 3.0	9.1

1.7 Extension Level 8		2011	Улучшение реализации криптографических алгоритмов.	X (10)

Таблица 1: Версии PDF в хронологическом порядке

2.2 Внутренняя структура PDF

Структура PDF формата представлена в виде схемы на Рисунке 1.

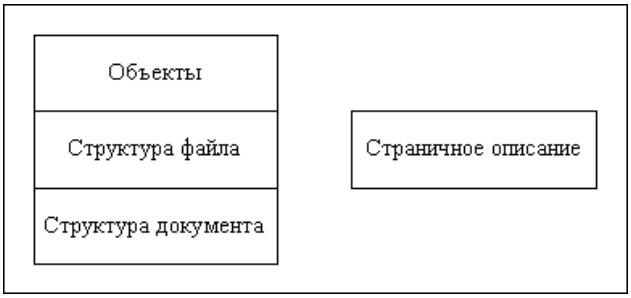

Рис. 1: Обобщенная структура формата PDF.

2.2.1 Объекты

Формат PDF поддерживает несколько основных типов объектов, к примеру:

- `boolean` – логические:

  `true, false`

- `number` – числовые:

  `integer, real`

- `string` – последовательность символов в круглых скобках

- `array` – последовательность PDF объектов различных типов

- `dictionary` – таблицы, состоящие из двух элементов: ключа и значения (используются для соединения атрибутов сложных объектов, например, в PDF–публикациях с помощью этих объектов представляются страницы и шрифты)

- `stream` – в виде объекта типа stream представлены большие объемы данных, такие как изображения и описания страниц.

- `null` - нулевой объект.

Косвенные объекты могут быть помечены так, чтобы на них ссылались другие объекты. Это используется, например, для создания объекта, значение которого изначально не известно. Любой тип объекта может быть помечен как косвенный. Такой объект содержит идентификатор объекта, который сохраняется даже при изменении самого объекта.

Любой объект, использующийся как элемент массива или значение dictionary, может быть определен, либо как обычный объект, либо как косвенная ссылка.

Определение 1. *Косвенная ссылка – это ссылка на косвенный объект, содержащая номер косвенного объекта (идентификатор), и ключевое слово R.*

2.2.2 Структура файла

Структуру файла формата PDF можно представить в виде схемы, приведенной на рисунке 2. Эта структура обеспечивает быстрый доступ к любой части файла и содержит механизм для его изменения.

Рис. 2: Структура файла формата PDF.

Как правило, PDF-файл содержит четыре раздела:

1. Заголовок (`header`)
2. Тело файла (`body`)
3. Таблица перекрестных ссылок (`cross-reference table`)
4. Trailer

Заголовок Первая строка PDF файла определяет номер спецификации PDF, которой придерживается данный файл (к примеру, `1.2 %PDF-1.2`).

```
<header> ::= <PDF version>
```

Тело файла Содержит последовательность косвенных объектов, входящих в состав публикации. Объекты – это компоненты публикации, такие как:

- страницы
- изображения
- шрифты.

Комментарии могут быть во всем PDF файле. Их синтаксис совпадает с синтаксисом комментариев в `PostScript`, начинаются с % и заканчиваются концом строки.

Таблица перекрестных ссылок Таблица содержит информацию о каждом объекте в файле в виде одной строки описания места объекта в файле. PDF файл содержит таблицу, состоящую из одного или более разделов. Если нет изменений, или добавлений в файл, таблица состоит из одного раздела. В противном случае, в ее состав добавляется новый раздел.

Каждый подраздел содержит данные для непрерывной области номеров объектов. Каждый подраздел перекрестной ссылки начинается со строки заголовка, содержащей два числа: объектный номер в этом подразделе и номере данных в подразделе.

Есть два формата данных таблицы: для объектов, которые используются и для объектов, которые были удалены.

```
<cross-reference entry> ::=

<in-use entry> | <free entry>
```

Для объекта, который используется в публикации, содержится байтовое смещение, определяющее количество байтов от начала файла к началу объекта, номер генерации объекта, и ключевое слово n:

```
<in-use entry>::= <byte offset> <generation number> n
```

Для объекта, который свободен, содержится объектный номер следующего свободного объекта, номер генерации, и ключевое слово f:

```
<free entry>::= <object number of next free object> <generation number> f.
```

Когда косвенный объект удаляется, его запись в таблице помечается как «свободная», и номер генерации объекта, увеличивается на единицу, чтобы открыть возможность использования объекта с таким номером.

Trailer Trailer позволяет программному приложению, при чтении PDF файла, быстро находить таблицу перекрестных ссылок и специальные объекты. Приложение должны читать PDF файл с конца. Последняя строка содержит маркер конца файла `%%EOF`. Две предшествующие строки содержат ключевое слово – `startxref` и байт смещения от начала файла к началу слова `xref` в последнем разделе таблицы ссылок в файле. `Trailer dictionary` предшествует этой строке.

2.2.3 Иерархия объектов файла PDF

.

PDF-документ может быть описана как иерархия объектов, содержащихся в теле PDF файла. Структура PDF документа представлена на рисунке 3.

Основными объектами в этой иерархии являются таблицы `dictionary`. Связи в иерархии представлены парами ключ → значение, в которых значение – косвенная ссылка на родительский или дочерний объект. Например, объект Catalog, который является «корнем» иерархического дерева, содержит «ключ страниц», и соответствующее ему значение – косвенная ссылка на объект корень «дерева» страниц (Pages tree).

Каждая страница документа включает ссылки к своим изображениям, миниатюрам и комментариям, которые появляются на странице. Trailer PDF файла, описанный выше, определяет место объекта Catalog, в виде значения корневого ключа (Root) в Trailer. Кроме того, Trailer задает с помощью ключа Info место информационной таблице dictionary публикации, т.е. структуре, которая содержит общие сведения о документе.

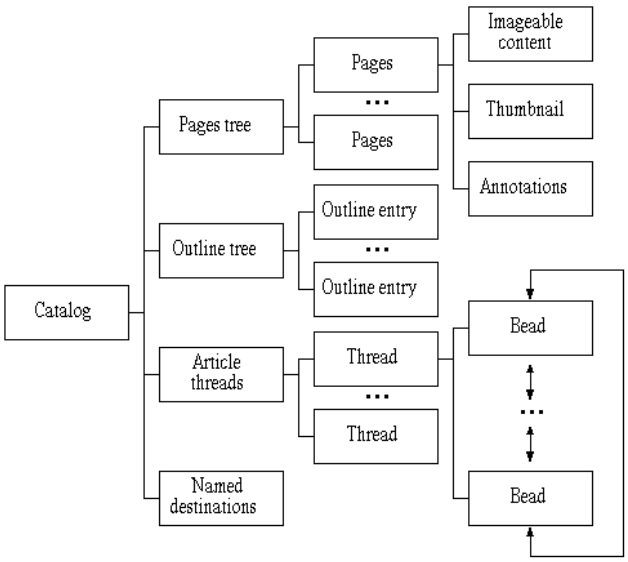

Рис. 3: Иерархия объектов в формате PDF.

Catalog

Определение 2. *Catalog – объект типа dictionary, являющийся корневым узлом документа, содержит ссылки на «дерево» страниц в документе, ссылку на дерево объектов, представляющих схему документа (bookmarks, или outline), ссылки на статьи и список named destinations.*

Catalog показывает также, появляется ли схема публикации или миниатюры автоматически, когда документ просматривается (задается атрибутом типа имя со значениями: UseNone, UseOutlines, UseThumbs, FullScreen), и должна ли быть показана при открытии иная, чем первая страница. С помощью этого объекта атрибутом ViewerPreferences можно задать также параметры программы просмотра при открытии публикации.

Пример объекта Catalog:

```
1 0 obj
<<

/Type /Catalog

/Pages 2 0 R

/Outlines 3 0 R

/PageMode /UseOutlines

>>

endobj
```

Дерево страниц (Pages tree) Доступ к страницам документа открывается через дерево узлов, названное – деревом страниц. Это дерево определяет порядок страниц в документе. Для оптимизации производительности программы просмотра, Acrobat Distiller

и Acrobat PDF Writer конструируют сбалансированное дерево. Структура дерева позволяет приложению быстро открыть документ, содержащий тысячи страниц используя ограниченный объем памяти. Простейшая структура состоит из единственного узла страниц, который ссылается на все страничные объекты. Структура дерева страниц документа не связана с содержимым документа. В PDF файле книги, например, не гарантируется, что глава представлена одним узлов в дереве. Корень и все другие узлы дерева страниц являются объектами типа dictionary. Их основными атрибутами являются: имя pages, список косвенных ссылок к непосредственным дочерним узлам, объект предок типа dictionary.

Пример:

```
2 0 obj

<<

/Type /Pages

/Kids [4 0 R 10 0 R 24 0 R]

/Count 3

>>

endobj
```

Объекты страницы (pages) Эти объекты являются объектами типа dictionary, ключи которого описывают текст, содержащийся на одной странице и изображения. Основные атрибуты:

1. имя – Page
2. MediaBox Rectangle – определяет «настоящий размер» страницы
3. Crop box – размер для печати
4. Parent – объект, непосредственный предок страницы
5. Resources – типа dictionary (ресурсы, требующиеся этой странице)
6. Contents – типа stream (определяет страничное описание посредством косвенной ссылки)
7. Thumb – типа stream (содержит ссылку на миниатюру)
8. Annots – типа array (содержит массив объектов, который определяет комментарии на страницу)
9. B – array (если страница содержит части статей)
10. H – boolean (true - страница скрыта во время показа документа)

Пример иллюстрирует страницу с миниатюрой и двумя комментариями.

```
3 0 obj

<<

/Type /Page

/Parent 4 0 R

/MediaBox [0 612 792]

/Resources <<

/Font << /F3 7 0 R /F5 9 0 R /F7 11 0 R >>

/Process [/PDF] >>

/Thumb 12 0 R

/Contents 14 0 R

/Annots [23 0 R 24 0 R]

>>

endobj
```

Миниатюры (Thumbnail) PDF документ может включать миниатюрные схемы страниц. Миниатюра задается значением ключа Thumb объекта страницы. Структура миниатюры подобна, за небольшими исключениями, структуре изображения.

Комментарии (Annotations) Комментарии – это заметки или другие объекты, которые связаны со страницей, но описываются отдельно от дескриптора страницы. PDF поддерживает несколько видов комментариев: текстовые; гипертекстовые связи; видео и аудиоинформацию.

Если страница содержит комментарии, они сохраняются в массиве как значение Annots ключа объекта страница. Каждый комментарий – объект типа dictionary. Основными ключами комментария являются: Type, Subtype, Rect.

Дерево закладок – Outline tree (bookmarks tree) Структура предоставляет пользователю возможность иметь доступ к различным видам публикации по имени. Активация outline entry (называемые также bookmark) «переносит» на новый вид, заданный в так называемом «описании места назначения» (destination description) для bookmark. Закладки часто образуют иерархическую структуру. Если документ включает закладки, они доступны по ключу Outlines в Catalog-объекте. Значение этого ключа – корень дерева закладок. Закладка верхнего уровня содержит связный список. В процессе просмотра закладки появляются в той последовательности, в которой они входят в данный список. Основные атрибуты этого объекта типа dictionary: Count (общее количество открытых закладок), First (ссылка на закладку-начало списка), Last (ссыл-

ка на конец списка),Title – название, Dest типа array или name – место назначения (Destination), A – действие, выполняющиеся при активации закладки, Parent – ссылка на закладку верхнего уровня иерархии, Prev – ссылка на предыдущую закладку, Next – ссылка на следующую закладку.

Пример:

```
22 0 obj

<<

/Parent 21 0 R

/Dest [3 0 R /Top 0 792 0]

/Title (Document)

/Next 29 0 R

/First 25 0 R

/Last 28 0 R

/Count 4

>>

endobj
```

Место назначения (Destinations) Комментарии и закладки могут определять место назначения, которое состоит из страницы, места на странице, и масштаба показа страницы. Назначение может быть представлено явно как массив или посредством имени. В первом случае значениями ключа Dest являются непосредственно данные о странице – Page, Top, Bottom, Left, Right, Zoom, в различных комбинациях задающие страницу и показанный «прямоугольник» на ней. Поименованные места назначения (тип string или name) часто применяются, когда закладка ссылается на другой файл. Catalog документа может содержать ключ Names со значениями, каждое из которых представляет собой дерево, подобное дереву страниц. Листья – содержат пары из strings и косвенных объектов, которые и являются destinations.

Дерево имен (Name tree) Дерево имен похоже на дерево страниц, но листья содержат пары string (имен) и объектов. Такое дерево применяется для организации поименованных мест назначения. Оно состоит из узлов трех видов: корень, промежуточное, листья. Корень содержит атрибут Kids – массив и Limits – массив. Лист содержит Limits и массив Names (форма массива name `valuenamevalue`..., где value – косвенная ссылка на объект). Имена в дереве сохраняются только в листьях.

Информационный объект типа dictionary (Info dictionary) Как упоминалось выше, trailer документа может содержать ссылки на Info dictionary, который содержит

информацию о публикации. Значения строковых атрибутов этого объекта представляют собой информационное окно о документе в Acrobat. В качестве атрибутов используются: Author, CreationDate, ModDate, Creator, Title, Subject, Keywords.

Статьи (article threads) Публикация может включать несколько статей (article thread), каждая из которых, в свою очередь, может содержать несколько фрагментов – bead. Статьи (threads) сохраняются в массиве как значение ключа Threads в Catalog-объекте. Каждая статья и ее фрагменты представляют собой объекты-dictionary. Атрибуты статьи:

- F (определяет первый фрагмент)
- I (содержит информацию о статье, подобен Info dictionary)

К атрибутам фрагмента относят:

- T (ссылка на статью)
- V (ссылка на предыдущий фрагмент)
- N (следующий фрагмент)
- P (страница, на которой размещается фрагмент)
- R (прямоугольник положения фрагмента на странице)

Криптографические объекты В иерархию объектов включаются также объекты, которые не отвечают непосредственно за хранение и организацию данных, а несут служебное назначение. Среди них нас наиболее интересуют объекты, связанные с защитой информации. PDF-файл может содержать объекты, отвечающие за подпись документа и его шифрование. Более подробно про данные объекты будет рассказано в следующих разделах.

3 Обзор необходимых средств криптозащиты

Как и в любом случае, связанном с хранением и изменением данных, работа с PDF файлами включает в себя вопрос информационной безопасности (ИБ). Классическое определение информационной безопасности говорит о том, что это процесс обеспечения:

1. Конфиденциальности

   Определение 3. *Конфиденциальность - обеспечение доступа к информации только авторизорованным пользователям*

2. Целостности

   Определение 4. *Целостность - обеспечение достоверности и полноты информации (и методов ее обработки)*

3. Доступности

   Определение 5. *Доступность - обеспечение доступа к информации и связанным с ней активам авторизованных пользователей по мере необходимости.*

В данной работе нами выдвигаются дополнительные требования к ИБ:

1. За достоверность и полноту информации отвечает ее владелец, который должен будет подтвердить свое право на владение информацией и свои гарантии ее подлинности

2. Доступность и конфиденциальность должны обеспечиваться таким образом, чтобы для любого конечного множества $\mathtt{A}$ пользователей было верно следующее: существует возможность передачи информации любому подмножеству пользователей $\mathtt{B}$ так, чтобы никто из множества $\mathtt{A} \setminus \mathtt{B}$ не мог получить доступ к информации.

Существует несколько возможных решений, обеспечивающих выполнение данных требований, однако наиболее действенными с точки зрения баланса быстродействия/удобства использования и реализации/требуемых ресурсов криптографическими решениями на данный момент являются:

- Электронно-цифровая подпись для целостности
- Широковещательное шифрование для конфиденциальности и доступности.

3.1 Электронно-цифровая подпись

Определение 6. *ЭЦП – последовательность символов, являющаяся реквизитом электронного документа и предназначенная для подтверждения целостности и подлинности электронного документа. Средство электронной цифровой подписи – программное, программно-аппаратное или техническое средство, реализующее одну или несколько следующих функций: выработку электронной цифровой подписи, проверку электронной цифровой подписи, создание личного ключа подписи или открытого ключа.*

3.1.1 Общие понятия об ЭЦП

Для каждой ЭЦП характерны следующие алгоритмы:

- Алгоритм $\mathcal{GEN}$ - генерация параметров ТДС (третья доверенная сторона, Key Authentication center). В данном алгоритме происходит выработка общих параметров, инициализация алгоритма выдачи ключей.
- Алгоритм $\mathcal{SIGN}$ - выработка ЭЦП. Данный алгоритм принимает на вход сообщение M, личный ключ пользователя e и на их основе, используя общие параметры, предоставляемые ТДС, вырабатывает ЭЦП S.
- Алгоритм $\mathcal{VERIFY}$ - проверка ЭЦП. Данный алгоритм принимает на вход сообщение M, подпись S, открытый ключ d пользователя ID и на их основе, также используя общие параметры, отвечает на вопрос: является ли данная подпись валидной. Под валидностью (правильностью) подписи понимается верность того факта, что пользователь ID подписал документ, используя свой набор ключей, и его подпись равна S.

Использование ЭЦП в общем случае служит следующим целям:

1. Контроль целостности передаваемого документа: при любом случайном или преднамеренном изменении документа подпись станет недействительной, потому что вычислена она на основании исходного состояния документа и соответствует лишь ему.
2. Защиту от изменений (подделки) документа: гарантия выявления подделки при контроле целостности делает подделывание нецелесообразным в большинстве случаев.
3. Невозможность отказа от авторства. Так как создать корректную подпись можно, лишь зная личный ключ, а он известен только владельцу, он не может отказаться от своей подписи под документом.
4. Доказательное подтверждение авторства документа: Так как создать корректную подпись можно, лишь зная личный ключ, а он известен только владельцу, он может доказать своё авторство подписи под документом. В зависимости от деталей определения документа могут быть подписаны такие поля, как «автор», «внесённые изменения», «метка времени» и т. д.

3.1.2 История и основные типы

В 1976 году Уитфилдом Диффи и Мартином Хеллманом было впервые предложено понятие «электронная цифровая подпись», хотя они всего лишь предполагали, что схемы ЭЦП могут существовать.

В 1977 году, Рональд Ривест, Ади Шамир и Леонард Адлеман разработали криптографический алгоритм RSA, который без дополнительных модификаций можно использовать для создания примитивных цифровых подписей. Вскоре после RSA были разработаны другие ЭЦП, такие как алгоритмы цифровой подписи Рабина, Меркле.

В 1984 году Шафи Гольдвассер, Сильвио Микали и Рональд Ривест первыми строго определили требования безопасности к алгоритмам цифровой подписи. Ими были описаны модели атак на алгоритмы ЭЦП, а также предложена схема GMR, отвечающая описанным требованиям (Криптосистема Гольдвассер — Микали).

После работ Н. Коблица по эллиптическим кривым стало возможным введение ЭЦП с шифрованием на эллиптических кривых; а после работ А. Шамира - безсертификатные и ID-схемы, используемые сейчас в качестве стандартов во многих странах.

3.1.3 Ситуация в РБ и использование в данной работе

В Республике Беларусь на данный момент принят стандарт [1], в котором указаны протоколы ЭЦП. Алгоритмы ЭЦП предоставляют собой измененные варианты модификации схемы Шнорра для эллиптических кривых. Данная схема имеет доказанно высокий уровень надежности и ее использование является безопасным и достаточным для обеспечения целостности информации. В качестве примитивов блочного шифрования используются алгоритмы из [2]. В данной работе будет использоваться подпись, определенная в стандарте.

3.2 Широковещательное шифрование

Определение 7. *Широковещательное шифрование - это криптографический механизм, позволяющий доставку зашифрованных данных через широковещательный канал таким образом, чтобы лишь "подписанные"пользователи могли расшифровать содержимое.*

3.2.1 История и общие принципы

Данный механизм получил широкое распространение в наше время, в качестве примеров можно привести системы подписки на спутниковое телевидение, IPTV и другие. Впервые формально поставлена и изучена проблема была в 1994 году А. Фиатом и М. Наором. На данный момент, техника решения проблемы обычно является одной из следующих:

1. Конструкции, созданные с помощью комбинаторики.
2. Конструкции, основанные на разделение секрета.
3. Конструкции, основанные на бинарных деревьях.

Наиболее изученным направлением является третье.
Общая концепция BE может быть выражена в виде следующей схемы:

- Алгоритм $\mathcal{GEN}$. На этом этапе происходит инициализация мастер-ключа, создание списка пользователей и выдача каждому из них неких токенов.

- Алгоритм $\mathcal{ENCRYPT}$. На данном этапе центру расчетов подается список запрещенных к рассылке пользователей, и на его основе генерируется заголовок (`header`) сообщения, который доступен в открытом виде. Происходит шифрование собственно сообщения на сессионном ключе, причем таким образом, что лишь разрешенный пользователь может получить из заголовка исходный ключ.

- Алгоритм $\mathcal{DECRYPT}$. На данном этапе по токену пользователя принимается решение: либо он входит в число запрещенных, и ему отказывается в доступе к информации, либо пользователь имеет возможность расшифровать содержимое.

Наиболее употребимой схемой в наши дни является схема `SD` - Subset Difference (разность подмножеств). Основной принцип состоит в том, что на основе идентификаторов пользователей строится бинарное дерево. Далее, множество запрещенных/разрешенных пользователей интерпретируется как покрытие подмножествами на этом бинарном дереве, при этом подмножества имеют вид $S_{i\setminus j}$, что означает "поддерево с корнем в i без собственного поддерева с корнем в j". То есть решается набор покрытия множества, что в общем случае - NP-полная задача. Пример визуальной интерпретации схемы для случая 24 пользователей дан на рисунке 4.

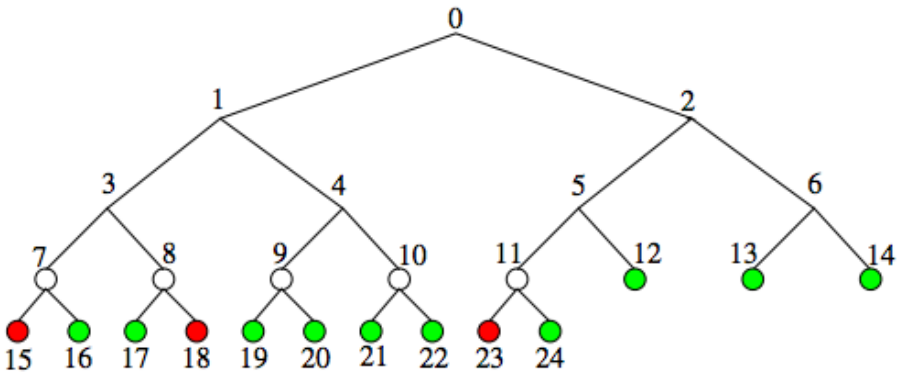

Рис. 4: Визуальная интерпретация SD-схемы.

Красным отмечены запрещенные пользователи, белым и зеленым - разрешенные. В данном случае примерами запрещенного подмножества могут служить множества $S_{7\setminus 16}, S_{8\setminus 17}$, а разрешенных - $S_{2\setminus 5}, S_{1\setminus 3}$. Тогда общее множество разрешенных пользователей можно представить как

$$S_{0\setminus 23} \uplus S_{0\setminus 15} \uplus S_{0\setminus 18}$$

Как видно, существуют и другие способы представления данного множества, и эффективный поиск нужного покрытия является одной из задач данной схемы.

3.2.2 Ситуация в РБ и использование в данной работе

В данный момент широковещательное шифрование является достаточно новой для нашей страны отраслью, вследствие чего пока отсутствуют стандарты данной тематики.

Однако, в НИЛ ПБИТ НИИ ППМИ в 2010 году был создан протокол, основанный на SD схеме и использующий алгоритмы шифрования из [2]. В данной работе упомянутый протокол будет использоваться частично.

4 Встраивание криптографических алгоритмов в документы формата PDF

Выше были рассмотрены различные стандарты и возможности PDF, а также структура данного формата. С другой стороны было показано, какие криптографические механизмы нам необходимы для обеспечения безопасности информации. В настоящем разделе будет предпринята попытка эффективного решения данных задач, для чего необходимо:

1. Выяснить структуру криптографических объектов PDF.
2. Рассмотреть возможные пути реализации создания данных объектов.
3. Реализовать один из путей с последующим анализом результатов.

Рассмотрим структуру объектов подписи и шифрования (под шифрованием имеется в виду обычное симметричное шифрование, так как широковещательное не поддерживается в PDF).

4.1 Цифровая подпись в формате PDF

4.1.1 Общие понятия об ЭЦП в PDF-файлах

Формат PDF поддерживает добавление одной или более ЭЦП в документ. В программах Adobe Acrobat и Adobe Reader добавление ЭЦП реализовано средствами графического интерфейса пользователя. Пользователь, подписывающий документ, имеет возможность заблокировать документ для дальнейших изменений (в том числе и для добавления новых подписей), то есть поставить конечную подпись на документ.

Названные выше программы позволяют также настроить графическое отображение подписи на документе, предоставляя пользователю возможность выбрать набор полей, которые будут отображаться, а именно:

- Время подписи
- Причина подписи
- Рисунок (который может быть копией рукописной подписи)
- Формат имени человека, подписывающего документ

Пользователь может выбрать для подписи любой из плагинов, предназначенных для этого. Встроенным плагином, поставляемым в комплекте с программами Adobe, является Adobe Default Security, поддерживающий следующие алгоритмы хэширования:

1. MD5
2. SHA-1

и следующие алгоритмы подписи:

1. PKCS1 (использующий RSA)
2. PKCS7Detached

3. PKCS7SHA1

Для подписи пользователь должен предоставить свой существующий сертификат либо создать новый. Сертификаты с точки зрения Adobe делятся на 2 вида:

- Созданные пользователем в Acrobat или Reader, являются корневыми самим к себе, не могут быть проверены и включены в валидную цепочку сертификатов.
- Существующие сертификаты, которым Adobe доверяет в следующих случаях:
    1. Корневой сертификат цепочки является сертификатом одним из партнеров Adobe (список партнеров доступен на web-сайте Adobe, а их сертификаты вшиты в программу)
    2. Корневой сертификат цепочки является таким, что ему доверяет система, на которой запущена программа Adobe (к примеру, Windows)

Полученная подпись записывается в содержимое PDF файла и используется для последующей проверки.

4.1.2 Внутреннее представление ЭЦП в формате PDF

В формате PDF для выработки и представления ЭЦП используются, как правило, два объекта: собственно ЭЦП и набор свойств Signature Handler - плагина, реализующего какой-либо механизм ЭЦП (не является обязательным, но рекомендуется Adobe). Оба объекта являются взаимосвязанными и идут в комплекте друг с другом.

4.1.3 Набор свойств Signature Handler

Набор свойств является объектом Content Stream и содержит информацию, экспортируемую плагином. Рассмотрим собственно свойства, их типы данных и значения.

Ключ	Тип данных	Значение
Filter	dictionary	Словарь данных для Signature Handler
PubSec	dictionary	Словарь данных для компонент PubSec
App	dictionary	Словарь данных для PDF/sigQ
SigQ	dictionary	Словарь данных для Pdf/sigQ версий старше, чем в пункте 3

Таблица 2: Свойства Signature Handler

Каждый из этих типов является словарем, который содержит следующие пары "ключ-значение":

Ключ	Тип данных	Значение
Name	string	Название программного модуля, используемого для создания подписи. При использовании в Filter является именем Signature Handler, как правило совпадает с атрибутом Filter
Date	string	Время сборки программного модуля.Не обязано быть в формате PDF Date.

R	number	Номер ревизии версии программного модуля.
PreRelease	number	Булевский флаг, показывающий, является ли версия модуля релизной.
OS	array	Операционная система, под которой может работать модуль.

Пример набора свойств в существующем документе:

```
/Prop_Build
<<
/Filter
<<
        /Name /Adobe.PPKLite
        /Date (Sep 27 2006 00:11:15)
        /R 131101
        /PreRelease true
>>
/PubSec
<<
        /Date (Sep 27 2006 00:05:54)
        /R 13102
        /PreRelease true
>>
/App
  <<
    /Name /Exchange-Pro
    /R 524288
```

4.1.4 Представление собственно ЭЦП в формате документа

В содержимом документа ЭЦП представляет собой объект типа Sig со следующими внутренними объектами:

- Сертификат
- Собственно подпись
- Время подписи

Схематично это можно изобразить как на рисунке 5.

Объект Sig имеет также набор полей, описывающих, к примеру, алгоритм подписи, ByteRange сообщения, которое подписывалось.

Подписывается каждый раз документ полностью, и это находит отображение в свойстве ByteRange. В контексте же содержимого подпись в документе имеет вид как на рисунке 6

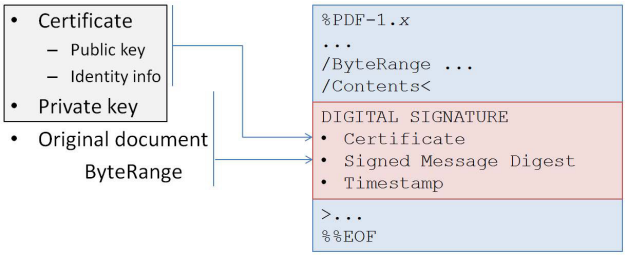

Рис. 5: Содержимое ЭЦП в PDF.

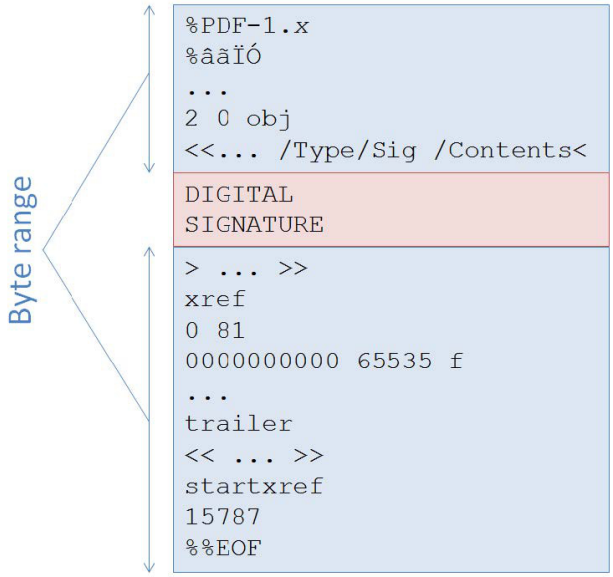

Рис. 6: Объект ЭЦП в контексте содержимого PDF.

4.2 Шифрование в формате PDF

4.2.1 Общее понятие о шифровании в PDF

В настоящий момент формат поддерживает 2 типа шифрования, каждое из которых является по сути накладыванием на файл пароля. Первый тип представляет собой такое шифрование, что для открытия документа необходимо ввести ключ. При неправильном ключе данные будут полностью недоступны. Второй способ представляет собой шифрование с добавленной к нему маской разрешенных действий, среди которых можно отметить:

- Разрешение печатать
- Разрешение менять порядок страниц
- Разрешение редактировать
- Разрешение добавлять комментарии

В обоих случаях используются алгоритмы RC4 и AES. При этом используются оба алгоритма одновременно. Также сущетсвует указать набор данных, который будет подвергаться шифрованию. Им может быть как и все содержимое, так и лишь мультимедийное, так и лишь текстовое.

4.2.2 Объект шифрования в PDF

В содержимом документа информации о шифровании хранится в виде объекта Encrypt, имеющего свои внутренние поля, возможные значения и тип данных которых представлены в таблице 4.

Ключ	Тип данных	Значение
Filter	String	Тип алгоритма, используемого для шифрования документа
V	Integer	Версия алгоритма
R	Integer	Ревизия документа
U	String	Хеш от пароля пользователя
O	String	Хеш от пароля владельца
P	Integer	Флаг разрешенных операций

Таблица 4: Свойства Signature Handler

Под владельцем понимается пользователь, который шифрует документ первый раз (и имеет возможность его дальнейшего изменения). Под собственно пользователем понимается тот, кто использует документ в дальнейшем.

Пример объекта Encryption в PDF-файле:

```
trailer
<<
    /Size 95
    /Root 93 0 R
    /Encrypt 94 0 R
    /ID [<1cf5...>]
>>

94 0 obj
<<
    /Filter /Standard
    /V 1
    /R 2
    /U (xxx...xxx)
    /O (xxx...xxx)
    /P 65472
>>
endobj
```

4.2.3 Механизм шифрования в PDF

Собственно шифрование происходит следующим образом:

1. Контакенация пароля, введенного пользователем со следующей строкой: `28 BF 4E 5E 4E 75 8A 41 64 00 4E 56 FF FA 01 08 2E 2E 00 B6 D0 68 3E 80 2F 0C A9 FE 64 53 69 7A` , после чего берутся первые 32 байта.

2. Добавляется захэшированный пароль владельца (если он существует).

3. Добавляется флаг разрешенных операций.

4. Добавляется идентификатор документа (ключ /ID из трейлера). Под добавлением в последних 3 пунктах имеется в виду контакенация строк.

5. Получение MD5 хэша от результирующей строки. Первые 5 байт хэша будут являться паролем, на котором шифруется ключ пользователя с помощью алгоритма RC4. Подобное ограничение длины связано с с регуляциями экспорта ключа США.

6. В криптообъект кладутся защищенные паролем из пункта 5 ключи пользователя и владельца, а сам документ шифруется ключем из пункта 1 алгоритмом AES. Длина ключа - 256 бит.

4.3 Возможные пути встраивания криптографических механизмов

Имея представление о внутренней структуре объектов шифрования и подписи, а также зная механизмы реализации данных криптографических алгоритмов, у нас теперь стоит задача объединить эти знания и встроить механизмы в документ. Среди различных способов решения поставленной задачи мы рассмотрим два:

1. Парсинг PDF файла и внедрение механизма вручную

2. Реализация механизма как некоего дополнения (plugin) к одному из продуктов Adobe

Рассмотрим каждый из способов в отдельности.

Преимуществом первого способа является сравнительная легкость реализации. Для реализации любого из механизмов нам нужно лишь распарсить файл, взять нужные данные (интерпретируя их как массив байт), применить алгоритмы механизма и записать результат в соответствие с необходимым форматом. Среди недостатков можно отметить:

- Сложность алгоритма `parse`. Несмотря на существующий стандарт формата, для этого алгоритма стоит предусмотреть несколько случаев, связанных с кросс-ссылками, закладками и прочими элементами, не являющимися основным текстом. В данном контексте использование сторонних библиотек недопустимо, так как в общем случае у нас нету оснований доверять компании-разработчику

- Необходимость запуска нашей программы каждый раз перед действиями с PDF-файлом. Это необходимо для того, чтобы избежать попыток программ-просмотрщиков и -редакторов перехватить процесс проверки подписи или дешифрации. Данный недостаток сводит на нет все преимущества использования PDF,поскольку в этом случае гораздо легче было бы интерпретировать целый файл как бинарный и применять механизмы к нему.

Второй способ лишен данных недостатков, так как написание plugin к Adobe Reader или Adobe Acrobat предполагает использование Adobe API, что ведет за собой следующие преимущества:

- Компания-разработчик (Adobe), к которой есть доверие

- Единый стандарт и поддержка предыдущих версий PDF

Мы будем использовать API к Adobe Acrobat, которое является свободно распространяемым и доступно для двух платформ (Windows, Mac OSX).

На данном этапе проектирования единственными минусами являются универсальность API, что ведет за собой сложную структуру и некую обфусцированность кода, а также отсутствие поддержки широковещательного шифрования в чистом виде. Ниже будут предложены попытки решения этих проблем.

4.4 Acrobat API для криптографических механизмов

В этом и следующих разделах будем рассматривать структуру Acrobat API, архитектуру plugin-ов к Adobe Acrobat на примере выработки и проверки ЭЦП, а также предложим решение проблемы поддержки широковещательного шифрования.

Определение 8. *Интерфейс программирования приложений, API — набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) для использования во внешних программных продуктах.*

Acrobat API, разработанный компанией Adobe, предоставляет широкий набор функций для изменения как и внешнего представления подписи в документе, так и работы непосредственно с криптографическими примитивами.

Схематично Signature API как часть Acrobat API можно представить как на рисунке 7

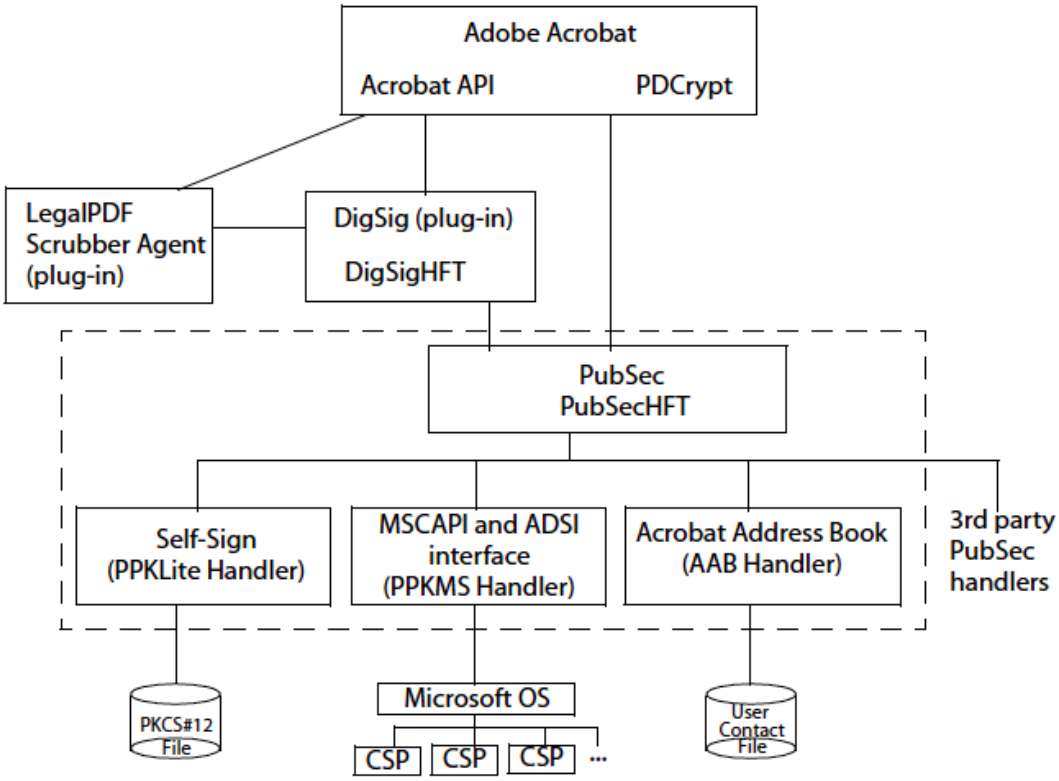

Рис. 7: Взаимоотношение криптографических блоков Acrobat API.

Для работы с цифровыми подписями предзначены 2 модуля: `PubSecHFT` и `DigSigHFT`. PubSec образует высокоуровневый интерфейс для работы с ЭЦП, при этом для работы с непосредственно ЭЦП используются вызовы методов DigSig. Таким образом, PubSec является надстройкой над DigSig и представляет в дополнение некоторые дополнительные сервисы, поэтому в нашей разработке будет использоваться PubSec.

PubSec предоставляет следующую функциональность:

- Открытие и закрытие зашифрованных документов

- Проверка и выработка ЭЦП
- Доступ и создание дайджестов буферов данных
- Импорт и экспорт данных сертификатов, управление сертификатами в Acrobat Address Book
- Управление графическим видом функции
- Использование внешних хэндлеров для
    1. Алгоритмов ЭЦП (выработка, проверка)
    2. Алгоримтов шифования
    3. Алгоритмов ID-based аутенфикации

PubSec содержит набор системных функций (CallBack), которые являются либо обработчиками событий (открытие документа), либо функциями, выполняющими примитивы (атомарные операции приложения). Для внедрения своего плагина к системе PubSec используется следующий алгоритм:

1. Реализовать необходимые CallBack и атомарные функции, соблюдая количество и порядок параметров ,а также тип возвращаемого значения как в API
2. Реализовать класс PubSecHandler, в котором зарегистрировать в системе реализованные в пункте 1 функции
3. Зарегистрировать хэндлер в системе PubSec

Рассмотрим далее конкретную архитектуру нашего плагина.

4.5 Архитектура плагина

Разделим все функции и классы нашего плагина на следующие группы:

- Непосредственно реализация CallBack и атомарных функций системы PubSec
- Служебные функции: инициализация плагина, информация о плагине
- Регистрация функций из пункта 1 и хэндлера плагина

Предлагается следующая архитектура:

- Реализация класса SignInit, который будет поддерживать функции из пункта 2.
- Поддержание набора тех классов из PubSec, которые необходимо реализовать
- Реализация указанных классов следующим образом: при использовании CallBack с одним из параметров как объектом данного класса в нашу реализацию класса добавляется такой же метод без этого параметра
- Реализация основного хэндлера SignHandler, в котором реализуем статические методы для переопределения CallBack и будем их использовать следующим образом: при наличии системного CallBack

```
def Foo(SysStruct a, param1, param2, ..., paramn):
        return retType
```

реализуем следующий метод

```
def static retType SignHandler::
        Foo(SysStruct a, param1, param2, ..., paramn): {
        if not initialized(mySysStruct) :
                a.Foo(param1, param2, ..., paramn)
        else:
                mySysStruct.Foo(param1, param2, ..., paramn)
        return retType
```

То есть в хэндлере будут храниться копии объектов классов из предыдущего пункта. Статичность необходима для передачи адреса при замене системной функции.

- Регистрация самого хэндлера и системных функций производится в одном месте в SignHandler.

Ниже приведены некоторые фрагменты исходного кода:

1. Регистрация хэндлера и реализации системных функций

```
bool BignSignHandler::Register() {
        fHandlerName = ASAtomFromString( DOCSIGN_HANDLER_NAME );

        ACROASSERT( !fbHandlerIsInit && !fbHandlerIsRegistered );

        if( gPubSecHFT == NULL || gDigSigHFT == NULL
        || gAcroFormHFT == NULL || fbHandlerIsRegistered )
```

```
        return false;

if( !fbHandlerIsInit ) {

        memset( &fHandlerRec, 0, sizeof(fHandlerRec) );

        BignSignEngine *psEngine = new BignSignEngine;
        if( !psEngine )
                return false;

        fHandlerRec.size = sizeof(PubSecHandlerRec);
        fHandlerRec.engine = (PubSecEngine) psEngine;

        fHandlerRec.getBoolProperty =
        ASCallbackCreateProto(PSGetBoolPropertyProc,
        BignSignHandler::GetBoolProperty );
        fHandlerRec.getAtomProperty =
        ASCallbackCreateProto(PSGetAtomPropertyProc,
        BignSignHandler::GetAtomProperty );
        fHandlerRec.getTextProperty =
        ASCallbackCreateProto(PSGetTextPropertyProc,
        BignSignHandler::GetTextProperty );
        fHandlerRec.getInt32Property =
        ASCallbackCreateProto(PSGetInt32PropertyProc,
        BignSignHandler::GetInt32Property );

        fHandlerRec.newEngine =
        ASCallbackCreateProto(PSNewEngineProc,
        BignSignHandler::NewEngine );
        fHandlerRec.destroyEngine =
        ASCallbackCreateProto(PSDestroyEngineProc,
        BignSignHandler::DestroyEngine );

        fHandlerRec.sessionAcquire =
        ASCallbackCreateProto(PSSessionAcquireProc,
         BignSignHandler::SessionAcquire );
        fHandlerRec.sessionRelease =
        ASCallbackCreateProto(PSSessionReleaseProc,
        BignSignHandler::SessionRelease );
        fHandlerRec.sessionReady =
        ASCallbackCreateProto(PSSessionReadyProc,
        BignSignHandler::SessionReady );
        fHandlerRec.performOperation =
        ASCallbackCreateProto(PSPerformOperationProc,
        BignSignHandler::PerformOperation );


        fHandlerRec.sigGetSigProperties =
```

```
        ASCallbackCreateProto(PSSigGetSigPropertiesProc,
        BignSignHandler::SigGetSigProperties );
        fHandlerRec.sigAuthenticate = ASCallbackCreateProto(
        PSSigAuthenticateProc, BignSignHandler::SigAuthenticate );
        fHandlerRec.sigGetSigValue = ASCallbackCreateProto(
        PSSigGetSigValueProc, BignSignHandler::SigGetSigValue );


        fHandlerRec.sigValidate = ASCallbackCreateProto(
        PSSigValidateProc, BignSignHandler::SigValidate );
        fHandlerRec.sigValidateDialog = NULL;
        fHandlerRec.sigPropDialog = NULL;

        // SigVal methods
        fHandlerRec.sigValGetText = ASCallbackCreateProto(
        PSSigValGetTextProc, BignSigVal::GetText );
        fHandlerRec.sigValGetAPLabel = ASCallbackCreateProto(
        PSSigValGetAPLabelProc, DSSigVal::GetAPLabel );

        // Cert exchange methods
        fHandlerRec.exportData =
        ASCallbackCreateProto(PSExportDataProc,
         BignSignHandler::ExportData );
        fHandlerRec.importData = NULL;

        // Encryption methods
        fHandlerRec.cryptOpenCMSEnvelope =
        ASCallbackCreateProto(
        PSOpenCMSEnvelopeProc,
         BignSignHandler::openCMSEnvelope);
        fHandlerRec.cryptGetImplicitRecipients =
        ASCallbackCreateProto(
        PSGetImplicitRecipientsProc,
         BignSignHandler::getImplicitRecipients);

        fbHandlerIsInit = true;
}



ASBool bOk = PSRegisterHandler( gExtensionID, &fHandlerRec );
if( !bOk ) {
        // Destroy fHandlerRec
        Unregister();
}

fbHandlerIsRegistered = true;
```

```
        return true;
}
```

2. Реализация статического метода хэндлера

```
ACCB1 DSRetCode ACCB2 BignSignHandler::SigGetSigValue(
 PubSecEngine engine, PSSigGetSigValueParams inOutParams ) {
        PSTRY(psEngine,engine) {
                if( inOutParams ) {
                         psEngine->sigGetSigValue( inOutParams);
                        retCode = kDSOk;
                }
        } PSCATCH;
}
```

Для внедрения алгоритмов BIGN была использована библиотека bee2, реализованная Агиевичем С.В.

4.6 Решение проблемы широковещательного шифрования

Структура плагина для шифрования данных будет практически идентичной, различие лишь в вызываемых системных функциях. Однако существует последняя проблема, которую необходимо решать. Как отмечалось выше, в формате PDF отсутствует поддержка широковещательного шифрования в чистом виде.

Анализ схемы реализации обычного шифрования (при разработке плагина) показал, что шаг 6 выполняется всегда, то есть шифрование информации происходит алгоритмом AES и у нас нету возможности встроиться в него. Данное ограничение приводит нас к следующему решению, которое будет сочетать в себе как элементы протокола BE, так и стандартного шифрования PDF.

Схема решения:

- Введем ограничения на длину сессионного ключа в BE и примем ее равной `len` = 32 байта.
- Будем перехватывать первые 5 пунктов шифрования PDF.
- При введенном пароле пользователя оставим шаг 1 шифрования неизменным, а далее проведем следующие шаги:
    1. Считаем с `.config` файла множество запрещенных пользователей и на их основе с помощью сессионого ключа (которым будет являться пароль пользователя) составим `header` шифртекста.
    2. Запишем `header` как пароль пользователя (здесь используется то, что заголовок является свободно распространяемой информацией).
    3. Зашифруем сам документ алгоритмом AES как и было раньше.
- При попытке открытия защищенного файла предпринимаются следующие шаги:
    1. Пользователь вводит свой `token`
    2. Алгоритм проверяет, возможно ли по введенным данным получить сессионный ключ из заголовка.

3. В случае положительного ответа, значение токена подменяется сессионным ключем и происходит расшифровка документа.

Несмотря на некоторую неестетсвенность подобной схемы, она эффективно решает вопрос о внедрении широковещательного шифрования в PDF файлы, что особенно ценно при том факте, что в открытом виде данная возможность отсутствует.

4.7 Недостатки предложенного решения

Несмотря на описанные достоинства предложенных решений, у них существуют некоторые недостатки, анализ которых и последующее исправление порождает область для дополнительных исследований. Основными недостатками являются:

1. Отсутствие решения для внедрения полного протокола BE
2. Неестественность схемы решения BE
3. "Закрытость плагинов". Несмотря на предоставление API и существующие лазейки использования своих алгоритмов, Acrobat будет показывать, что в качестве алгоритмов подписи и шифрования будут использоваться стандартные методы Acrobat, которые обрабатываются в наших плагинах. Это влечет за собой введение дополнительной документации для пользователей во избежание непонимания.

5 Заключение

В данной работе был изучен один из возможных стандартов формата электронного документа, обосновано его использование и его преимущество. Были предприняты практические попытки внедрения криптографических механизмов в PDF, используя синтез знаний предыдущих лет и изучения API и формата PDF.

Список литературы

[1] СТБ 34.101.45-2011 "Информационные технологии и безопасность. Алгоритмы электронной цифровой подписи на основе эллиптических кривых".

[2] СТБ 34.101.31-2011 "Информационные технологии. Защита информации. Криптографические алгоритмы шифрования и контроля целостности"

[3] C.P. Schnorr, Frankfurt University, Efficient identification and signatures for smart cards.

[4] Bellare M., Neven G. Multi-signatures in the plain public-key model and a general forking lemma. In: ACM CCS 06, ACM Press, 2006, 390–399.

[5] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. Contemporary Mathematics, 324:71–90, 2003.

[6] The LSD Broadcast Encryption Scheme, CRYPTO 2002, LNCS 2442, pp. 47 - 60

[7] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas R. Stinson, editor, CRYPTO, volume 773 of Lecture Notes in Computer Science, pages 480-491. Springer, 1993.

[8] http://www.adobe.com/devnet/acrobat/overview.html

[9] http://livedocs.adobe.com/acrobat_sdk/9.1/Acrobat9_1_HTMLHelp/wwhelp/wwhimpl/js/html/wwhelp.htm?& accessible=true

[10] http://livedocs.adobe.com/acrobat_sdk/9.1/Acrobat9_1_HTMLHelp/API_References/Acrobat_API_Reference/index.html

[11] http://www.adobe.com/devnet-docs/acrobatetk/tools/QuickKeys/Acrobat_SignatureCreationQuickKeyAll.pdf

[12] http://www.datalogics.com/pdf/doc/Version6.1/CoreAPIReference.pdf