Инфраструктура Открытого Ключа

- Уязвимости открытого ключа
- Основные понятия Инфраструктуры Открытого Ключа
- Архитектура PKI
- ASN.1: спецификация базовой нотации
- LDAP - Lightweight Directory Access Protocol
  - LDAP vs базы данных
  - Модели LDAP
  - Модель именования
  - Распределенное множество серверов LDAP
  - Протокол LDAP
- Расширения сертификата
- On-line протокол запроса статуса сертификата (OCSP)
- Изменение ключа CA

Уязвимости открытого ключа

- Одним из требований к алгоритмам цифровой подписи является требование, чтобы было вычислительно невозможно, зная открытый ключ $\mathbf{Key}_{\mathbf{pub}}$, определить закрытый ключ $\mathbf{Key}_{\mathbf{priv}}$.
- Казалось бы, открытый ключ $\mathbf{Key}_{\mathbf{pub}}$ можно распространять по небезопасным сетям и хранить в небезопасных репозиториях. Но при этом следует помнить, что при использовании цифровой подписи **необходимо быть уверенным, что субъект, с которым осуществляется взаимодействие с использованием алгоритма открытого ключа, является собственником соответствующего закрытого ключа**.
- В противном случае возможна **атака**, когда **оппонент заменяет открытый ключ законного участника своим открытым ключом, оставив при этом идентификатор законного участника без изменения**. Это позволит ему создавать подписи от имени законного участника и читать зашифрованные сообщения, посланные законному участнику, используя для этого свой закрытый ключ, соответствующий подмененному открытому ключу.

Уязвимости открытого ключа

- Для предотвращения такой ситуации следует использовать сертификаты, которые являются структурами данных, связывающими открытый ключ с субъектом. Для связывания необходимо наличие доверенного сертификационного центра (**Certification Authority – CA**), который **проверяет идентификацию субъекта** и **подписывает его открытый ключ** и некоторую дополнительную информацию своим закрытым ключом.

- **Целью PKI является предоставление доверенного и действительного открытого ключа участника, а также управление всем жизненным циклом сертификата открытого ключа.**

Основные понятия Инфраструктуры Открытого Ключа

- Основным понятием Инфраструктуры Открытого Ключа является понятие сертификата.
- Сертификат участника, созданный CA, имеет следующие характеристики:

Любой участник, имеющий открытый ключ CA, может восстановить открытый ключ участника, для которого CA создал сертификат.

Никто другой, кроме данного сертификационного центра, не может модифицировать сертификат без обнаружения этого проверяющей стороной.

Мы будем рассматривать сертификаты **X.509**, хотя существует достаточно много сертификатов других форматов.

- Стандарт X.509 первоначально являлся частью **стандарта X.500** и описывал основные требования к аутентификации в Каталогах X.500. Но X.509 используется не только в контексте сервиса Каталога X.500. Сертификаты, определяемые данным стандартом, используются практически всеми программными продуктами, относящимися к обеспечению сетевой безопасности.

Основные понятия Инфраструктуры Открытого Ключа

- Стандарты ITU-TX.509 и ISO/IEC 9594-8, которые впервые были опубликованы в 1988 году как часть рекомендаций **X.500 Директории**, определили формат сертификата **X.509**. Формат сертификата в стандарте 1988 года называется форматом версии 1 (v1). Стандарт X.500 был пересмотрен в 1993 году, в результате чего было добавлено несколько новых полей в формат сертификата X.509, который был назван форматом версии 2 (v2).
- Опыт реализации первой и второй версий говорит о том, что форматы сертификата v1 и v2 имеют ряд недостатков. Самое важное, что для хранения различной информации требуется больше полей. В результате ISO/IEC, ITU-T и ANSI X9 разработали формат сертификата X.509 версии 3 (v3). Формат v3 расширяет формат v2, обеспечивая возможность добавления дополнительных полей расширения. Конкретные типы полей расширения могут быть определены в стандартах или определены и зарегистрированы любой организацией или сообществом. В 1996 году стандартизация базового формата v3 была завершена.
- Стандарт ISO/IEC, ITU-T и ANSI X9 являются очень общими, чтобы применять их на практике. Для того чтобы разрабатывать интероперабельные реализации, использующие сертификаты X.509 v3, необходимо четко специфицировать формат сертификата X.509 v3. Специалисты IETF разработали профиль сертификата X.509 v3 и опубликовали его в **RFC 3280**.

Основные понятия Инфраструктуры Открытого Ключа

Пояснение	Поле сертификата	Версия 1	Версия 2	Версия 3
Версия сертификата	v1, v2 или v3			
Целое число, идентифицирующее данный сертификат, которое должно быть уникальным для всех сертификатов, выпущенных данным CA	Серийный номер			
CA, который создал и подписал сертификат	Имя CA, выпустившего сертификат			
Период действительности сертификата состоит из двух дат, в промежутке между которыми сертификат считается действительным	Дата начала периода действительности сертификата			
	Дата конца периода действительности сертификата			
Конечный участник, для которого создан данный сертификат	Имя субъекта (конечного участника)			
Открытый ключ субъекта и алгоритм, для которого этот ключ был создан	Алгоритм			
	Параметры			
	Открытый ключ субъекта			
	Уникальный идентификатор CA			
	Уникальный идентификатор субъекта			
	Расширения			
Подпись охватывает все поля сертификата, кроме данного, и состоит из хеш-кода этих полей, подписанного закрытым ключом CA	Подпись, созданная закрытым ключом CA для всех полей сертификата	Все версии		

Основные понятия Инфраструктуры Открытого Ключа

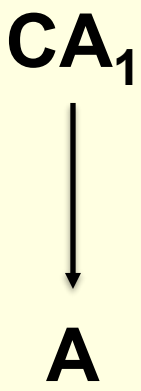
- Часто используется следующая нотация для обозначения сертификата:

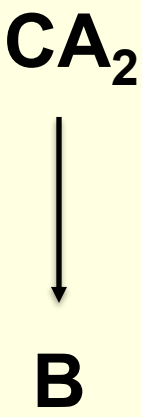
CA << A >>

- сертификат пользователя **A**, выданный сертификационным центром **CA**.

- **CA** подписывает сертификат своим закрытым ключом. Если соответствующий открытый ключ известен проверяющей стороне, то она может проверить, что сертификат, подписанный **CA**, действителен.
- Так как сертификаты не могут быть изменены без обнаружения этого, их можно разместить в общедоступной директории и пересылать по открытым каналам связи без опасения, что кто-то может их изменить.

Основные понятия Инфраструктуры Открытого Ключа



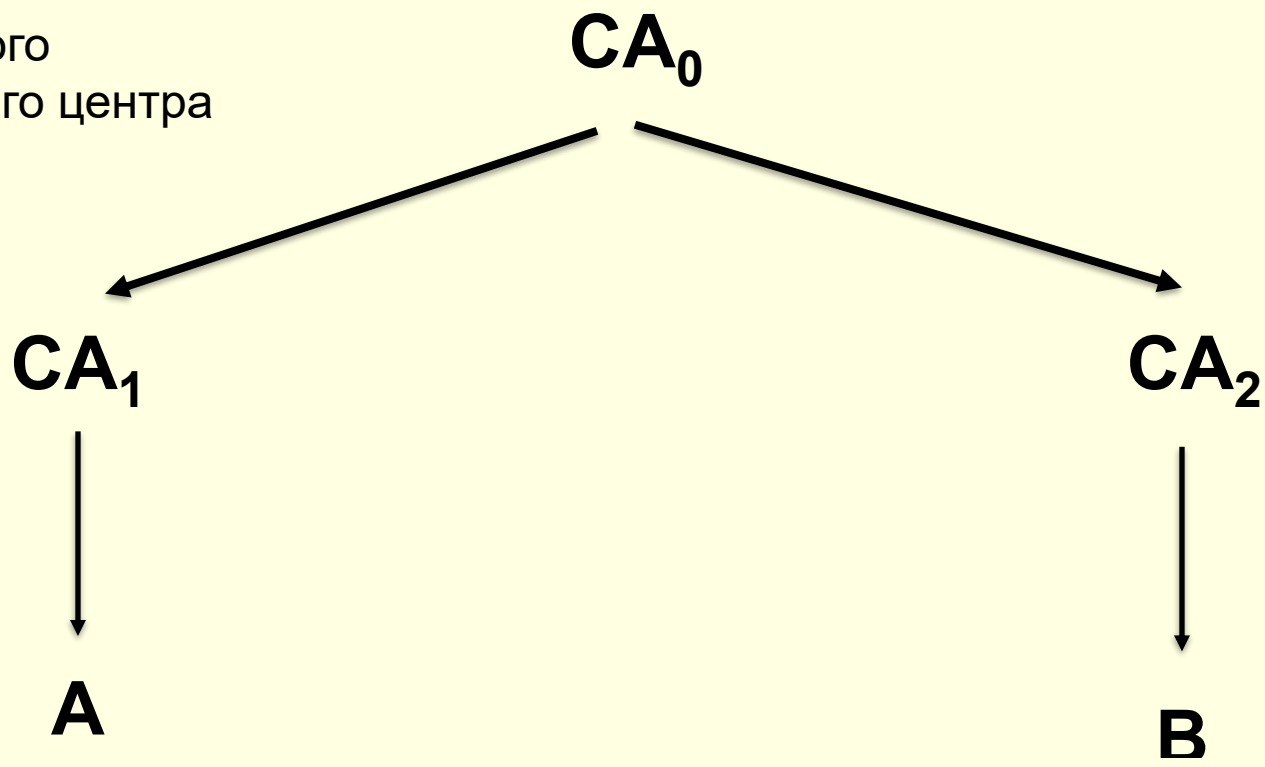


CA_1<<A>>

CA_2<<B>>

Основные понятия Инфраструктуры Открытого Ключа

Создание головного сертификационного центра

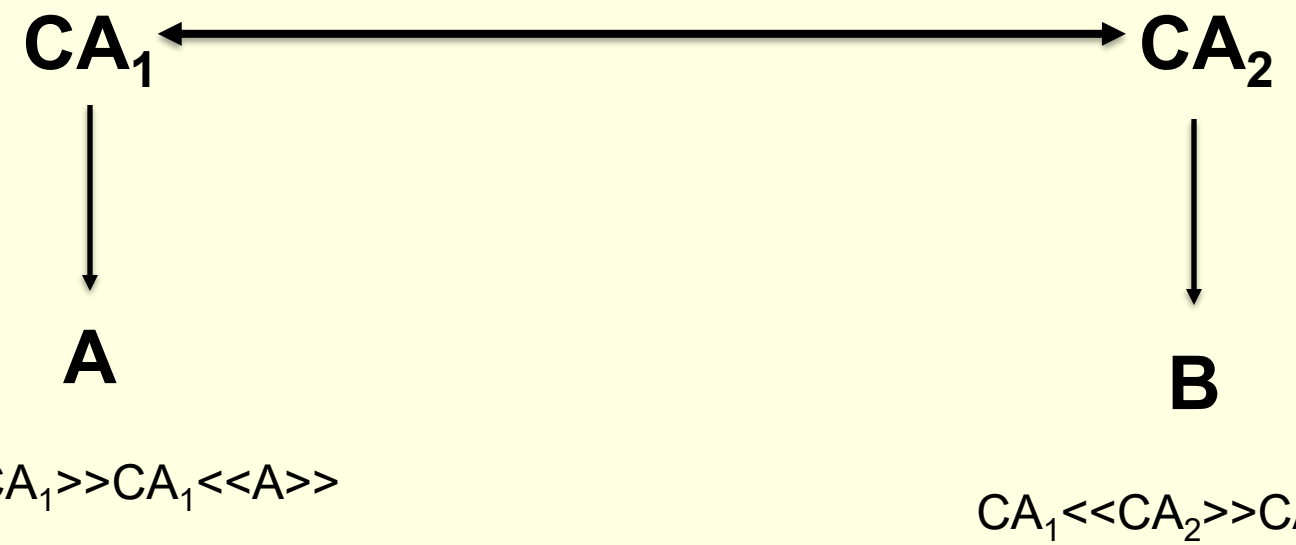


$CA_0<<CA_1>>CA_1<<A>>$

$CA_0<<CA_2>>CA_2<<B>>$

Основные понятия Инфраструктуры Открытого Ключа

Кросс-сертификация



Основные понятия Инфраструктуры Открытого Ключа

- В любом случае если **B** имеет сертификат **A**, **B** уверен, что сообщение, которое он зашифровывает открытым ключом **A**, никто не может просмотреть, и что сообщение, подписанное закрытым ключом **A**, не изменялось.
- При большом количестве пользователей неразумно подписывать сертификаты всех пользователей у одного **CA**. Кроме того, если существует единственный **CA**, который подписывает сертификаты, каждый пользователь должен иметь копию открытого ключа **CA**, чтобы проверять подписи. Этот открытый ключ должен быть передан каждому пользователю абсолютно безопасным способом (с обеспечением целостности и аутентификации), чтобы пользователь был уверен в подписанных им сертификатах. Таким образом, в случае большого количества пользователей лучше иметь несколько **CA**, каждый из которых безопасно предоставляет свой открытый ключ некоторому подмножеству пользователей.
- Теперь предположим, что **A** получил сертификат от уполномоченного органа $\mathbf{CA}_1$, и **B** получил сертификат от уполномоченного органа $\mathbf{CA}_2$. Если **A** не знает безопасным способом открытый ключ $\mathbf{CA}_2$, то сертификат **B**, выданный $\mathbf{CA}_2$, для него бесполезен. **A** может прочитать сертификат **B**, но не в состоянии проверить подпись. Тем не менее, если два **CA** могут безопасно обмениваться своими открытыми ключами, возможна следующая процедура для получения **A** открытого ключа **B**.

Основные понятия Инфраструктуры Открытого Ключа

1. **A** получает сертификат $\mathbf{CA_2}$, подписанный $\mathbf{CA_1}$. Так как **A** знает открытый ключ $\mathbf{CA_1}$ надежным способом, **A** может получить открытый ключ $\mathbf{CA_2}$ из данного сертификата и проверить его с помощью подписи $\mathbf{CA_1}$ в сертификате.
2. Затем **A** получает сертификат **B**, подписанный $\mathbf{CA_2}$. Так как **A** теперь имеет открытый ключ $\mathbf{CA_2}$ надежным способом, **A** может проверить подпись и безопасно получить открытый ключ **B**.

   - Для получения открытого ключа **B** **A** использует цепочку сертификатов. В приведенной выше нотации эта цепочка выглядит следующим образом:

     $$\mathbf{CA_1\langle\langle\ CA_2\ \rangle\rangle\ CA_2\langle\langle\ B\ \rangle\rangle}$$

   - Аналогично **B** может получить открытый ключ **A** с помощью такой же цепочки:

     $$\mathbf{CA_2\langle\langle\ CA_1\ \rangle\rangle\ CA_1\langle\langle\ A\ \rangle\rangle}$$

Основные понятия Инфраструктуры Открытого Ключа

- Данная схема не обязательно ограничена цепочкой из двух сертификатов. Для получения цепочки может использоваться путь **CA** произвольной длины. Цепочка, содержащая N элементов, выглядит следующим образом:

$$CA_1 << CA_2 >> CA_2 << CA_3 >> \ . \ . \ . \ CA_N << B >>$$

- В этом случае каждая пара **CA** в цепочке (CA_i, CA_{i+1}) должна создать сертификаты друг для друга.
- Все эти сертификаты CA необходимо разместить в каталоге, и пользователи должны иметь информацию о том, как они связаны друг с другом, чтобы получить путь к сертификату открытого ключа другого пользователя. Это определяет Инфраструктуру Открытого Ключа, изучению которой и посвящены следующие лекции.

Основные понятия Инфраструктуры Открытого Ключа

- Когда сертификат выпущен, считается, что он будет использоваться в течение всего своего **периода действительности**. Однако могут произойти события, в результате которых сертификат станет **недействительным до завершения своего периода действительности**. К таким событиям относится изменение имени, изменение связывания между субъектом и **CA** (например, сотрудник увольняется из организации) и компрометация или предположение компрометации соответствующего закрытого ключа. При таких обстоятельствах **CA** должен отменить сертификат.
- X.509 определяет следующий **способ отмены сертификата**. Предполагается, что каждый **CA** периодически выпускает подписанную структуру данных, называемую списком отмененных сертификатом **(Certificate Revocation List – CRL).**

Отметка времени
SN #1
…
SN #2
Подпись выпускающего CRL

Основные понятия Инфраструктуры Открытого Ключа

- CRL является помеченным временем списком, который подписан CA или специальным выпускающим CRL и в котором перечислены отмененные на текущий момент сертификаты с не истекшим периодом действительности, указанным в сертификате. Данный список становится свободно доступен в открытом репозитории. Каждый отмененный сертификат идентифицируется в CRL своим серийным номером. Когда использующая сертификат система получает сертификат, эта система не только проверяет подпись сертификата и действительность, но также получает свежий CRL и проверяет, не находится ли в этом CRL серийный номер сертификата. Понятие «свежий» может зависеть от локальной политики, но обычно означает самый последний выпущенный CRL. Новый CRL выпускается регулярно (например, каждый час, день или неделю). При получении уведомления об отмене запись добавляется в CRL.

Основные понятия Инфраструктуры Открытого Ключа

- **Преимущество** данного метода отмены состоит в том, что **CRL могут распространяться теми же способами, что и сами сертификаты, а именно через небезопасные серверы и коммуникации**.
- При этом **недостатком** данного метода отмены CRL является то, что **точность отмены ограничена периодом выпуска CRL**. Например, если об отмене сообщено в текущий момент, то об отмене не будут оповещены использующие сертификат системы до тех пор, пока не будет выпущен новый CRL – это может занять час, день или неделю, в зависимости от частоты создания CRL.

Основные понятия Инфраструктуры Открытого Ключа

- Как и формат сертификата X.509 v3, для того чтобы обеспечить интероперабельность реализаций различных производителей, необходимо иметь строгий формат X.509 CRLv2. Существуют также **специальные протоколы** и соответствующие им форматы сообщений, поддерживающих on-line оповещения об отмене. On-line методы оповещения об отмене могут применяться в некоторых окружениях как альтернатива CRL. On-line проверка отмены может существенно уменьшить промежуток между отправкой сообщения об отмене и получением этой информации проверяющей стороной. После того как CA получает сообщение об отмене, любой запрос к on-line сервису будет корректно отображать действительность сертификата. Однако эти методы навязывают новые требования безопасности: проверяющая сторона должна доверять on-line сервису действительности, в то время как репозиторий не обязательно должен быть доверяемым.

Основные понятия Инфраструктуры Открытого Ключа

В Инфраструктуре Открытого Ключа используются следующие термины и понятия.

- **Public Key Infrastructure (PKI)** – инфраструктура открытого ключа – это множество аппаратуры, ПО, людей, политик и процедур, необходимых для создания, управления, хранения, распределения и отмены сертификатов, основанных на криптографии с открытым ключом.
- **End Entity (EE) –** конечный участник, для которого выпущен данный сертификат. Важно заметить, что здесь под конечными участниками подразумеваются не только люди и используемые ими приложения, но также и исключительно сами приложения (например, для безопасности уровня IP).
- **Certificate Authority (CA)** – сертификационный или удостоверяющий центр – это уполномоченный орган, который создает и подписывает сертификаты открытого ключа. Дополнительно CA может создавать пары закрытый/открытый ключ для конечного участника. Важно заметить, что CA отвечает за сертификаты открытого ключа в течение всего времени их жизни, а не только в момент выпуска.

Основные понятия Инфраструктуры Открытого Ключа

- **Public Key Certificate (PKC)** — сертификат открытого ключа или просто сертификат – это структура данных, содержащая открытый ключ конечного участника и другую информацию, которая подписана закрытым ключом CA, выпустившим данный сертификат.
- **Registration Authority (RA)** — регистрационный центр - необязательный участник, ответственный за выполнение некоторых административных задач, необходимых для регистрации конечных участников. Такими задачами могут быть: подтверждение идентификации конечного участника; проверка значений, которые будут указаны в создаваемом сертификате; проверка, знает ли конечный участник закрытый ключ, соответствующий открытому ключу, указанному в сертификате. Заметим, что RA сам также является конечным участником.

Основные понятия Инфраструктуры Открытого Ключа

Причины, по которым могут создаваться RA, разделяют на технические и организационные. К техническим относятся следующие причины:

Если используется аппаратный токен, то не все конечные участники имеют необходимое оборудование для его инициализации; оборудование RA может включать необходимую функциональность, что также может быть вопросом политики.

Не все конечные участники могут иметь возможность опубликовать свои сертификаты; для этого может использоваться RA.

RA может иметь возможность выпускать подписанные запросы отмены от имени конечного участника, тогда как конечный участник не всегда имеет возможность сделать это (например, если пара ключей потеряна).

Основные понятия Инфраструктуры Открытого Ключа

Организационные причины для использования RA следующие:

- Может оказаться более эффективным сконцентрировать некоторую функциональность в оборудовании RA, чем иметь данную функциональность для всех конечных участников (особенно если используется специальное оборудование для инициализации токена).
- Наличие RA может уменьшить число необходимых CA.
- RA могут быть оптимальнее размещены с целью идентификации людей и присваивании им «электронных» имен, особенно если CA физически удален от конечного участника.
- Во многих случаях уже существуют определенные административные структуры, которые могут играть роль RA.

Основные понятия Инфраструктуры Открытого Ключа

- **Выпускающий CRL -** необязательный компонент, которому CA делегирует функции опубликования списков отмененных сертификатов.
- **Certificate Policy (CP) –** политика сертификата - поименованное множество правил, которое определяет применимость сертификата открытого ключа для конкретного сообщества или класса приложений с общими требованиями безопасности. Например, конкретная политика сертификата может указывать применимость сертификата открытого ключа для аутентификации транзакций данных при торговле товарами в данном ценовом диапазоне.
- **Certificate Practice Statement (CPS) –** утверждение об используемой практике, в соответствии с которой сотрудники сертификационного центра выпускают сертификаты открытого ключа.
- **Relying Party (RP) –** проверяющая сторона - пользователь или агент (например, клиент или сервер), который использует сертификат для надежного получения открытого ключа субъекта и, быть может, некоторой дополнительной информации.

Основные понятия Инфраструктуры Открытого Ключа

- **Root CA** — CA, которому непосредственно доверяет конечный участник; это означает безопасное получение открытого ключа корневого CA, требующее некоторых внешних по отношению к PKI шагов. Этот термин не означает, что корневой CA обязательно должен быть вершиной иерархии. Заметим, что термин «доверенный якорь» имеет то же значение, что и корневой CA в данном случае.
- **Репозиторий** - система или набор распределенных систем, которые хранят сертификаты и CRL и предназначены для распределения этих сертификатов и CRL между конечными участниками.

Основные понятия Инфраструктуры Открытого Ключа

- **Пользователи** систем, основанных на открытом ключе, должны быть уверены, что, когда они используют открытый ключ, субъект, с которым они связываются, является собственником соответствующего закрытого ключа. Эта уверенность достигается благодаря использованию **сертификата открытого ключа**, которые являются структурами данных, связывающих значения открытого ключа с субъектами. Связывание обеспечивается наличием доверенного CA, который проверяет идентификацию субъекта и подписывает сертификат.
- **Сертификат имеет ограниченное время жизни**, которое указывается в подписанном содержимом. Так как подпись и своевременность могут быть независимо проверены клиентом, использующим сертификат, **сертификаты могут распространяться по небезопасным коммуникациям и серверным системам** и кэшироваться в небезопасном хранилище в системах, использующих сертификаты.

Основные понятия Инфраструктуры Открытого Ключа

- **Сертификаты используются для проверки действительности подписанных данных.** Имеется определенная специфика, относящаяся к используемому алгоритму, но общий процесс выглядит следующим образом (не существует требований, относящихся к порядку, в котором должны выполняться перечисленные проверки; разработчики свободны в выборе наиболее эффективного способа для своих систем).

Основные понятия Инфраструктуры Открытого Ключа

- Проверяющая сторона (Relying Party – RP) выполняет проверку подписи в соответствии с **математическим алгоритмом**.
- RP при получении подписанных данных должна убедиться, что **предоставленная идентификация пользователя соответствует идентификации, содержащейся в сертификате.**
- RP смотрит, **не отменен ли в полученной цепочке сертификатов какой-либо РКС** (например, посредством получения соответствующего текущего CRL или on-line запроса статуса сертификата) и все ли сертификаты были действительны в момент подписывания данных.
- **Если все эти проверки проходят,** PR может считать, что данные были подписаны указанной подписывающей стороной. Процесс для ключей, используемых для шифрования, аналогичен.

Архитектура PKI

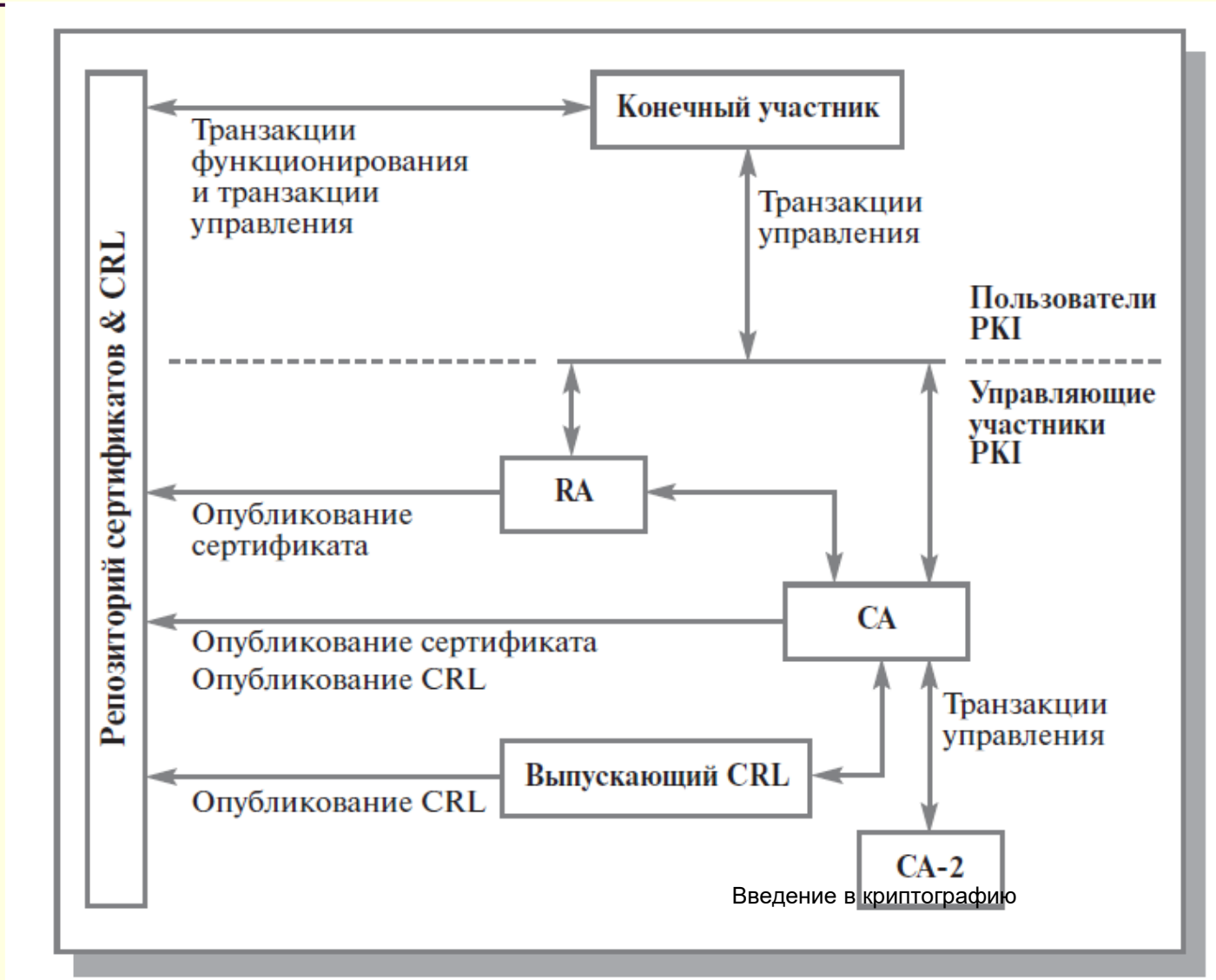
- Основой для достижения безопасности в интернет являлось создание протоколов безопасности, таких как TLS, SSH и IPSec. Все эти протоколы используют криптографию с открытым ключом для предоставления таких сервисов как конфиденциальность, целостность данных, аутентификация исходных данных и невозможность отказа. **Целью PKI является предоставление доверенного и действительного ключа и управление сертификатом открытого ключа,** который необходим для аутентификации, невозможности отказа и конфиденциальности.

Архитектура PKI

PKC применяются в **процессе проверки действительности подписанных данных**. Имеется определенная специфика, относящаяся к используемому алгоритму, но общий процесс выглядит следующим образом (заметим, что не существует особенностей, относящихся к порядку, в котором должны выполняться перечисленные проверки; разработчики свободны в выборе наиболее эффективного способа для своих систем).

- Получатель подписанных данных должен убедиться, что предоставленная идентификация пользователя соответствует идентификации, содержащейся в PKC.
- Получатель проверяет, не отменен ли в полученной цепочке сертификатов какой-либо PKC (например, посредством получения соответствующего текущего CRL или on-line запроса статуса сертификата) и все ли PKC находились в рамках своих периодов действительности в момент подписывания данных.
- Получатель должен убедиться, что данные не содержат никаких значений, для которых подписывающая сторона не имеет авторизации.
- Получатель проверяет, не изменялись ли данные после подписывания, используя открытый ключ в PKC.
- Если все эти проверки проходят, получатель может считать, что данные были подписаны указанной подписывающей стороной. Процесс для ключей, используемых для шифрования, аналогичен.

Архитектура PKI



ASN.1: спецификация базовой нотации

- В этой лекции мы кратко рассмотрим стандартную нотацию для определения типов и значений данных – **Abstract Syntax Notation One (ASN.1).** Значение данных является экземпляром определенного типа. Стандарт ASN.1 определяет несколько базовых типов и синтаксис соответствующих им значений, а также правила для получения составных типов и значений.
- При описании протоколов взаимодействия или систем, которые совместно используют определенные структуры данных, требуется определить типы данных, передаваемые этими протоколами или совместно используемые различными системами. Для того чтобы определить эти типы данных, требуется специальная нотация. Такой нотацией является ASN.1.
- Данная нотация, с одной стороны, интуитивно понятна, а с другой стороны, может использоваться как протоколами, так и программными системами. Неотъемлемой частью ASN.1 являются базовые правила представления BER (Basic Encoding Rules). BER описывает принцип представления любой величины в рамках стандарта ASN.1. Практически все величины представляются в виде последовательности 8-битных октетов. Восьмой бит октета считается самым старшим. BER позволяет представить величину в виде последовательности 8-битных октетов несколькими способами. Имеется также поднабор правил представления DER (Distinguished Encoding Rules), который определяют однозначные способы представления величин в ASN.1.

ASN.1: спецификация базовой нотации

- Ниже приведены базовые правила обозначений в ASN.1. Все нотации ASN.1 будут печататься шрифтом **Courier New**.
- В ASN.1 типы и значения выражаются в нотации, близкой к используемой в языках программирования. Множественные пробелы и разрывы строк рассматриваются как один пробел. Комментарий может располагаться как на одной строке (в этом случае он начинается с пары символов -- и заканчивается концом строки), так и на нескольких строках (в этом случае он начинается с /* и заканчивается */). Идентификаторы (имена значений и полей) и имена типов состоят из букв, цифр и пробелов. Идентификаторы начинаются со строчной буквы, а имена типов – с прописной. В ASN.1 используются следующие обозначения:

[] квадратные скобки указывают на то, что терм является необязательным.

{} фигурные скобки группируют родственные термы.

| вертикальная черта выделяет альтернативные значения.

… многоточие обозначает повторения.

= знак равенства описывает терм как подтерм.

ASN.1: спецификация базовой нотации

- ASN.1 определяет следующие разновидности типов: простые типы, не имеющие компонентов, структурные типы, имеющие компоненты, помеченные (тегированные - tagged) типы, которые получаются из других типов добавлением метки (тега), а также такие типы, как CHOICE, ANY и некоторые другие. Типам и значениям могут присваиваться имена с помощью оператора присваивания «::=». Эти имена в дальнейшем могут использоваться для определения других типов и значений.
- Определены следующие простые типы:

INTEGER	Любое целое число
BIT STRING	Произвольная строка бит
OCTET STRING	Произвольная последовательность октетов
NULL	0
OBJECT IDENTIFIER	Последовательность компонентов, однозначно идентифицирующих объект
PrintableString	Последовательность печатных символов
IA5String	Произвольная строка символов IA5 (ASCII)
UTCTime	Универсальное время (по Гринвичу; GMT)

ASN.1: спецификация базовой нотации

- Для строчных типов может быть введено ограничение на максимальный размер.
- В ASN.1 определено четыре структурных типа:

SEQUENCE Упорядоченный набор из одного или более типов, некоторые из которых могут быть объявлены как необязательные.

SEQUENCE OF Упорядоченный набор из нуля или более значений данного типа.

SET Неупорядоченный набор из одного или более типов, некоторые из которых могут быть объявлены как необязательные.

SET OF Неупорядоченный набор из нуля или более значений данного типа.

ASN.1: спецификация базовой нотации

- Структурные типы могут иметь **необязательные компоненты**, в том числе **со значениями по умолчанию**.
- Типы могут быть помечены явно или неявно. Неявно помеченные типы получаются из других типов путем изменения метки. Для неявной пометки используется ключевое слово **IMPLICIT**. Явно помеченные типы получаются из других типов путем добавления внешней метки. Для явной пометки используется ключевое слово **EXPLICIT**. Помеченный явно тип – это структурный тип, состоящий из одного существующего типа и тега. Пометка (тегирование) весьма удобна, чтобы различать типы в пределах одного приложения.

LDAP - Lightweight Directory Access Protocol

- CCITT (Consultative Committee for International Telegraphy and Telephony) разработал серию рекомендаций для создания так называемого сервиса Директории. Директория является сервером или распределенным набором серверов, которые поддерживают распределенную базу данных, содержащую информации о различных субъектах, таких как пользователи, устройства и т.п. Эта распределенная база данных называется Информационной Базой Директории (Directory Information Base - DIB). Информация включает имя субъекта, а также различные атрибуты, характеризующие этот субъект. Данные рекомендации носят название стандарта X.500. **Первоначально LDAP начал развиваться как программный продукт переднего плана (front end) для Директории X.500.**

LDAP - Lightweight Directory Access Protocol

- LDAP предоставляет большинство возможностей **X.500** при существенно меньшей стоимости реализации. Например, удалены избыточные и редко используемые операции. **LDAP**, в отличие от X.500, использует стек **TCP**, а не OSI.
- Базовые операции протокола могут быть отображены на подмножество сервисов директории X.500. Однако не существует отображения один-к-одному между операциями протокола LDAP и операциями протокола DAP (Directory Access Protocol) стандарта X.500.
- Первая реализация LDAP написана в Мичиганском университете. Большинство ранних реализаций LDAP основано на ней.

LDAP - Lightweight Directory Access Protocol

- LDAP (Lightweight Directory Access Protocol) – это протокол, который используется для доступа к информации, хранящейся на распределенных в сети серверах.
- Эта информация представляет собой данные, хранящиеся в атрибутах. При этом предполагается, что такие данные чаще читаются, чем модифицируются. LDAP основан на клиент-серверной модели взаимодействия.
- Общая модель данного протокола состоит в том, что клиент выполняет операции протокола на серверах. Клиент передает запрос, описывающий операцию, которая должна быть выполнена сервером. Сервер выполняет необходимые операции в Директории. После завершения операции (операций) сервер возвращает клиенту ответ, содержащий результаты или ошибки.

LDAP - Lightweight Directory Access Protocol

- Информация на сервере LDAP представляет **собой совокупность записей**, которые содержат набор атрибутов и сгруппированы в **древовидную иерархическую структуру (DIT)**.
- Запись идентифицируется глобально уникальным именем (**Distinguished Name – DN**) – подобно имени домена в структуре DNS.
- DN состоит из упорядоченной (в соответствии с DIT) последовательностью RDN (**Relative Distinguished Name**).
- Значения RDN должны быть уникальны относительно родительского узла.

LDAP - Lightweight Directory Access Protocol

- Основные причины роста популярности LDAP связаны с тем, что:

– LDAP имеет **стандартную схему хранения** информации в отличие от реляционных баз данных, когда в каждом случае определяется своя схема хранения в терминах таблиц и столбцов. Поэтому в LDAP нет специфичного для каждой директории и для каждого приложения управления - нет так называемой «проблемы N+1 директории». Для всех серверов LDAP используется единая схема хранения, единый способ именования хранимых объектов и единый протокол доступа.

– LDAP позволяет **быстро искать** необходимые данные, поскольку ориентирован в большей степени на чтение и поиск информации, чем на модификацию.

– LDAP не обязательно должен быть ограничен конкретным сервером, есть возможность организовывать **распределенные системы из нескольких серверов**. В LDAP предусмотрена возможность создавать ссылки между различными серверами LDAP, что обеспечивает возможность поиска сразу на нескольких серверах LDAP.

– Как протокол LDAP, так и структура директории LDAP организованы в соответствии со стандартами, в результате чего можно **единообразно использовать реализации LDAP различных производителей**.

– Еще одно важное назначение LDAP – хранение всей информации, относящейся к PKI, а именно сертификатов, CRL и т.п.

LDAP vs базы данных

- Сравним два наиболее популярных на сегодня способа хранения информации, реляционные базы данных и серверы LDAP, по следующим характеристикам:

– **Соотношение чтение-запись** – LDAP, в отличие от реляционных баз данных, оптимизирован для чтения.

– **Расширяемость** – схемы LDAP легче изменить в процессе функционирования, чем схемы баз данных.

– **Распределенность** – данные LDAP могут располагаться на нескольких серверах, поиск на которых может осуществляться с использованием одной команды. В результате можно создавать конфигурации серверов LDAP, оптимально расположенные в зависимости от того, где требуется та или иная информация, одновременно обеспечивая возможность поиска всей информации, хранящейся на всех серверах LDAP. Тем самым достигается более высокая степень распределенности по сравнению с реляционными базами данных.

– **Репликация** – данные LDAP могут храниться на нескольких серверах, при этом существует возможность использования различных способов синхронизации информации.

LDAP vs базы данных

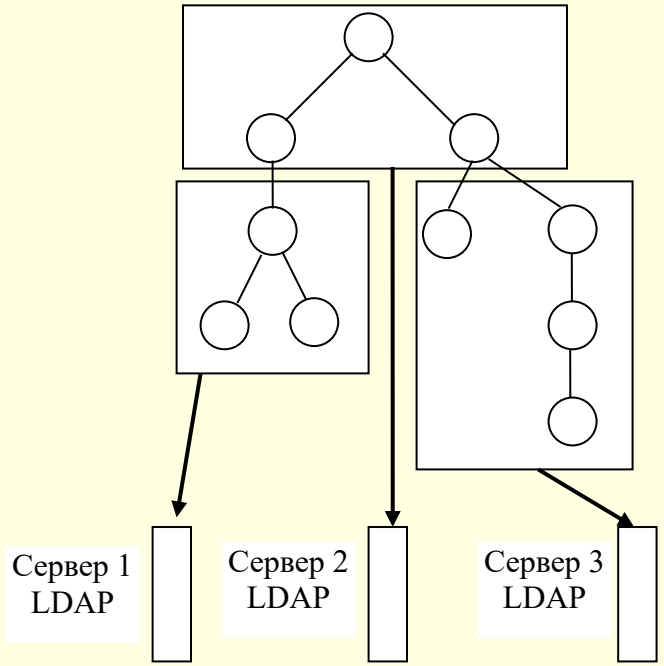
- **Применение данных** – LDAP разработан для возможности использования хранимых данных разными приложениями, базы данных разрабатываются для одного приложения.
- **Сложность взаимосвязей между объектами** - объекты баз данных имеют более сложные взаимосвязи, чем записи LDAP.
- **Транзакции** – в LDAP транзакции проще, обычно изменяется одна запись, транзакции в базах данных более сложные.
- **Тип хранимой информации** – LDAP хранит информацию в атрибутах.
- **Способ именования** – LDAP представляет собой иерархию. Имя объекта определяется путем в дереве иерархии, аналогично именованию файлов в обычных файловых системах.
- **Схемы** – схемы реляционных баз данных полностью определяются разработчиком соответствующей базы данных, LDAP имеет стандартные схемы, включая схему ядра (core), общую для любой директории. Этим достигается большая интероперабельность по сравнению с базами данных.

LDAP vs базы данных

- **Стандарты** – использование стандартных схем хранения информации и стандартного протокола доступа является преимуществом LDAP по сравнению с базами данных, так как в этом случае клиенты LDAP могут общаться с любым сервером LDAP, а клиенты баз данных могут взаимодействовать только с базой данных, для которой они разработаны.
- **Возможность отката при неудачных операциях** – реляционные базы данных имеют более гибкие возможности отката, следовательно, они больше подходят для модификации информации, чем LDAP. Для динамических объектов возможностей LDAP недостаточно.

Распределенное множество серверов LDAP

- Протокол LDAP предполагает, что существует один или несколько серверов, которые сообща предоставляют доступ к Информационному Дереву Директории (DIT).



Структура DN

Пример DN:

```
cn=Ivan Ivanov, ou=Physics Department, dc=Caltech, dc=edu
```

Обычно привязано к DNS (dc- domain controller)

Расширения сертификата

- Расширения, определенные для сертификатов **X.509 v3**, предоставляют методы для связывания дополнительных атрибутов с пользователями или открытыми ключами и для управления сертификатами. Формат сертификата X.509 v3 также допускает определение частных расширений.
- Каждое расширение в сертификате может быть либо **критичным**, либо **некритичным**. Система, использующая сертификаты, должна отвергать сертификат, если она встретила критичное расширение, которое не в состоянии распознать; однако некритичные расширения могут игнорироваться, если они не распознаются. Рассмотрим рекомендуемые в интернет расширения сертификатов. Могут использоваться дополнительные расширения; однако следует осторожно устанавливать любые критические расширения в сертификаты, так как это может препятствовать проверке действительности сертификатов.
- Каждое расширение должно иметь соответствующий OID и определяться ASN.1-структурой. Когда расширение появляется в сертификате, OID появляется как поле **`extnID`**, и соответствующая ASN.1 структура представления является значением строки октетов **`extnValue`**. Сертификат не должен включать более одного экземпляра конкретного расширения. Например, сертификат может содержать только одно расширение для идентификатора ключа уполномоченного органа. Расширение включает булево значение критичности со значением по умолчанию, равным FALSE. Для каждого расширения должны быть определены допустимые значения для поля критичности.

Расширения сертификата

- CA должны поддерживать расширения для **идентификатора ключа**, основных **ограничений использования ключа** и **политик сертификатов**. Если CA выпустил сертификаты с пустой последовательностью для поля субъекта, CA должен указать расширение альтернативного имени субъекта. Поддержка оставшихся расширений необязательна. CA могут поддерживать расширения, которые в данной спецификации не определены; при этом сертификационные центры должны учитывать, что расширения, помеченные как критичные, могут препятствовать интероперабельности.

Расширения сертификата

Стандартные расширения

Идентификатор ключа сертификационного центра

- Расширение для идентификатора ключа сертификационного центра предоставляет способ идентификации открытого ключа, соответствующего закрытому ключу, который использовался для подписывания сертификата. Данное расширение используется, когда выпускающий имеет несколько ключей для подписывания. Идентификация может быть основана либо на идентификаторе ключа (идентификатор ключа субъекта в сертификате выпускающего), либо на имени выпускающего и серийном номере.
- Поле **`keyIdentifier`** расширения **`authorityKeyIdentifier`** должно быть включено во все сертификаты, созданные CA для обеспечения возможности создания сертификационного пути.
- Существует одно исключение: когда CA распространяет свой открытый ключ в форме самоподписанного сертификата, идентификатор ключа уполномоченного органа может быть опущен. Подпись для самоподписанного сертификата создается закрытым ключом, соответствующим открытому ключу субъекта. Это доказывает, что выпускающий обладает как открытым ключом, так и закрытым. В таком случае идентификаторы субъекта и уполномоченного органа должны быть одинаковыми, и идентификатор ключа может быть указан только для открытого ключа субъекта.

Расширения сертификата

- Значение поля **keyIdentifier** должно быть получено из открытого ключа, используемого для проверки подписи сертификата, или методом, который создает уникальные значения. Рассмотрим два метода создания идентификаторов ключей из открытого ключа и один метод создания уникальных значений для **keyIdentifier**. Если идентификатор ключа не был предварительно определен, рекомендуется использовать один из этих методов для создания **keyIdentifiers**. Если идентификатор ключа был предварительно определен, CA должен задействовать предварительно определенный идентификатор.

Расширения сертификата

Идентификатор ключа субъекта

- Расширение идентификатора ключа субъекта предоставляет способ идентификации сертификата, содержащего конкретный открытый ключ.
- Для упрощения создания сертификационного пути данное расширение должно появляться во всех **сертификатах CA**, т.е. во всех сертификатах, в которых значение **`cA`** есть **TRUE**. Значение идентификатора ключа субъекта должно быть значением, указанным в поле идентификатора ключа сертификационного центра, который выпустил сертификат для данного субъекта.
- Для сертификатов CA идентификаторы ключа субъекта должны быть получены из открытого ключа или методом, который создает уникальные значения. Существует два метода для создания идентификаторов ключей из открытого ключа:

1. **`keyIdentifier`** получается из 160-битного хеша SHA-1 значения битовой строки **`subjectPublicKey`** (исключая тег, длину и неиспользуемые биты).
2. **`keyIdentifier`** получается из четырех битов поля типа со значением 0100, следующим за младшими 60 битами хеша SHA-1 значения битовой строки **`subjectPublicKey`** (исключая тег, длину и неиспользуемые биты).

Расширения сертификата

Использование ключа

- Расширение **keyUsage** определяет назначение (например, шифрование, подпись, подписывание сертификатов) ключа, содержащегося в сертификате. Ограничение использования должно применяться, когда ключ ограничивается использованием только в одной операции. Например, когда ключ RSA должен использоваться только для проверки подписей для объектов, отличных от сертификатов открытого ключа и CRL, биты **digitalSignature** и/или **nonRepudiation** должны присутствовать. Также, когда ключ RSA должен использоваться только для управления ключом, бит **keyEncipherment** должен присутствовать.
- Данное расширение должно присутствовать в сертификатах, которые содержат открытые ключи, используемые для проверки действительности цифровых подписей над другими сертификатами открытых ключей или CRL. Когда данное расширение присутствует, оно должно помечаться как критичное.

Расширения сертификата

Период использования закрытого ключа

- Данное расширение не предполагается использовать в PKI интернет. Оно скорее предназначено для использования в системах электронного документооборота, когда **требуется иметь возможность проверять подписи хранимых документов, но не подписывать новые документы закрытым ключом, срок действительности которого истекает в ближайшее время**.
- Увеличение периода использования закрытого ключа позволяет выпускающему сертификат указать период действительности закрытого ключа, отличный от периода действительности сертификата. Данное расширение предназначено для использования с ключами цифровых подписей. Расширение состоит из двух необязательных компонентов, **`notBefore`** и **`notAfter`**. Закрытый ключ, связанный с сертификатом, не должен использоваться для подписывания объектов до и после времени, указанного соответственно этими двумя компонентами. CA не должны создавать сертификаты с расширениями периода использования закрытого ключа, если, по крайней мере, один из компонентов не присутствует; расширение является некритичным.

Расширения сертификата

Политики сертификата

- Расширение политик сертификата **`certificatePolicies`** содержит последовательность из одного или более идентификаторов политики, каждый из которых может иметь квалификатор. Эти квалификаторы не должны изменять определение политики.
- В сертификате конечного участника данное расширение определяет политику, при которой был выпущен сертификат, и цели, для которых он может использоваться. В сертификате CA данное расширение ограничивает множество политик, которые могут присутствовать в сертификатах из сертификационного пути, включающего данный сертификат. Когда CA не хочет ограничивать множество политик для сертификатов из сертификационного пути, который включает данный сертификат, CA может указать специальную политику **`anyPolicy`**.
- Считается, что приложения, допускающие только определенные политики, имеют список тех политик, которые они принимают, и сравнивают OIDs политик в сертификате с этим списком. Если данное расширение критичное, ПО проверки действительности пути должно иметь возможность интерпретировать его (включая необязательный квалификатор) или ему придется отвергать сертификат.

Расширения сертификата

Отображения политик

- Данное расширение используется в сертификатах CA. Оно перечисляет одну или более пар OID; каждая пара включает **`issuerDomainPolicy`** и **`subjectDomainPolicy`**. Пара определяет, что выпустивший CA считает свою **`issuerDomainPolicy`** эквивалентной **`subjectDomainPolicy`** субъекта CA.
- Отображение политик определяет список политик, связанных с субъектом CA, которые могут приниматься как эквивалентные **`issuerDomainPolicy`**.
- Каждая **`issuerDomainPolicy`**, перечисленная в расширении отображения политик, должна также быть установлена в расширении политик сертификата в том же самом сертификате. Политики не должны отображаться в специальное значение **`anyPolicy`**.

Расширения сертификата

Альтернативное имя субъекта

Расширение альтернативных имен субъекта допускает дополнительные идентификации, связанные с субъектом сертификата. Данная опция включает e-mail адрес, DNS-имя, IP-адрес и URI. Существуют и другие формы имени, в том числе полностью локальные определения. Может быть включено несколько форм имени и несколько экземпляров для каждой из них. Если подобные идентификации необходимо ввести в сертификат, должно использоваться расширение, описывающее альтернативное имя субъекта (или альтернативное имя выпускающего); однако DNS-имя может быть представлено в поле субъекта посредством атрибута **`domainComponent`**.

Так как альтернативное имя субъекта надежно связывается с открытым ключом, считается, что все части альтернативного имени субъекта должны быть проверены CA.

Более того, если идентификация субъекта, включенная в сертификат, является только формой альтернативного имени (например, адрес электронной почты), то расширение **`subjectAltName`** должно быть помечено как критичное.

Когда расширение **`subjectAltName`** содержит электронный адрес, он должен иметь вид, определенный в RFC 822, т.е. иметь форму «**`local-part@domain`**».

Расширения сертификата

Когда расширение **`subjectAltName`** содержит **`iPAddress`**, адрес должен храниться в строке октетов в «сетевом порядке байтов», как определено в RFC 791. Для IP версии 4 строка октетов должна содержать строго четыре октета. Для IP версии 6 строка октетов должна содержать строго 16 октетов.

Альтернативные имена выпускающего

- Данное расширение используется для связывания идентификации в стиле интернет с выпускающим сертификаты. Альтернативные имена выпускающего должны иметь такое же представление, как альтернативные имена субъекта.

Расширения сертификата

Ограничения политики

- Расширение ограничений политики может использоваться в сертификатах, выпущенных CA. Расширение ограничений политики ограничивает действительность пути двумя способами. Оно может применяться для запрещения отображения политики или требования, чтобы каждый сертификат в пути содержал идентификатор принимаемой политики.
- Если поле **`inhibitPolicyMapping`** представлено, значение указывает количество дополнительных сертификатов, которые могут появиться в пути до того, как отображение политики будет запрещено. Например, это значение указывает, что отображение политики может обрабатываться в сертификатах, выпущенных субъектом данного сертификата, но не в остальных сертификатах в пути.
- Если поле **`requireExplicitPolicy`** присутствует, значение **`requireExplicitPolicy`** указывает количество дополнительных сертификатов, которые могут появиться в пути до того, как потребуется явное указание политики. Когда требуется явная политика, необходимо, чтобы все сертификаты в пути содержали идентификатор принимаемой политики. Идентификатор принимаемой политики является идентификатором политики, принимаемой пользователем, или идентификатором политики, которая объявлена эквивалентной посредством отображения политик.

Расширения сертификата

Расширенное использование ключа

- Данное расширение определяет одну или более целей, для которых может использоваться сертифицированный открытый ключ, в дополнение к основным целям, указанным в расширении **`keyUsage`**. Данное расширение появляется только в сертификатах конечного участника.
- Цели ключа могут быть при необходимости определены любой организацией.
- Данное расширение может быть как критичным, так и некритичным.
- Если расширение присутствует, то сертификат должен использоваться только для одной из указанных целей. Если указано несколько целей, то приложение не обязательно должно распознавать их все, а также расширенную цель, если она присутствует. Приложение, использующее сертификат, может потребовать указать конкретную цель, чтобы сертификат принимался данным приложением.
- Если CA включает расширенные использования ключа, чтобы соответствовать таким приложениям, но не хочет ограничивать использование ключа, он может включить специальный **`keyPurposeID anyExtendedKeyUsage`**. Если **`keyPurposeID anyExtendedKeyUsage`** присутствует, расширение не должно быть критичным.

Расширения сертификата

Точки распространения CRL

- Расширение для точек распространения CRL определяет, как может быть получена информация CRL. Расширение должно быть некритичным, но рекомендуется, чтобы оно поддерживалось как CA, так и приложениями. Далее мы подробно рассмотрим управление CRL.
- Расширение **cRLDistributionPoints** является последовательностью **DistributionPoint.** **DistributionPoint** состоит из трех полей, каждое из которых является необязательным: **distributionPoint**, **reasons** и **cRLIssuer.** Хотя каждое из этих полей является необязательным, **DistributionPoint** не должно состоять только из поля **reasons**; либо **distributionPoint**, либо **cRLIssuer** должно присутствовать. Если выпускающий сертификата не является выпускающим CRL, то поле **cRLIssuer** должно присутствовать и содержать имя выпускающего CRL. Если выпускающий сертификата является также и выпускающим CRL, то поле **cRLIssuer** должно быть опущено, и поле **distributionPoint** должно присутствовать. Если поле **distributionPoint** опущено, **cRLIssuer** должно присутствовать и включать имя, соответствующее записи директории X.500 или LDAP, в которой размещен CRL.

Расширения сертификата

- Когда поле **distributionPoint** присутствует, оно содержит либо последовательность общих имен, либо единственное значение **nameRelativeToCRLIssuer.** Если расширение **cRLDistributionPoints** содержит общее имя типа URI, предполагается следующая семантика: URI является указателем на текущий CRL для соответствующих кодов причин и выпускается соответствующим **cRLIssuer.**

Предотвращение anyPolicy

- Расширение предотвращения **anyPolicy** может быть использовано в сертификатах, выпущенных CA. Предотвращение **anyPolicy** указывает, что конкретный **anyPolicy** OID со значениями { 2 5 29 32 0 } не считается явно соответствующим остальным политикам сертификата. Значение определяет количество дополнительных сертификатов, которые могут появиться в пути до того, как **anyPolicy** будет запрещена. Например, значение, равное единице, указывает, что **anyPolicy** может быть обработана в сертификатах, выпущенных субъектом данного сертификата, но не в дополнительных сертификатах в пути.

On-line протокол запроса статуса сертификата (OCSP)

- Рассмотрим протокол, используемый для определения текущего статуса сертификата без использования CRL.
- В противоположность периодической проверке CRL иногда бывает необходимо получать текущую информацию о статусе сертификата. В качестве примера таких случаев можно привести пересылку ценных бумаг большого достоинства или масштабных фондовых продаж.
- OCSP необходим приложениям для определения состояния отмены указанного сертификата в текущий момент времени. OCSP может использоваться для обеспечения требований, касающихся получения более своевременной информации об отмене, чем это возможно с использованием CRL. Данный протокол также может применяться для получения дополнительной информации о статусе. Клиент OCSP выполняет запрос статуса и приостанавливает решение о признании сертификата действительным до тех пор, пока не получит ответ.
- Данный протокол определяет данные, которые необходимы для осуществления операций обмена между приложением, проверяющим статус сертификата, и сервером, предоставляющим этот статус.

Персональное безопасное окружение (Personal Security Environment - PSE) конечного участника

- Все конечные участники требуют локального безопасного доступа к некоторой информации – как минимум к своему собственному имени и закрытому ключу, имени CA, который является доверенным для данного участника, и открытому ключу CA (или дайджесту открытого ключа, если допустима самосертифицированная версия). Может использоваться локальное безопасное хранение и для большего количества информации (например, сертификата конечного участника или информации, относящейся к приложению). Форма хранения также может быть различной – от файлов до неподделываемых криптографических токенов. Такое локальное доверенное хранение мы будем определять как персональное безопасное окружение (**Personal Security Environment - PSE**) конечного участника.

Изменение ключа CA

- Как и в случае ключей конечных участников пара ключей CA должна регулярно изменяться; при этом требуются механизмы, позволяющие в течение некоторого времени использовать как новый, так и старый открытые ключи CA.
- Основа данной процедуры состоит в том, что CA защищает свой новый открытый ключ, используя предыдущий закрытый ключ, и наоборот. Таким образом, если CA изменяет свою пару ключей, он должен создать два сертификата **`cACertificate`**, т.е всего четыре сертификата:

`OldWithOld; OldWithNew; NewWithOld; NewWithNew`

Изменение ключа CA

- Когда CA изменяет свою пару ключей, это влияет на тех участников, которые получили старый открытый ключ внешними способами. Они должны будут получить доступ к новому открытому ключу CA, защищенному с помощью старого закрытого ключа CA. Однако это будет возможно только в течение ограниченного периода времени (до тех пор, пока они не смогут получать новый открытый ключ CA с помощью внешнего механизма). Это обычно происходит, когда сертификаты конечных участников истекают.
- Структурой данных, используемой для защиты нового и старого открытых ключей, является стандартный сертификат (который также может содержать расширения). Никаких новых структур данных не требуется.

Изменение ключа CA

- Хотя схема и может охватывать случаи, когда CA изменяет свою пару ключей более одного раза в течение периода действительности хотя бы одного из сертификатов конечного участника, это может привести к непредсказуемому результату. Чтобы этого не было, период действительности пары ключей CA должен быть больше, чем период действительности любого сертификата, выпущенного данным CA с использованием данной пары ключей.

Изменение ключа CA

Действия администратора сертификационного центра

1. Для изменения ключа CA администратор сертификационного центра должен выполнить следующее:
2. Создать новую пару ключей;
3. Создать сертификат, содержащий старый открытый ключ, подписанный новым закрытым ключом («старый с новым» сертификат);
4. Создать сертификат, содержащий новый открытый ключ CA, подписанный старым закрытым ключом («новый со старым» сертификат);
5. Создать сертификат, содержащий новый открытый ключ CA, подписанный новым закрытым ключом («новый с новым» сертификат).
6. Опубликовать эти новые сертификаты в директории и/или другими способами;
7. Экспортировать новый открытый ключ CA так, чтобы конечные участники могли получить его, используя «внешний» механизм (если нужно).

Изменение ключа CA

- Старый закрытый ключ CA более не требуется. Тем не менее старый открытый ключ еще некоторое время используется. Старый открытый ключ не будет требоваться, когда все конечные участники данного CA безопасно получат новый открытый ключ CA.
- Сертификат «старый с новым» должен иметь период действительности, начинающийся с момента создания новой пары ключей и кончающейся временем, когда все конечные участники данного CA безопасно получат новый открытый ключ CA (самое позднее – это дата истечения срока действия старого открытого ключа).

Изменение ключа CA

- Сертификат «новый с новым» должен иметь период действительности, начинающийся со времени создания новой пары ключей и заканчивающийся временем следующего изменения CA своей пары ключей.

Изменение ключа CA

	Репозиторий содержит новый и старый открытые ключи		Репозиторий содержит только старый открытый ключ (например, задержка опубликования)	
	PSE содержит новый открытый ключ	**PSE содержит старый открытый ключ**	**PSE содержит новый открытый ключ**	**PSE содержит старый открытый ключ**
Сертификат подписавшего защищен с использованием нового открытого ключа	**Случай 1:** Это стандартный случай, когда проверяющий может непосредственно проверить сертификат без использования директории	**Случай 3:** В этом случае проверяющий должен иметь доступ к директории, чтобы получить значение нового открытого ключа	**Случай 5:** Хотя оператор CA не изменил директорию, проверяющий может проверить сертификат непосредственно – аналогично случаю 1	**Случай 7:** В этом случае оператор CA не изменил директорию, и проверка не проходит
Сертификат подписавшего защищен с использованием старого открытого ключа	**Случай 2:** В этом случае проверяющий должен иметь доступ к директории, чтобы получить значение старого открытого ключа	**Случай 4:** В этом случае проверяющий может непосредственно проверить сертификат без использования репозитория	**Случай 6:** Проверяющий думает, что это ситуация случая 2 и хочет получить доступ к директории; однако проверка не проходит	**Случай 8:** Хотя оператор CA не изменил директорию, проверяющий может проверить сертификат непосредственно – случай аналогичен случаю 4