Internet Research Task Force (IRTF)
Request for Comments: 8609
Category: Experimental
ISSN: 2070-1721

M. Mosko
PARC, Inc.
I. Solis
LinkedIn
C. Wood
University of California Irvine

July 2019

Content-Centric Networking (CCNx) Messages in TLV Format

Сообщения ориентированных на содержимое сетей (CCNx) в формате TLV

Аннотация

CCNx[1] представляет собой сетевой протокол, использующий иерархические имена для пересылки запросов и сопоставления откликов с запросами. Этот документ содержит спецификацию кодирования сообщений CCNx в формате TLV, включая типы TLV, применяемые каждым элементом сообщения и кодирование каждого значения. Семантика сообщений CCNx описана в отдельной спецификации.

Документ является результатом работы исследовательской группы ICNRG[2]. Документ был рассмотрен множеством участников ICNRG. Две реализации активно применяются и показывают техническую зрелость спецификации протокола.

Статус документа

Документ не является спецификацией стандартного протокола Internet (Standards Track) и публикуется для проверки, экспериментальной реализации и оценки.

Документ определяет экспериментальный протокол для сообщества Internet. Документ является результатом работы IRTF[3]. IRTF публикует результаты связанных с Internet исследований и разработок. Эти результаты могут оказаться не подходящими для развёртывания. Данный RFC представляет согласованное мнение группы ICNRG в составе IRTF. Документ одобрен для публикации IRSG и не претендует на роль стандарта Internet (см. раздел 2 в RFC 7841).

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке https://www.rfc-editor.org/info/rfc8609.

Оглавление

[1]Content-Centric Networking - ориентированные на содержимое сети.
[2]Information-Centric Networking Research Group - группа по исследованию ориентированных на информацию сетей.
[3]Internet Research Task Force.

1. Введение

Этот документ задаёт формат пакетов TLV[1], а также кодирование типа и значения TLV для сообщений CCNx. Полное описание сетевого протокола CCNx с независимым от кодирования описанием сообщений CCNx и их элементов можно найти в [RFC8569]. Сетевой протокол CCNx использует иерархические имена для пересылки запросов и сопоставления откликов с запросами. Протокол не использует адресов конечных точек, как это делает протокол IP. Ограничения в запросе могут лимитировать отклик открытым ключом подписавшей его стороны или криптографическим хэш-значением отклика. Каждый узел пересылки CCNx на пути выполняет сопоставление имён и проверку ограничений. Протокол CCNx вписывается в более широкую модель протоколов ICN[2] [RFC7927].

Этот документ описывает схему TLV, использующую 2-байтовые поля T и L. Основания этого выбора приведены в разделе 5. Вкратце, это позволяет избежать множества вариантов кодирования одного значения (псевдонимов) и уменьшить работу при проверке пригодности для обеспечения соответствия. В отличие от некоторых других применений TLV в сетях, каждый интервал пересылки должен проверять кодирование, поэтому даже небольшая задержка на каждом узле будет приводить к значительной суммарной задержке пересылки. Для очень мелких пакетов и низкоскоростных каналов, где лишние байты могут иметь значение, можно применять протокол сжатия TLV (например, [compress] и [CCNxz]).

В документе используются термины пакет CCNx, сообщение CCNx и TLV сообщения CCNx. Пакетом CCNx называют всю дейтаграмму L3, как указано в параграфе 3.1. Сообщение CCNx - это маркер ABNF, определённый семантикой CCNx [RFC8569]. TLV сообщения CCNx говорит о кодировании сообщения CCNx в соответствии с параграфом 3.6.

Этот документ задаёт:

- формат пакетов CCNx;
- формат TLV сообщений CCNx;
- типы TLV, используемые в сообщениях CCNx;
- кодирование значений каждого типа;
- типы верхнего уровня, существующие во внешнем окружении;
- Interest TLV, существующие в Interest;
- Content Object TLV, используемые в Content Object.

Данный документ дополняется указанными ниже документами:

[1]Type-Length-Value - тип, размер, значение.
[2]Information-Centric Networking - ориентированная на информацию сеть.

- [RFC8569], описывающий семантику сообщений и протокольные операции, относящиеся к сообщениям Interest и Content Object, включая протокол Interest Return;
- [CCNxURI], описывающий нотацию CCNx URI.

Значения типов в разделе 4 соответствуют выделенным IANA номерам для протокола CCNx. В документе применяются символьные имена, определённые в указанном разделе. Все значения для типов TLV указываются относительно их родительских контейнеров. Например, каждый уровень вложенной структуры TLV может определять type = 1 со своим смыслом.

Пакеты представляются в форме 32-битовых слов. По причине вложенности TLV и наличия необязательных полей нет возможности показать все варианты. Поля на рисунках, ограниченные символами | представляют точный размер в битах. Поля с символами / имеют переменный размер и обычно дополняются до границы слова для удобства восприятия.

Документ выражает согласованные взгляды ICN RG. Это первый протокол ICN рабочей группы, созданный на базе раннего протокола CCNx [nnc] с существенным пересмотром и учётом предложения сообщества ICN и членов исследовательской группы. Документ прошёл рецензирование нескольких участников сообщества ICN и исследовательской группы. Авторы и руководитель исследовательской группы одобрили документ. Подготовка документа поддержана IRTF, документ выпущен за рамками процесса IETF и не является стандартом IETF. Это экспериментальный протокол, который может не подойти для конкретных применений. Спецификация в будущем может измениться.

1.1. Уровни требований

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с BCP 14 [RFC2119] [RFC8174] тогда и только тогда, когда они выделены шрифтом, как показано здесь.

2. Определения

Приведённые ниже определения взяты из [RFC8569]. Данный документ определяет их кодирование.

Name
: Иерархически организованный идентификатор переменного размера, представленный упорядоченным списком сегментов пути, являющихся строками октетов с переменным размером. В удобной для человека форме имена представляются в формате URI, как ccnx:/path/part. Здесь нет строк хостов и запросов. Подробности именования приведены в [CCNxURI].

Interest
: Сообщение, запрашивающее Content Object с совпадающим именем (Name) и другими необязательными селекторами для выбора из множества объектов с одинаковым именем. Любой Content Object с Name и атрибутами, соответствующими Name и необязательным селекторам в Interest считается удовлетворяющим Interest.

Content Object
: Объект данных, передаваемый в ответ на запрос Interest, имеющий необязательное имя (Name) и данные содержимого, связанные вместе криптографическими средствами.

3. Пакеты TLV

Для кодирования пакетов на основе TLV применяются 16-битовые поля Type и Length. Это обеспечивает 65536 значений типов и размер до 64 Кбайт. При 65536 возможных типах на каждом уровне кодирования TLV пространства базовых протокольных типов вполне достаточно и остаётся место для экспериментов, приложений, фирменных расширений и последующего роста. При таком кодировании не возникает больших пакетов (jumbo), размер которых превышает 64 Кбайта. При использовании в средах с поддержкой кадров jumbo можно определить агрегирование множества мелких кадров в один большой.

Таблица 1. Резервные типы TLV.

Сокращение	***Имя***	***Описание***
T_ORG	Информация производителя	Информация, относящаяся к реализации данного производителя (параграф 3.3.2).
T_PAD	Заполнение	Дополнение поля до нужного размера (параграф 3.3.1).
нет	Эксперимент	Экспериментальное использование.

Имеется несколько глобальных определений TLV, зарезервированных для всех вариантов иерархического контекста. TLV типов 0x1000 - 0x1FFF являются резервом для экспериментов (Reserved for Experimental Use), TLV типа T_ORG - для фирменных расширений (Reserved for Vendor Extensions, см. параграф 3.3.2), TLV типа T_PAD служат для заполнения полей до нужной границы.

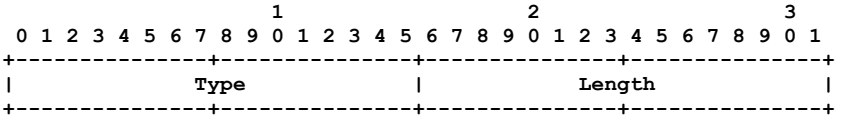

Рисунок 1. Кодирование типа и размера.

Поле Length указывает размер поля Value в октетах, не включая размер полей Type и Length. Поле Length **может** иметь значения 0.

Структуры TLV могут быть вложенными, когда поле Value одной структуры TLV включает целиком другие структуры TLV. Внешняя структура TLV называется контейнером вложенного TLV.

Значения типов зависят от контекста. Внутри контейнера TLV могут применяться ранее использованные типы для новых целей, зависящих от контекста.

3.1. Общий формат пакетов

Каждый пакет CCNx включает 8-байтовый фиксированный заголовок, описанный ниже, за которым следует набор TLV. Эти поля содержат необязательные поэтапные (hop-by-hop) заголовки и данные пакета (Packet Payload).

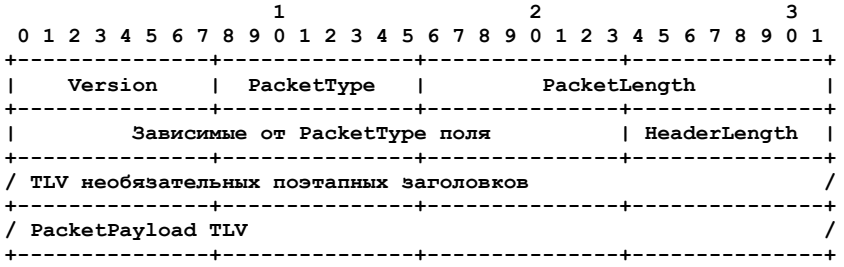

Рисунок 2. Общий формат пакета.

PacketPayload в пакете CCNx представляет собой само протокольное сообщение. Значение Content Object Hash рассчитывается только для PacketPayload без учёта фиксированных и поэтапных заголовков, которые могут меняться в пути. В подписанную информацию или хэш-значения сходства не следует включать фиксированные или поэтапные заголовки. Полю PacketPayload следует быть самодостаточным на случай удаления фиксированных и поэтапных заголовков (см. параграф 3.4.3).

Как и TLV сообщения CCNx, поле PacketPayload может включать необязательный блок Validation TLV.

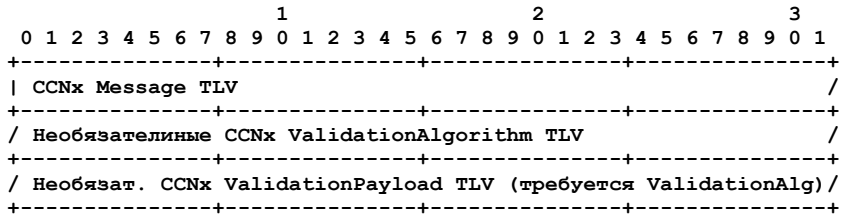

Рисунок 3. PacketPayload TLV.

После отбрасывания фиксированных и поэтапных заголовков оставшееся поле PacketPayload должно быть действительным протокольным сообщением. Поэтому PacketPayload всегда начинается с 4 байтов типа и размера, которые указывают протокольное сообщение (Interest, Content Object или иное) и его полный размер. Встраивание самодостаточного протокольного блока данных с фиксированными и поэтапными заголовками позволяет сетевому стеку отбросить заголовки и работать лишь с сообщением. Это также отвязывает поле PacketType, которое указывает, как передавать пакет, от PacketPayload.

Набор байтов, защищённых полем Validation, включает CCNx Message TLV и ValidationAlgorithm TLV.

ContentObjectHash начинается с CCNx Message TLV и завершается в конце CCNx Packet.

3.2. Фиксированные заголовки

Ниже приведены описания полей фиксированного заголовка, показанных на рисунке 2.

Version
Определяет версию пакета и **должно** иметь значение 1.

HeaderLength
Указывает размер фиксированного (8 байтов) и поэтапных заголовков. Значение **должно** быть не меньше 8.

PacketType
Описывает действия узла пересылки по отношению к пакету.

PacketLength
Общее число октетов в пакете с учётом всех заголовков и протокольного сообщения.

Зависимые от PacketType поля
Зависящие от PacketType поля пакета.

Поле PacketType указывает, как узлу пересылки следует обрабатывать пакет. Пакеты запроса (Interest) имеют PacketType = PT_INTEREST, отклики (Content Object) - PacketType = PT_CONTENT, а Interest Return - PacketType = PT_RETURN.

HeaderLength указывает число октетов от начала пакета CCNx (Version) до конца поэтапных заголовков, PacketLength - число октетов от начала до конца пакета. Оба поля имеет значение не меньше 8 (размер фиксированного заголовка).

Зависящие от PacketType поля являются резервными битами, использование которых зависит от типа пакета. Они служат для сигнализации сетевого уровня.

3.2.1. Фиксированный заголовок Interest

Если PacketType = PT_INTEREST, это говорит о том, что пакет следует пересылать в соответствии с конвейером Interest, описанным в параграфе 2.4.4 [RFC8569].Для этого типа пакетов фиксированный заголовок включает поле HopLimit, а также поля Reserved и Flags. Поле Reserved в сообщениях Interest **должно** иметь значение 0. Флаги в настоящее время не определены и поле Flags **должно** иметь значение 0.

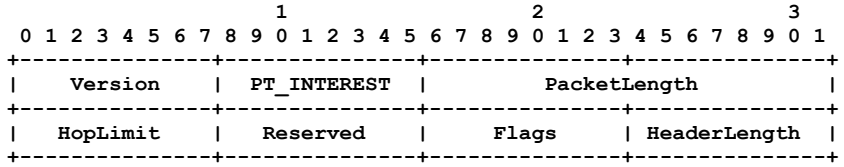

Рисунок 4. Заголовок Interest.

3.2.1.1. Interest HopLimit

Для сообщений Interest поле HopLimit содержит счётчик, декрементируемый на каждом интервале пересылки (hop). Оно ограничивает дальность перемещения Interest через сеть. Создавший Interest узел **может** поместить в это поле любое значение до 255. Каждый узел, принявший Interest с HopLimit декрементирует значение поля при получении. Если после декремента значение становится 0, Interest **недопустимо** пересылать за пределы узла.

Приём от удалённого узла сообщений Interest с HopLimit = 0 является ошибкой.

3.2.2. Фиксированный заголовок Content Object

Если PacketType = PT_CONTENT, это указывает, что пакет следует пересылать в соответствии с конвейером Content Object, определенным в параграфе 2.4.4 [RFC8569]. Content Object включает поле Flags, однако флаги в настоящее время не определены и поле должно иметь значение 0.

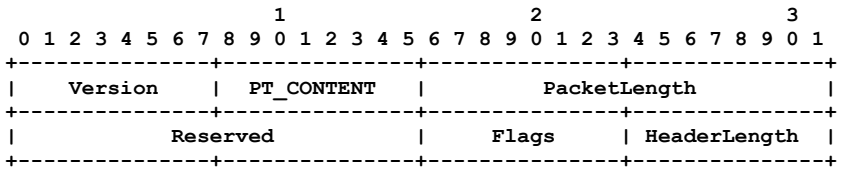

Рисунок 5. Заголовок Content Object.

3.2.3. Фиксированный заголовок Interest Return

Если PacketType = PT_RETURN, это указывает, что пакет следует обрабатывать в соответствии с правилами Interest Return, указанными в разделе 10 [RFC8569]. Единственным различием сообщения Interest Return и исходного Interest является замена PacketType на PT_RETURN и включение ReturnCode в поле ReturnCode. Все прочие поля сохраняются неизменными из пакета Interest. Целью этого заключается в предотвращении смены размера, чтобы не нужно было добавлять байты для возврата Interest на предыдущий интервал.

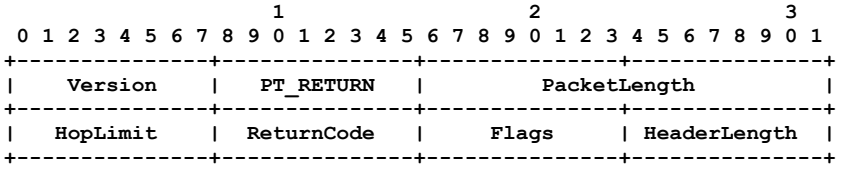

Рисунок 6. Заголовок Interest Return.

3.2.3.1. Interest Return HopLimit

Это поле HopLimit из сообщения Interest до декрементирования его узлом, передающим Interest Return.

3.2.3.2. Флаги Interest Return

Поле Flags из сообщения Interest.

3.2.3.3. Коды возврата

В этом параграфе имена кодов возврата из [RFC8569] отображены на символические имена TLV. В параграфе 4.2 символические имена отображены на числовые значения. Это поле устанавливается узлом, создающим Interest Return.

Код возврата 0 **недопустимо** устанавливать, поскольку он показывает, что возвращающая сообщение система не изменила поле Return Code.

Таблица 2. Коды возврата.

ReturnType	***Имя в 8569***
T_RETURN_NO_ROUTE	No Route
T_RETURN_LIMIT_EXCEEDED	Hop Limit Exceeded
T_RETURN_NO_RESOURCES	No Resources
T_RETURN_PATH_ERROR	Path Error
T_RETURN_PROHIBITED	Prohibited
T_RETURN_CONGESTED	Congested
T_RETURN_MTU_TOO_LARGE	MTU too large
T_RETURN_UNSUPPORTED_HASH_RESTRICTION	Unsupported ContentObjectHashRestriction
T_RETURN_MALFORMED_INTEREST	Malformed Interest

3.3. Глобальные форматы

В этом параграфе определены глобальные форматы, которые могут быть встроены в другие TLV.

3.3.1. Заполнение

Тип pad может применяться отправителями, которые хотят выравнивать данные по границе слова. Дополнение 4-байтовых слов может иметь размер 1, 2 или 3 байта, заполнение 8-байтовых слов - 0, 1, 2, 3, 5, 6 или 7 байтов.

Недопустимо применять заполнение в поле Name. Заполнение **можно** включать в любые другие TLV в CCNx Message TLV или ValidationAlgorithm TLV. В оставшейся части документа не будут указываться необязательные Pad TLV.

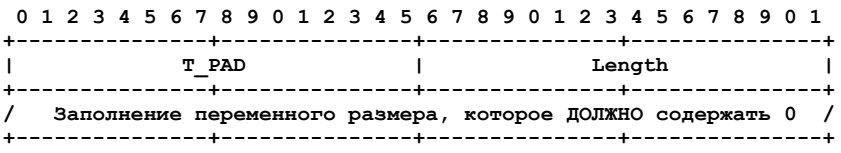

Рисунок 7. Заполнение.

3.3.2. Фирменные TLV

Специфичные для организация TLV (фирменные) **должны** использовать тип T_ORG. Поле Length указывает размер фирменной информации + 3. Поле Value начинается с 3-байтового номера организации, выведенного из значения в реестре IANA Private Enterprise Numbers [IANA-PEN] с сетевым порядком байтов, а за ним следуют специфичные для организации данные.

T_ORG MAY может служит сегментом пути в Name, которые трактуется как все прочие сегменты пути.

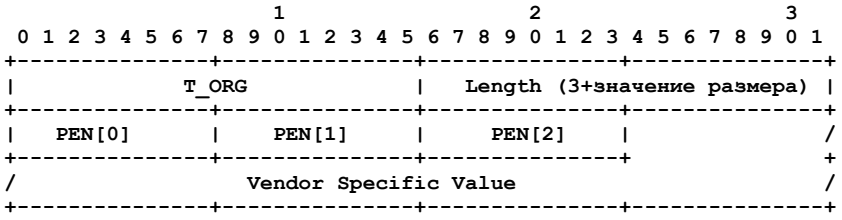

Рисунок 8. Organization-Specific TLV.

3.3.3. Формат хэш-значений

Хэш-значения применяются в нескольких полях пакетов. В такие поля обычно встраиваются TLV для указания используемой хэш-функции и её значения. Отметим, что здесь также имеются резервные типы TLV для определяемых пользователями экспериментальных функций.

Поле LENGTH хэш-значения **должно** быть не больше размера выхода хэш-функции. Если LENGTH меньше размера выходного значения функции, из этого значения берутся LENGTH байтов слева. Разрешены только заданные, а не произвольные отсечки значений.

Применение вложенного формата обусловлено тем, что он позволяет выполнять двоичное сравнение хэш-значений некоторых полей без необходимости понимания маршрутизатором новой хэш-функции. Например, поле KeyIdRestriction в Interest сравнивается побитово с полем KeyId в ContentObject. Этот формат означает, что значения внешнего поля не меняются при разных хэш-функциях, поэтому маршрутизатор может идентифицировать эти поля и выполнить двоичное сравнение хэш-TLV, не зная конкретной функции. Другой подход (например, использование T_KEYID_SHA512-256) будет требовать от маршрутизатора обновлять анализатор и поддерживать определяемые пользователями хэш-функции, что связано со значительными издержками при анализе.

Элементы CCNx **должны** поддерживать хэширование T_SHA-256 и **могут** поддерживать другие типы.

Таблица 3. Хэш-функции CCNx.

Сокращение	***Размер в октетах***
T_SHA-256	32
T_SHA-512	64, 32
нет	Экспериментальные типы TLV

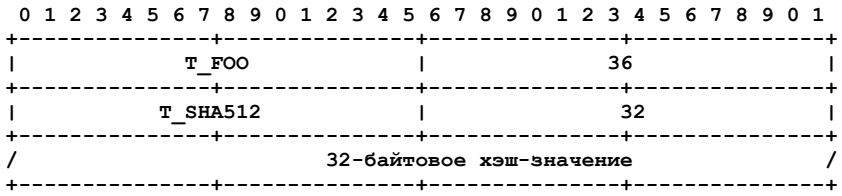

Рисунок 9. Пример вложенности в тип T_FOO.

3.3.4. Link

Link представляет собой кортеж {Name, [KeyIdRestr], [ContentObjectHashRestr]}. Это базовая форма кодирования, применяемая в данных Content Object с PayloadType = Link и поле KeyLink объектов Content. Link по сути является телом Interest.

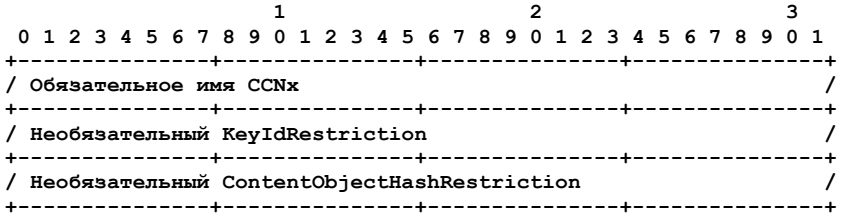

Рисунок 10. Кодирование Link.

3.4. TLV поэтапных заголовков

TLV поэтапных (hop-by-hop) заголовков являются неупорядоченными и упорядочивать их **недопустимо**. В документе определены три заголовка hop-by-hop, показанных в таблице.

Таблица 4. Типы поэтапных заголовков.

Сокращение	***Имя***	***Описание***
T_INTLIFE	Interest Lifetime (параграф 3.4.1)	Время, в течение которого сообщению Interest можно оставаться ожидающим на промежуточном узле.
T_CACHETIME	Recommended Cache Time (параграф 3.4.2)	Рекомендуемое время кэширования для Content Object.
T_MSGHASH	Message Hash (параграф 3.4.3)	Криптографическая хэш-функция (параграф 3.3.3).

Дополнительные поэтапные заголовки определены в спецификациях верхнего уровня, таких как фрагментирование.

3.4.1. Срок действия Interest

Поле Interest Lifetime указывает время, в течение которого Interest следует сохраняться в состоянии ожидания на промежуточном узле, указанное в миллисекундах целым числом без знака с сетевым порядком байтов.

Значение 0 (кодируется одним байтом 0x00) указывает, что сообщение Interest не вызвало получения отклика Content Object. Сообщение все ещё может пересылаться, но отклика не ожидается и узел пересылки может не создавать запись PIT.

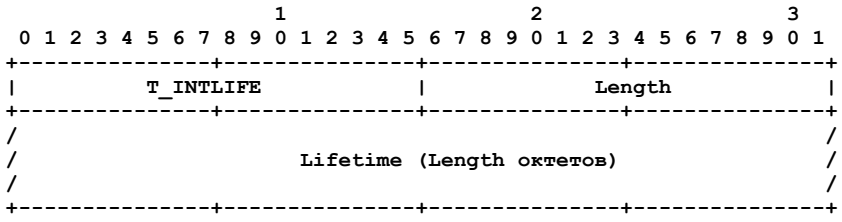

Рисунок 11. Кодирование Interest Lifetime.

3.4.2. Рекомендуемое время кэширования

RCT[1] указывает срок действия Content Object, назначенный издателем содержимого или восходящим узлом. Это значение служит рекомендацией для хранилищ Content Store при определении срока кэширования объектов Content. Кэш имеет право игнорировать эти рекомендации. Это отличается от времени ExpiryTime (параграф 3.6.2.2.2), которое имеет преимущество перед RCT и должно соблюдаться.

Поскольку RCT является необязательным поэтапным заголовком, а не частью подписанного сообщения, издатель может заново выпустить подписанный Content Object с обновлённым RCT без необходимости смены подписи сообщения. Замена RCT атакующим не оказывает существенного вреда, поскольку RCT служит лишь рекомендацией.

RCT (временная метка в миллисекундах с начала эпохи в UTC) является 64-битовым целым числом без знака с сетевым порядком байтов, которое указывает время завершения срока действия данных UTC).

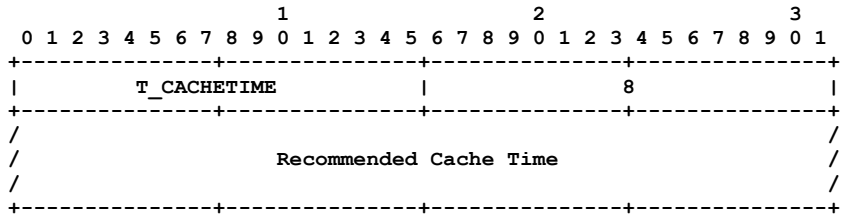

Рисунок 12. Кодирование рекомендуемого времени кэширования.

3.4.3. Хэш сообщения

Внутри доверенного домена оператор может рассчитать хэш сообщения на граничном устройстве и поместить его в поэтапные заголовки сообщения. Выходному устройству следует это значение удалить. Такой подход позволит промежуточным устройствам доверенного домена сопоставлять ContentObjectHashRestriction без расчётов на каждом узле.

Криптографическое хэш-значение сообщения учитывает данные от начала CCNx Message TLV до конца пакета. Оно служит для сопоставления с ContentObjectHashRestriction (параграф 3.6.2.1.2). Message Hash может превышать по размеру ограничение Interest и в этом случае устройству следует принимать расположенные слева байты Message Hash для сравнения с Interest.

Message Hash может содержать только одно хэш-значение и заголовок Message Hash может быть лишь один.

Заголовок Message Hash не защищён, поэтому практический смысл имеет лишь в доверенном домене, таком как автономная система оператора.

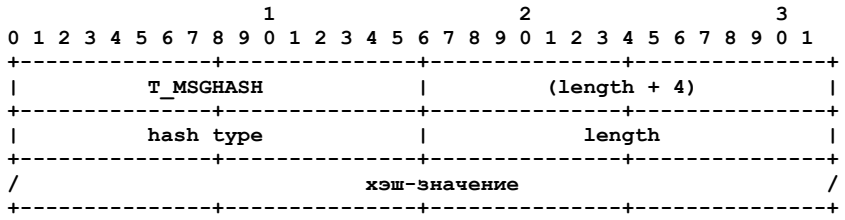

Рисунок 13. Заголовок Message Hash.

3.5. Типы верхнего уровня

Типы TLV верхнего уровня, показанные в таблице, размещаются на внешнем уровне CCNx Message TLV.

Таблица 5. Типы верхнего уровня CCNx.

Сокращение	*Имя*	*Описание*
T_INTEREST	Interest (параграф 3.6)	Сообщение типа Interest.
T_OBJECT	Content Object (параграф 3.6)	Сообщение типа Content Object.
T_VALIDATION_ALG	Validation Algorithm (параграф 3.6.4.1)	Метод проверки сообщения, такой как MIC[2], MAC[3] или криптографическая подпись.
T_VALIDATION_PAYLOAD	Validation Payload (параграф 3.6.4.2)	Выход проверки, такой как код CRC32C или подпись RSA.

[1]Recommended Cache Time - рекомендуемое время кэширования.
[2]Message Integrity Check - код целостности сообщения.
[3]Message Authentication Code - код аутентификации сообщения.

3.6. TLV сообщений CCNx

Здесь описан формат самого сообщения CCNx. TLV сообщения CCNx является частью пакета CCNx между поэтапными заголовками и Validation TLV. На рисунке показано расширение CCNx Message TLV, указанное в начале раздела 3. CCNx Message TLV начинается с MessageType и тянется вплоть до необязательного Payload. Один базовый формат применяется для сообщений Interest и Content Object, различающихся лишь полями MessageType. Первым TLV в CCNx Message TLV всегда является Name TLV, если имя имеется. Далее следуют необязательные Message TLV и необязательный Payload TLV.

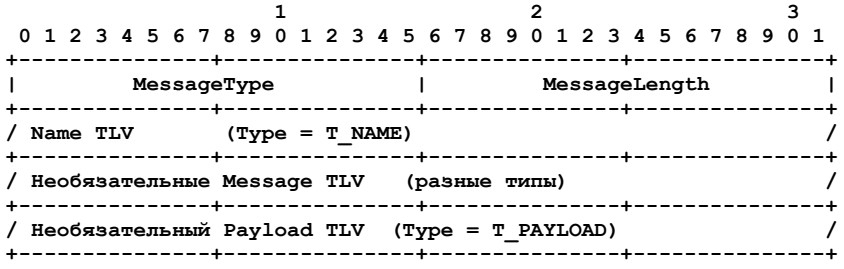

Рисунок 14. Кодирование TLV сообщения CCNx.

Таблица 6. Типы TLV сообщений CCNx.

Сокращение	*Имя*	*Описание*
T_NAME	Name (параграф 3.6.1)	Имя CCNx Name запрошенное в Interest или опубликованное в Content Object.
T_PAYLOAD	Payload (параграф 3.6.3)	Данные сообщения.

3.6.1. Name

Name представляет собой TLV, кодированный последовательностью сегментов. В таблице 7 указаны типы сегментов имён. В Name **недопустимо** включать Pad TLV.

Как указано в семантике CCNx [RFC8569], при использовании нотации CCNx URI [CCNxURI] T_NAME нулевого размера соответствует ccnx:/ (маршрут по умолчанию). Грамматика сообщений не позволяет иметь нулевой размер первому сегменту в CCNx Message TLV Name. При кодировании TLV значение ccnx:/ соответствует T_NAME размера 0.

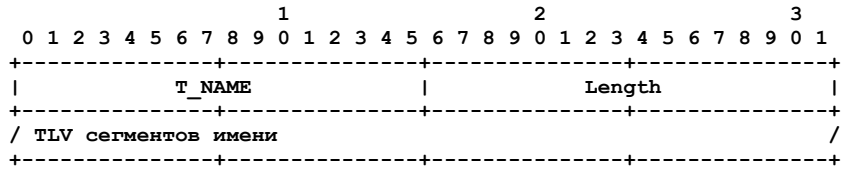

Рисунок 15. Кодирование Name.

Таблица 7. Типы имён CCNx.

Символическое имя	*Имя*	*Описание*
T_NAMESEGMENT	Name segment (параграф 3.6.1.1)	Базовый сегмент имени.
T_IPID	Interest Payload ID (параграф 3.6.1.2)	Идентификатор, представляющий поле Interest Payload. Например, Payload ID может быть хэш-значением Interest Payload. Это позволяет разделять сообщения Interest на основе их данных без анализа всех байтов данных лишь по сегменту имени Payload ID.
T_APP:00 - T_APP:4096	Application Components (параграф 3.6.1.1)	Определяемые приложением данные в сегменте имени. Приложение может использовать свою семантику для 4096 зарезервированных типов.

3.6.1.1. Сегменты имён

Выделено 4096 типа сегментов имён приложений, семантика которых задаётся приложениями. Хорошим тоном является указание приложения в имени до использования других сегментов.

На рисунке 16 показано возможное представление имени ccnx:/foo/bar/hi.

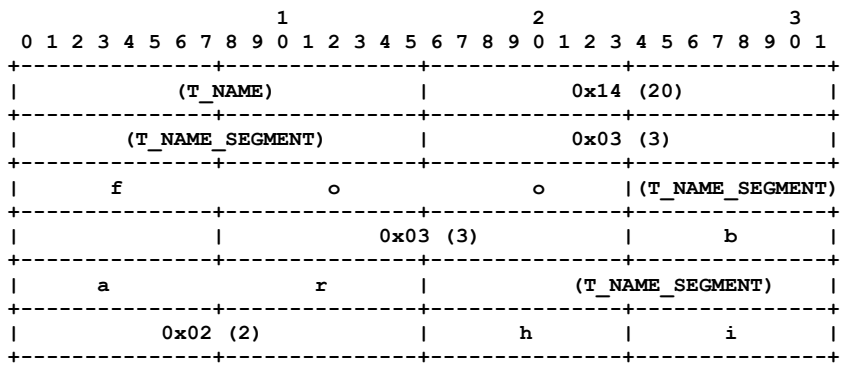

Рисунок 16. Пример кодирования имени.

3.6.1.2. Идентификатор данных Interest

InterestPayloadID - это сегмент имени, создаваемый источником Interest для представления Interest Payload. Это позволяет мультиплексировать Interest на основе имён, если они различаются содержимым. Базовым вариантом является использование хэш-значений Interest Payload в качестве InterestPayloadID.

Как часть поля Value в TLV значение InterestPayloadID содержит 1-октетный идентификатор метода, использованного для создания InterestPayloadID, за которым следует строка октетов переменного размера. От реализаций не требуется поддержка какого-либо метода для приёма Interest, поле InterestPayloadID можно считать просто строкой октетов для мультиплексирования Interest с разными данными. Алгоритмы требуется реализовать лишь на устройствах, создающих и проверяющих сегмент имени InterestPayloadID.

В поле применяется кодирование хэш-значений, описанное в параграфе 3.3.3.

При нормальной работе рекомендуется отображать InterestPayloadID просто как строку октетов в CCNx URI.

В InterestPayloadID (даже если это хэш) не следует передавать какой-либо контекст защиты. Если система требует подтверждения события, связанного с созданием Interest Payload, ей следует применять криптографическую подпись Interest в ValidationAlgorithm и ValidationPayload или использовать свой метод внутри Interest Payload.

3.6.2. TLV сообщений

С каждым типом сообщений (Interest и Content Object) связан набор Message TLV. Для создания дополнительных типов могут быть подготовлены другие спецификации.

3.6.2.1. TLV сообщений Interest

В настоящее время имеется два Message TLV, связанных с сообщениями Interest, - селектор KeyIdRestriction и селектор ContentObjectHashRestr, используемые для сужения пространства подходящих Content Object, которые удовлетворяют запросу Interest.

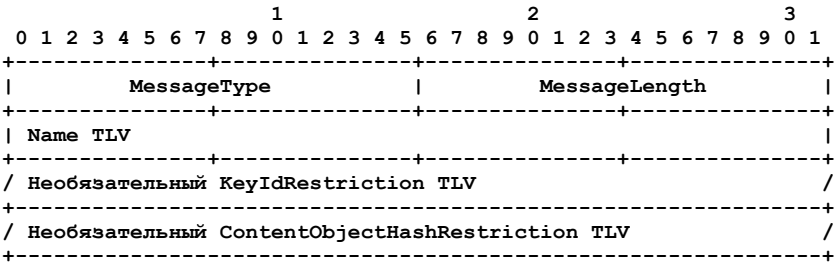

Рисунок 17. Interest Message TLV.

Таблица 8. Типы CCNx Interest Message TLV.

Сокращение	***Имя***	***Описание***
T_KEYIDRESTR	KeyIdRestriction (параграф 3.6.2.1.1)	Представление KeyId (параграф 3.3.3).
T_OBJHASHRESTR	ContentObjectHashRestriction (параграф 3.6.2.1.2)	Представление хэш-значения конкретного Content Object, который удовлетворяет Interest (параграф 3.3.3).

3.6.2.1.1. KeyIdRestriction

Сообщение Interest **может** включать селектор KeyIdRestriction, который задаёт соответствие Interest лишь объектов Content с совпадающими KeyId (см. параграф 3.6.4.1.4.1).

3.6.2.1.2. ContentObjectHashRestriction

Сообщение Interest **может** включать селектор ContentObjectHashRestriction. Это хэш-значение Content Object - самосертифицированное ограничение имени, которое должно проверяться в сети, если Interest содержит это расширение (см. параграф 3.4.3). Поле LENGTH **должно** иметь одно из разрешённых для этого хэширования значений (см. параграф 3.3.3).

ContentObjectHashRestriction **следует** иметь тип T_SHA-256 и размер 32 байта.

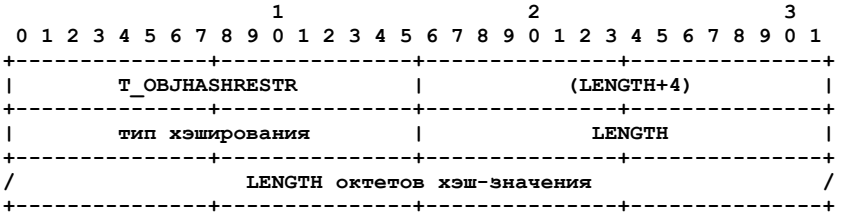

Рисунок 18. Кодирование ContentObjectHashRestriction.

3.6.2.2. TLV сообщений Content Object

Для сообщений Content Object в настоящее время определены 2 необязательных TLV - PayloadType и ExpiryTime.

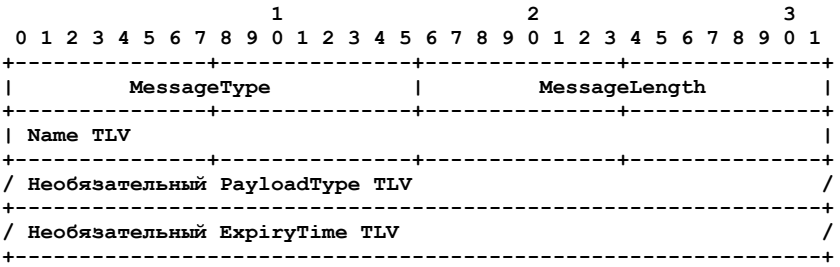

Рисунок 19. Content Object Message TLV.

Таблица 9. Типы CCNx Content Object Message TLV.

Сокращение	*Имя*	*Описание*
T_PAYLDTYPE	PayloadType (параграф 3.6.2.2.1)	Указывает тип содержимого Payload.
T_EXPIRY	ExpiryTime (параграф 3.6.2.2.2)	Время завершения срока действия Payload, указанное числом миллисекунд с начала эпохи в UTC. При отсутствии этого значения срок действия Content Object не ограничен.

3.6.2.2.1. PayloadType

Октет PayloadType представляет базовый тип Payload TLV.

T_PAYLOADTYPE_DATA
Data (возможно шифрование)
T_PAYLOADTYPE_KEY
Key
T_PAYLOADTYPE_LINK
Link

Тип Data указывает, что Payload в сообщении ContentObject содержит неанализируемые байты приложения. Тип Key показывает, что Payload содержит открытый ключ в DER-представлении. Тип Link указывает, что Payload содержит один или несколько каналов (параграф 3.3.4). При отсутствии поля предполагается тип Data.

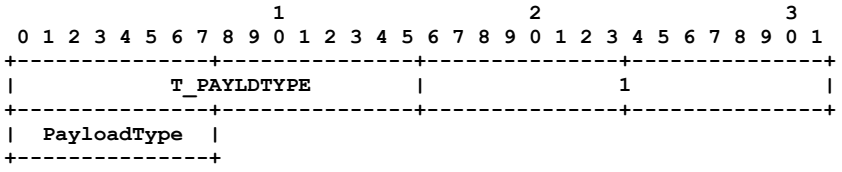

Рисунок 20. Кодирование PayloadType.

3.6.2.2.2. ExpiryTime

ExpiryTime показывает время окончания срока действия Payload, выраженное числом миллисекунд с начала эпохи в UTC (64-битовое целое число без знака с сетевым порядком байтов). Системе кэширования или конечной системе не следует отдавать отклики с просроченными Content Object (ExpiryTime в прошлом). Маршрутизаторы, пересылающие Content Object, не обязаны проверять ExpiryTime. При отсутствии ExpiryTime срок действия Content Object считается неограниченным, поэтому системы кэширования и конечные системы могут использовать его неограниченно долго.

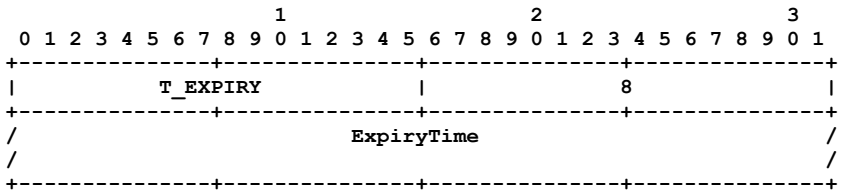

Рисунок 21. Кодирование ExpiryTime.

3.6.3. Payload

В Payload TLV передаётся содержимое пакетов, которое **может** иметь нулевой размер. Если пакет не содержит данных, это поле **следует** опускать, не помещая данные с размером 0.

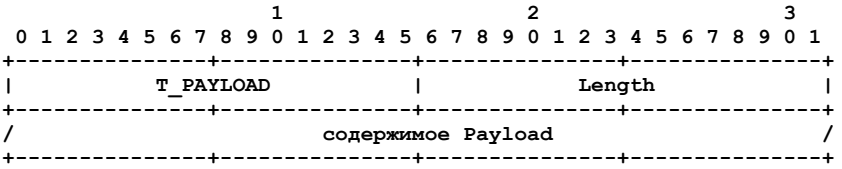

Рисунок 22. Кодирование Payload.

3.6.4. Validation

Сообщения Interest и Content Object могут включать информацию, указывающую способ проверки пригодности сообщения CCNx. Эта информация содержится в двух TLV - ValidationAlgorithm и ValidationPayload. Блок ValidationAlgorithm TLV задаёт механизм проверки сообщения CCNx (например, MIC, MAC или криптографическая подпись). ValidationPayload TLV содержит выход проверки, такой как код CRC32C или подпись RSA.

В сообщениях Interest скорей всего будет применяться только тип MIC - CRC, контрольная сумма или подпись.

3.6.4.1. Алгоритм проверки

ValidationAlgorithm - это набор вложенных TLV, которые содержат всю информацию, требуемую для проверки сообщения. Внешний контейнер имеет тип T_VALIDATION_ALG. Первый вложенный блок TLV определяет конкретный тип проверки, применяемой для сообщения. Тип задаётся полем ValidationType, как показано на рисунке 23 и описано в таблице 10. В контейнере размещаются TLV для всех зависящих от ValidationType данных (например, идентификатор ключа, его местоположение и т. п.). Примеры для нескольких типов проверок представлены в параграфе 3.6.4.1.5.

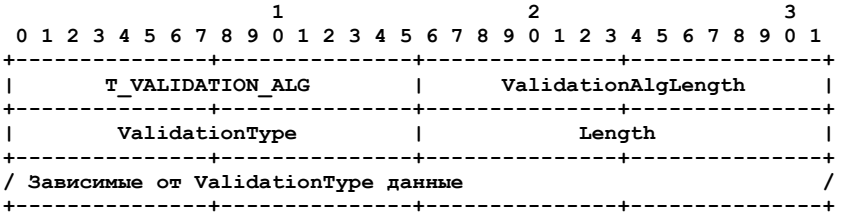

Рисунок 23. Кодирование алгоритма проверки.

Таблица 10. Типы CCNx Validation TLV.

Сокращение	*Имя*	*Описание*
T_CRC32C	CRC32C (параграф 3.6.4.1.1)	Castagnoli CRC32 (iSCSI, ext4 и т. п.) с полиномом нормальной формы 0x1EDC6F41.
T_HMAC-SHA256	HMAC-SHA256 (параграф 3.6.4.1.2)	HMAC (RFC 2104) с использованием SHA256.
T_RSA-SHA256	RSA-SHA256 (параграф 3.6.4.1.3)	Подпись RSA с открытым ключом SHA256.
T_EC-SECP-256K1	SECP-256K1 (параграф 3.6.4.1.3)	Подпись на основе эллиптической кривой с параметрами SECP-256K1 [ECC].
T_EC-SECP-384R1	SECP-384R1 (параграф 3.6.4.1.3)	Подпись на основе эллиптической кривой с параметрами SECP-384R1 [ECC].

3.6.4.1.1. Проверка целостности сообщения

Коды MIC не требуют других данных для выполнения проверки (например, CRC32C имеет значение нулевого размера).

3.6.4.1.2. Коды аутентификации сообщений

Коды MAC полезны для коммуникаций между двумя доверяющими одна другой сторонами, которые имеют общий секретный ключ. Примером может служит алгоритм HMAC. MAC использует поле KeyId для указания применяемого секрета. Назначение поля KeyId согласовано между сторонами и это может быть, например, просто номер ключа. Если новый механизм MAC требует дополнительное поле (например, вектор инициализации), это поле должно быть определено в обновлённой спецификации.

3.6.4.1.3. Подпись

Signature задаёт механизм и алгоритм подписи для проверки сообщения. Примеры включают подписи RSA с SHA256, подписи на основе эллиптической кривой с параметрами SECP-256K1 и т. п. Для проверок требуется KeyId и механизм получения открытого ключа издателя (KeyLocator), а также может применяться PublicKey, Certificate или KeyLink.

3.6.4.1.4. Зависящие от способа проверки данные

Различные алгоритмы проверки требуют доступа к разным частям данных в ValidationAlgorithm TLV. Как указано выше, Key Id, Key Locator, Public Key, Certificate, Link и Key Name играют роль в разных алгоритмах проверки. В Validation Algorithm TLV может присутствовать любой число зависящих от способа проверки контейнеров данных.

Таблица 11. Типы данных, зависящие от способа проверки.

Сокращение	*Имя*	*Описание*
T_KEYID	SignerKeyId (параграф 3.6.4.1.4.1)	Идентификатор общего секрета или открытого ключа, связанного с MAC или Signature.
T_PUBLICKEY	PublicKey (параграф 3.6.4.1.4.2)	Открытый ключ в DER-представлении.
T_CERT	Certificate (параграф 3.6.4.1.4.3)	Ключ X.509 в DER-представлении.
T_KEYLINK	KeyLink (параграф 3.6.4.1.4.4)	Объект CCNx Link.
T_SIGTIME	SignatureTime (параграф 3.6.4.1.4.5)	Временная метка (в миллисекундах) момента создания подписи.

3.6.4.1.4.1. KeyId

KeyId для подписи является идентификатором открытого ключа издателя. Это похоже на Subject Key Identifier в X.509 (см. параграф 4.2.1.2 в [RFC5280]). Значение следует выводить из ключа, использованного для подписи (например, хэш SHA-256 для ключа). Это применимо к системам как с симметричными, так и с открытыми ключами. KeyId представляется с использованием формата хэш-значений, описанного в параграфе 3.3.3. Если прикладной протокол применяет иной идентификатор, ему следует использовать зарезервированные значения.

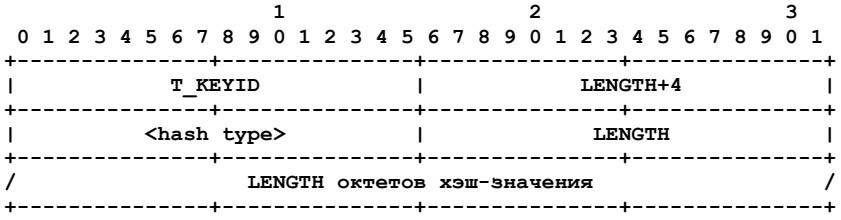

Рисунок 24. Кодирование KeyId.

3.6.4.1.4.2. Открытый ключ

Открытый ключ указывается DER-представлением блока SPKI[1] как в сертификате X.509.

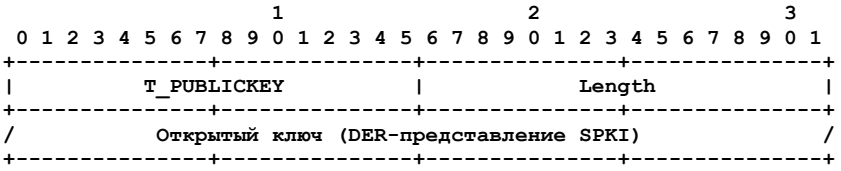

Рисунок 25. Кодирование открытого ключа.

3.6.4.1.4.3. Сертификат

Certificate указывается DER-представлением сертификата X.509, из которого выводится KeyId (параграф 3.6.4.1.4.1).

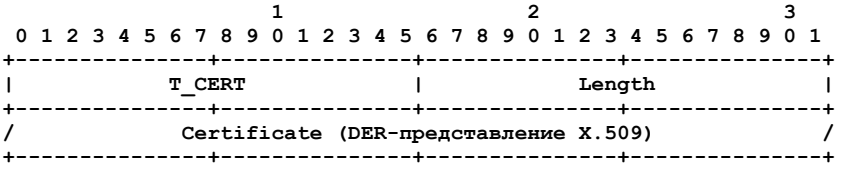

Рисунок 26. Кодирование сертификата.

[1]Subject Public Key Info - информация о субъекте открытого ключа.

3.6.4.1.4.4. KeyLink

KeyLink типа KeyLocator представляет Link.

При наличии KeyLink ContentObjectHashRestr это поле содержит дайджест Content Object, указанный KeyLink, а не подпись открытого ключа. Аналогично, KeyIdRestr в KeyLink является KeyId для ContentObject, а не обязательно «завёрнутого» ключа.

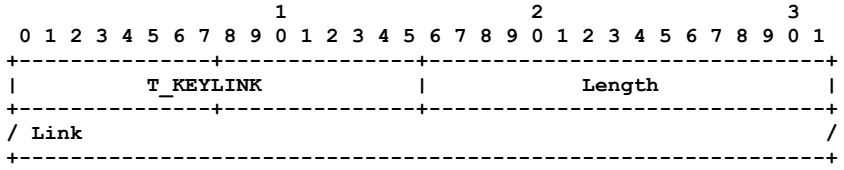

Рисунок 27. Кодирование KeyLink.

3.6.4.1.4.5. SignatureTime

SignatureTime указывает временную метку (в миллисекундах) момента создания подписи. Подписывающий устанавливает это поле в соответствии с текущим временем. Проверяющий может использовать это значение для проверки того, что подпись была создана в период действия ключа или это происходило в разумной последовательности с другими связанными подписями. Значение SignatureTime не связано с моментом создания сообщения CCNx (оно могло быть создано задолго до подписания). По умолчанию принято включать SignatureTime при создании аутентифицированных сообщений (например, HMAC или RSA).

SignatureTime является 64-битовым целым числом без знака с сетевым порядком байтов, которое указывает число миллисекунд с начала эпохи в UTC при создании подписи.

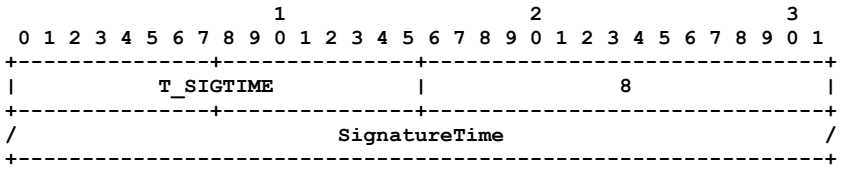

Рисунок 28. Кодирование SignatureTime.

3.6.4.1.5. Примеры проверок

Как пример проверки типа MIC кодирование CRC32C может иметь вид, представленный ниже.

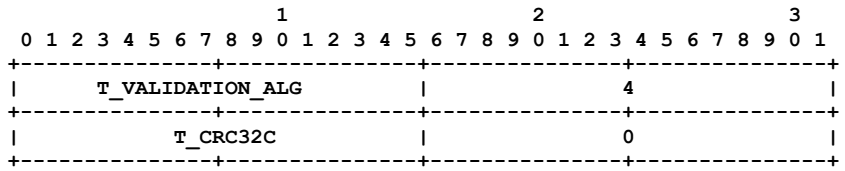

Рисунок 29. Пример кодирования CRC32C.

Как пример проверки типа MAC кодирование HMAC с использованием SHA256 может иметь вид

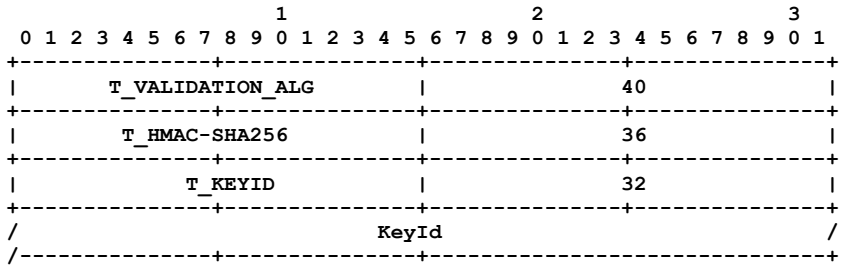

Рисунок 30. Пример кодирования HMAC-SHA256.

Как пример проверки типа Signature кодирование подписи с открытым ключом RSA при использовании подписи SHA256 и Public может иметь вид

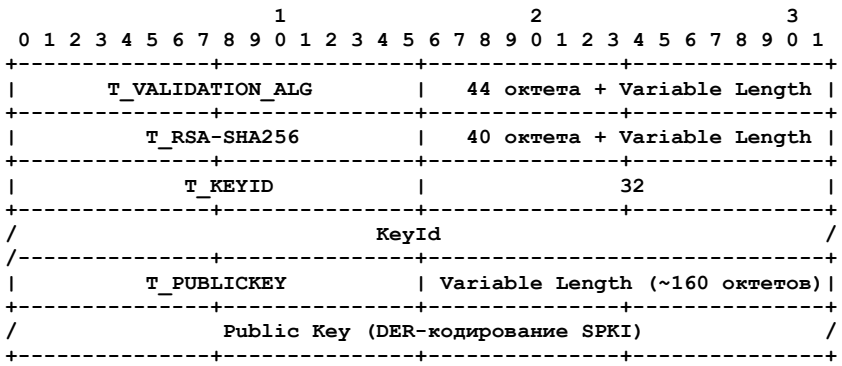

Рисунок 31. Пример кодирования RSA-SHA256.

3.6.4.2. Данные Validation

ValidationPayload содержит вывод проверки, такой как код CRC32C или подпись RSA.

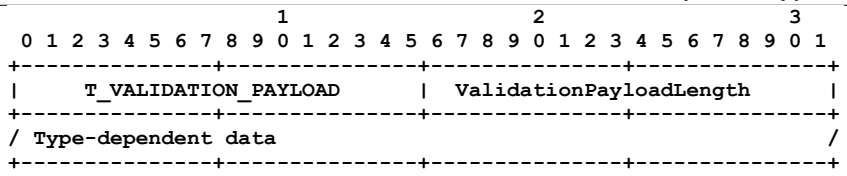

Рисунок 32. Пример кодирования данных проверки.

4. Взаимодействие с IANA

В этом разделе подробно описаны все типы значений протокола CCNx, которые могут регистрироваться. Каждый реестр может обновляться путём постепенного расширения пространства, например, за счёт выделения и резервирования новых типов. В соответствии с [RFC8126] этот раздел описывает создание реестра Content-Centric Networking (CCNx) и его субреестров.

4.1. Реестр типов пакетов

Агентство IANA создало реестр CCNx Packet Types и выделило значения для типа пакетов, приведённые ниже. Регистрация выполняется по процедуре RFC Required. Размер Type составляет 1 октет, диапазон - 0x00-0xFF.

Типы пакетов.

Тип	Имя	Документ
0x00	PT_INTEREST	Фиксированные заголовки (параграф 3.2)
0x01	PT_CONTENT	Фиксированные заголовки (параграф 3.2)
0x02	PT_RETURN	Фиксированные заголовки (параграф 3.2)

4.2. Реестр кодов Interest Return

Агентство IANA создало реестр CCNx Interest Return Code Types и выделило значения кодов, приведённые ниже. Регистрация выполняется по процедуре RFC Required. Размер Type составляет 1 октет, диапазон - 0x00-0xFF.

Типы CCNx Interest Return.

Тип	Имя	Документ
0x00	Резерв	
0x01	T_RETURN_NO_ROUTE	Фиксированные заголовки (параграф 3.2.3.3)
0x02	T_RETURN_LIMIT_EXCEEDED	Фиксированные заголовки (параграф 3.2.3.3)
0x03	T_RETURN_NO_RESOURCES	Фиксированные заголовки (параграф 3.2.3.3)
0x04	T_RETURN_PATH_ERROR	Фиксированные заголовки (параграф 3.2.3.3)
0x05	T_RETURN_PROHIBITED	Фиксированные заголовки (параграф 3.2.3.3)
0x06	T_RETURN_CONGESTED	Фиксированные заголовки (параграф 3.2.3.3)
0x07	T_RETURN_MTU_TOO_LARGE	Фиксированные заголовки (параграф 3.2.3.3)
0x08	T_RETURN_UNSUPPORTED_HASH_RESTRICTION	Фиксированные заголовки (параграф 3.2.3.3)
0x09	T_RETURN_MALFORMED_INTEREST	Фиксированные заголовки (параграф 3.2.3.3)

4.3. Реестр типов Hop-by-Hop

Агентство IANA создало реестр CCNx Hop-by-Hop Types и выделило значения кодов, приведённые ниже. Регистрация выполняется по процедуре RFC Required. Размер Type составляет 2 октета, диапазон - 0x0000-0xFFFF.

Типы CCNx Hop-by-Hop

Тип	Имя	Документ
0x0000	Резерв	
0x0001	T_INTLIFE	TLV поэтапных заголовков (параграф 3.4)
0x0002	T_CACHETIME	TLV поэтапных заголовков (параграф 3.4)
0x0003	T_MSGHASH	TLV поэтапных заголовков (параграф 3.4)
0x0004 - 0x0007	Резерв	
0x0FFE	T_PAD	Заполнение (параграф 3.3.1)
0x0FFF	T_ORG	Фирменные TLV (параграф 3.3.2)
0x1000 - 0x1FFF	Резерв	Экспериментальное использование (раздел 3)

4.4. Реестр типов верхнего уровня

Агентство IANA создало реестр CCNx Top-Level Types и выделило значения кодов, приведённые ниже. Регистрация выполняется по процедуре RFC Required. Размер Type составляет 2 октета, диапазон - 0x0000-0xFFFF.

Типы верхнего уровня CCNx

Тип	Имя	Документ
0x0000	Резерв	
0x0001	T_INTEREST	Типы верхнего уровня (параграф 3.5)
0x0002	T_OBJECT	Типы верхнего уровня (параграф 3.5)
0x0003	T_VALIDATION_ALG	Типы верхнего уровня (параграф 3.5)
0x0004	T_VALIDATION_PAYLOAD	Типы верхнего уровня (параграф 3.5)

4.5. Реестр типов сегментов имён

Агентство IANA создало реестр CCNx Name Segment Types и выделило значения кодов, приведённые ниже. Регистрация выполняется по процедуре RFC Required. Размер Type составляет 2 октета, диапазон - 0x0000-0xFFFF.

Типы сегментов имён CCNx.

Тип	Имя	Документ
0x0000	Резерв	
0x0001	T_NAMESEGMENT	Name (параграф 3.6.1)
0x0002	T_IPID	Name (параграф 3.6.1)

0x0010 - 0x0013	Резерв	RFC 8609
0x0FFF	T_ORG	Фирменные TLV (параграф 3.3.2)
0x1000 - 0x1FFF	T_APP:00 - T_APP:4096	Сегменты имён (параграф 3.6.1.1)

4.6. Реестр типов сообщений

Агентство IANA создало реестр CCNx Message Types и выделило значения кодов, приведённые ниже. Регистрация выполняется по процедуре RFC Required. Размер Type составляет 2 октета, диапазон - 0x0000-0xFFFF.

Типы сообщений CCNx.

Тип	***Имя***	***Документ***
0x0000	T_NAME	TLV сообщений CCNx (параграф 3.6)
0x0001	T_PAYLOAD	TLV сообщений CCNx (параграф 3.6)
0x0002	T_KEYIDRESTR	TLV сообщений CCNx (параграф 3.6)
0x0003	T_OBJHASHRESTR	TLV сообщений CCNx (параграф 3.6)
0x0005	T_PAYLDTYPE	TLV сообщений Content Object (параграф 3.6.2.2)
0x0006	T_EXPIRY	TLV сообщений Content Object (параграф 3.6.2.2)
0x0007 - 0x000C	Резерв	RFC 8609
0x0FFE	T_PAD	Заполнение (параграф 3.3.1)
0x0FFF	T_ORG	Фирменные TLV (параграф 3.3.2)
0x1000-0x1FFF	Резерв	Экспериментальное использование (раздел 3)

4.7. Реестр типов данных

Агентство IANA создало реестр CCNx Payload Types и выделило значения кодов, приведённые ниже. Регистрация выполняется по процедуре Specification Required. Размер Type составляет 1 октет, диапазон - 0x00-0xFF.

Типы CCNx Payload.

Тип	***Имя***	***Документ***
0x00	T_PAYLOADTYPE_DATA	PayloadType (параграф 3.6.2.2.1)
0x01	T_PAYLOADTYPE_KEY	PayloadType (параграф 3.6.2.2.1)
0x02	T_PAYLOADTYPE_LINK	PayloadType (параграф 3.6.2.2.1)

4.8. Реестр типов алгоритма проверки

Агентство IANA создало реестр CCNx Validation Algorithm Types и выделило значения кодов, приведённые ниже. Регистрация выполняется по процедуре Specification Required. Размер Type составляет 2 октета, диапазон - 0x0000-0xFFFF.

Типы алгоритмов проверки CCNx.

Тип	***Имя***	***Документ***
0x0000	Резерв	
0x0002	T_CRC32C	Алгоритм проверки (параграф 3.6.4.1)
0x0004	T_HMAC-SHA256	Алгоритм проверки (параграф 3.6.4.1)
0x0005	T_RSA-SHA256	Алгоритм проверки (параграф 3.6.4.1)
0x0006	T_EC-SECP-256K1	Алгоритм проверки (параграф 3.6.4.1)
0x0007	T_EC-SECP-384R1	Алгоритм проверки (параграф 3.6.4.1)
0x0FFE	T_PAD	Заполнение (параграф 3.3.1)
0x0FFF	T_ORG	Фирменные TLV (параграф 3.3.2)
0x1000 - 0x1FFF	Резерв	Экспериментальное использование (раздел 3)

4.9. Реестр зависимых от типа проверки данных

Агентство IANA создало реестр CCNx Validation-Dependent Data Types и выделило значения кодов, приведённые ниже. Регистрация выполняется по процедуре RFC Required. Размер Type составляет 2 октета, диапазон - 0x0000-0xFFFF.

Зависящие от способа проверки типы данных CCNx.

Тип	***Имя***	***Документ***
0x0000	Резерв	
0x0009	T_KEYID	Зависящие от способа проверки данные (параграф 3.6.4.1.4)
0x000A	T_PUBLICKEYLOC	Зависящие от способа проверки данные (параграф 3.6.4.1.4)
0x000B	T_PUBLICKEY	Зависящие от способа проверки данные (параграф 3.6.4.1.4)
0x000C	T_CERT	Зависящие от способа проверки данные (параграф 3.6.4.1.4)
0x000D	T_LINK	Зависящие от способа проверки данные (параграф 3.6.4.1.4)
0x000E	T_KEYLINK	Зависящие от способа проверки данные (параграф 3.6.4.1.4)
0x000F	T_SIGTIME	Зависящие от способа проверки данные (параграф 3.6.4.1.4)
0x0FFF	T_ORG	Фирменные TLV (параграф 3.3.2)
0x1000-0x1FFF	Резерв	Экспериментальное использование (раздел 3)

4.10. Реестр типов хэш-функций

IANA создан реестр CCNx Hash Function Types и выделило значения кодов, приведённые ниже. Регистрация выполняется по процедуре Specification Required. Размер Type составляет 2 октета, диапазон - 0x0000-0xFFFF.

Типы хэш-функций CCNx.

Тип	***Имя***	***Документ***
0x0000	Резерв	
0x0001	T_SHA-256	Формат хэш-значений (параграф 3.3.3)
0x0002	T_SHA-512	Формат хэш-значений (параграф 3.3.3)
0x0FFF	T_ORG	Фирменные TLV (параграф 3.3.2)
0x1000-0x1FFF	Резерв	Экспериментальное использование (раздел 3)

5. Вопросы безопасности

CCNx является протоколом сетевого уровня (L3), который может работать в режиме наложения с использованием такого транспорта, как UDP или туннели. Протокол имеет встроенную поддержку аутентификации сообщений с помощью подписи (например, RSA или эллиптическая кривая) или кода аутентификации (например, HMAC). Взамен аутентификатора может применяться проверка целостности сообщения (например, SHA или CRC). CCNx не определяет «конверт» шифрования, оставляя это вышележащим протоколам (например, [esic]).

Формат пакетов CCNx позволяет присоединить MIC (например, CRC32C), MAC (например, HMAC) и подписи (например, RSA или ECDSA) к пакетам любого типа. Это не означает, что хорошей идеей будет использование произвольного ValidationAlgorithm или включать ресурсоёмкие алгоритмы в пакеты Interest, поскольку это может приводить к DoS-атакам за счёт вычислений. Приложениям следует использовать явный протокол для руководства применением подписей пакетов. В качестве общего руководства приложения могут использовать MIC в сообщениях Interest для обнаружения непреднамеренно повреждённых пакетов. Если нужна защита Interest, следует рассмотреть возможность шифрования и протокол, предотвращающий атаки с повторным использованием, особенно при использовании Interest в качестве исполнительного механизма (actuator). Простое применение кода аутентификации или подписи не обеспечивает защиты Interest. В литературе имеется несколько примеров защиты обмена сообщениями в стиле ICN [mobile] [ace].

Поскольку документ относится к протоколу L3, он не описывает способов доставки ключей и механизмов доверия к ним. Content Object в CCNx может включать открытый ключ или сертификат, а также может использовать поле KeyLink для указания открытого ключа или сертификата для проверки подлинности сообщения. Одной из спецификаций обмена ключами является CCNxKE [ccnx-ke] [mobile], где обмен похож на процедуру TLS 1.3, отличаясь тем, что происходит через сообщения CCNx L3. Вопросы доверия выходят за рамки протокола L3 и решаются приложениями.

Комбинация обмена эфемерными ключами (например, CCNxKE [ccnx-ke]) с инкапсулирующим шифрованием (например, [esic]) обеспечивает эквивалент туннеля TLS. Промежуточные узлы могут пересылать сообщения Interest и Content Object, но не будут видеть их содержимого. Это также полностью скрывает внутренние имена, заменяя их именами, используемыми уровнем шифрования. Этот тип туннельного шифрования полезен для передачи содержимого, которое не мало или совсем не подходит для кэширования, поскольку оно может применяться лишь теми, кто знает эфемерный ключ. Краткосрочное кэширование может помочь на каналах с потерями, но длительное кэширование обычно не представляет интереса.

Шифрование широковещательных пакетов и перешифрование на прокси могут быть полезны для содержимого, используемого многократно в течение времени или множеством потребителей. В настоящее время нет рекомендаций по этим вариантам шифрования.

Конкретное кодирование сообщений будет влиять на безопасность. В [RFC8609] применяется кодирование TLV. Был выбран компромисс между расширяемостью и однозначностью кодирования типа и размера. Некоторые TLV используют поля T и L переменного размера, чтобы вместить более широкий диапазон значений с сохранением эффективного использования байтов. Здесь TLV кодируются с 2-байтовыми полями T и L, что решает 2 проблемы. Первая проблема связана с псевдонимами. Если можно кодировать одно значение в полях разного размера (например, %x02 и %x0002), кто-то может счесть их одним, а другой - разными. Если есть различие, оно должно определяться в буферах, а не числовом представлении. Если от этого отказаться, придётся проверять кодирование TLV в каждом поле каждого пакета на всех узлах пересылки. Если они одинаковы, возникает другая проблема - как задавать фильтры. Например, если имя включает 6 компонент, имеется 7 полей T и 7 полей L, каждое из которых может иметь до 4 представлений одного значения. В результате имеется 14 полей с 4 представлениями для каждого или «1001 комбинация». Это также означает невозможность сравнения, например, имени через функции памяти, поскольку нужно учитывать разные форматы T и L.

Сообщение Interest Return не имеет аутентификатора от предыдущего интервала. Поэтому данные из Interest Return следует применять лишь локально для сопоставления с Interest. Узлу не следует пересылать эти Interest Payload как Interest. Он должен также убедиться, что передал Interest в Interest Return нужному узлу и не позволять кому-либо отменять сообщения Interest.

Кэширующие узлы должны соблюдать осторожность при обработке Content Object. Важно, чтобы хранилище Content Store следовало правилам параграфа 2.4.3 во избежание некоторых типов атак. CCNx 1.0 не имеет механизмов обхода нежелательных результатов из сети (нет «исключений»), поэтому при отравлении кэша непригодным содержимым это может вызвать проблемы при поиске. Существует 3 типа доступа к содержимому из Content Store - неограниченный, а также ограниченные подписью и хэш-значением. Если Interest не имеет ограничений, запрашивающая сторона не заботится о том, что она получит в ответ и подойдёт любой кэшированный объект. При ограничении по хэш-значению запрашивающая сторона очень точно определяет желаемый результат и Content Store (и каждый пересылающий узел) может легко проверить соответствие содержимого запросу. При ограничении подписью (часто служит для начального обнаружения манифеста) запрашивающий знает лишь KeyId для подписи содержимого. Этот случай требует пристального внимания в Content Store, чтобы избежать усиления неверных данных. Хранилище Content Store должно отвечать лишь объектами Content с проверенной подписью. Это значит, что Content Object содержит в себе открытый ключ или Interest передаёт открытый ключ в дополнение к KeyId. Если это не выполняется, хранилищу Content Store следует рассматривать Interest как отсутствие кэша и позволять конечной точке ответить.

Кэш пользовательского уровня может выполнять полную проверку подписи путём извлечения открытого ключа или сертификата по KeyLink. Однако это не та задача, которую можно отдать узлу пересылки. Пользовательский кэш может также полагаться на внешнюю (out-of-band) аттестацию, например, если оператор кэша включает лишь информацию, для которой у него есть корректная подпись.

Грамматика CCNx обеспечивает гибкость алгоритма хэширования с помощью HashType, указывая список приемлемых алгоритмов, которые следует реализовать на каждом узле пересылки. Некоторые типы пригодны лишь для конечных систем и обновление их не влияет на узлы пересылки, которые будут просто сопоставлять буферы, включающие type-length-hash. Некоторые поля (например, ConObjHash) должны проверяться на каждом узле, поэтому узел пересылки (или связанная с ним система) должен знать алгоритм хэширования. Это может вызвать проблемы совместимости при смене типа хэша. [RFC8609] является официальным источником данных о разрешённых типах хэширования.

Для имён CCNx применяется двоичное сопоставление, тогда как для URI используются имена хостов без учёта регистра. Некоторые системы могут применять независимое от регистра сопоставление путей URI к ресурсу. В результате введённые человеком имена CCNx будут скорей всего сталкиваться с несоответствием символов разных регистров и символы не-ASCII, если не применять нормализацию URI для имён CCNx. Это означает, что объект, регистрирующий маршрутизируемый префикс CCNx, скажем ccnx:/example.com, должен зарегистрировать варианты типа ccnx:/Example.com. Если это не учтено в нормализации URI normalization и соглашениях протокола маршрутизации, становятся возможными фишинговые атаки.

Более общее рассмотрение вопросов безопасности ICN приведено в [RFC7927] и [RFC7945].

6. Литература

6.1. Нормативные документы

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

6.2. Дополнительная литература

[ace] Shang, W., Yu, Y., Liang, T., Zhang, B., and L. Zhang, "NDN-ACE: Access control for constrained environments over named data networking", NDN Technical Report NDN-0036, 2015, <http://new.named-data.net/wp-content/uploads/2015/12/ndn-0036-1-ndn-ace.pdf>.

[ccnxke] Mosko, M., Uzun, E., and C. Wood, "CCNx Key Exchange Protocol Version 1.0", Work in Progress, draft-wood-icnrg-ccnxkeyexchange-02, March 2017.

[CCNxURI] Mosko, M. and C. Wood, "The CCNx URI Scheme", Work in Progress, draft-mosko-icnrg-ccnxurischeme-01, April 2016.

[CCNxz] Mosko, M., "CCNxz TLV Header Compression Experimental Code", commit f1093a2, March 2018, <https://github.com/PARC/CCNxz>.

[compress] Mosko, M., "Header Compression for TLV-based Packets", ICNRG Interim Meeting, 2016, <https://datatracker.ietf.org/meeting/interim-2016-icnrg-02/materials/slides-interim-2016-icnrg-2-7>.

[ECC] Certicom Research, "SEC 2: Recommended Elliptic Curve Domain Parameters", 2010, <http://www.secg.org/sec2-v2.pdf>.

[esic] Mosko, M. and C. Wood, "Encrypted Sessions In CCNx (ESIC)", Work in Progress, draft-wood-icnrg-esic-01, September 2017.

[IANA-PEN] IANA, "Private Enterprise Numbers", <http://www.iana.org/assignments/enterprise-numbers>.

[mobile] Mosko, M., Uzun, E., and C. Wood, "Mobile Sessions in Content-Centric Networks", IFIP Networking, 2017, <http://dl.ifip.org/db/conf/networking/networking2017/1570334964.pdf>.

[nnc] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking Named Content", Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT '09), 2009, <http://dx.doi.org/10.1145/1658939.1658941>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlisch, "Information-Centric Networking (ICN) Research Challenges", RFC 7927, DOI 10.17487/RFC7927, July 2016, <https://www.rfc-editor.org/info/rfc7927>.

[RFC7945] Pentikousis, K., Ed., Ohlman, B., Davies, E., Spirou, S., and G. Boggia, "Information-Centric Networking: Evaluation and Security Considerations", RFC 7945, DOI 10.17487/RFC7945, September 2016, <https://www.rfc-editor.org/info/rfc7945>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8569] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Semantics", RFC 8569, DOI 10.17487/RFC8569, July 2019, <https://www.rfc-editor.org/info/rfc8569>.

Адреса авторов

Marc Mosko

PARC, Inc.

Palo Alto, California 94304

United States of America

Phone: +01 650-812-4405

Email: mmosko@parc.com

Ignacio Solis

LinkedIn

Mountain View, California 94043

United States of America

Email: nsolis@linkedin.com

Christopher A. Wood

University of California, Irvine

Irvine, California 92697

United States of America

Phone: +01 315-806-5939

Email: woodc1@uci.edu

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru