Network Working Group
Request for Comments: 5116
Category: Standards Track

D. McGrew
Cisco Systems, Inc.

January 2008

Интерфейс и алгоритмы аутентифицированного шифрования
An Interface and Algorithms for Authenticated Encryption

Статус документа

Этот документ представляет проект стандартного протокола для сообщества Internet и служит приглашением к дискуссии в целях развития и совершенствования протокола. Информацию о текущем состоянии стандартизации протокола можно найти в документе Internet Official Protocol Standards (STD 1). Настоящий документ может распространяться без ограничений.

Аннотация

Этот документ определяет алгоритмы для аутентифицированного шифрования со связанными данными (AEAD[1]), а также интерфейс и реестр для таких алгоритмов. Интерфейс и реестр могут использоваться в качестве независимого от приложений набора шифров. Такая модель обеспечивает преимущества в плане эффективности и безопасности, позволяя многократно применять криптографический код.

Оглавление

1. Введение

Аутентифицированное шифрование [BN00] в дополнение к обычной защите конфиденциальности шифруемой информации обеспечивает способ проверки её целостности и подлинности (аутентичности). Алгоритм аутентифицированного шифрования со связанными данными (AEAD) [R02] добавляет возможность проверки целостности и подлинности некоторых связанных данных (AD[2]), называемых также дополнительными аутентифицированными данными, которые не шифруются.

1.1. Предпосылки

Для многих криптографических приложений требуется обеспечение конфиденциальности и аутентификации сообщений. Защита конфиденциальности обеспечивает невозможность получения доступа к информации неуполномоченным сторонам и обычно реализуется путём шифрования. Аутентификации сообщений требуется для того, чтобы обеспечить невозможность изменения или подмены сообщения неуполномоченной стороной, и реализуется обычно за счёт использования кодов MAC[3]. Такой сервис называют также защитой целостности данных. Многие приложения используют шифрование и MAC совместно для обеспечения обоих видов защитных услуг, но в каждом алгоритме применяются независимые ключи. Сравнительно недавно возникла идея объединения защитных

[1]Authenticated Encryption with Associated Data.
[2]Associated Data.
[3]Message Authentication Code - код аутентификации сообщения.

услуг путём использования для обеих одного криптоалгоритма. В рамках этой концепции шифрование и MAC заменяются одним алгоритмом шифрования со связанными данными (AEAD).

Было определено несколько криптоалгоритмов, реализующих AEAD, включая поддержку блочного режима шифрования и выделенные алгоритмы. Некоторые из этих алгоритмов были приняты для практического применения и доказали свою полезность. Кроме того, алгоритмы AEAD близки к «идеальным» алгоритмам шифрования, используемым в автоматизированных системах криптоанализа протоколов (см., например, параграф 2.5 в [BOYD]).

Преимущества алгоритмов AEAD и их интерфейс описаны в параграфе 1.3.

1.2. Сфера действия документа

А этом документе алгоритм AEAD определяется абстрактно, путём спецификации интерфейса AEAD и определения реестра IANA для алгоритмов AEAD. В этот реестр включаются алгоритмы AEAD на основе AES[1] в режиме GCM[2] [GCM] с ключами размером 128 и 256 битов и в режиме CCM[3] [CCM] с ключами размером 128 и 256 битов.

Ниже определён интерфейс AEAD (раздел 2) и приведены рекомендации по использованию алгоритмов AEAD (раздел 3), а также очерчены требования, которым должен соответствовать каждый алгоритм AEAD (раздел 4). Далее определены некоторые алгоритмы AEAD (раздел 5) и описан реестр IANA для алгоритмов AEAD (раздел 6). В заключение рассмотрены некоторые дополнительные соображения (раздел 7).

Спецификация интерфейса AEAD не решает вопросов защиты протокола от повторного использования пакетов (anti-replay) или управления доступом на основе аутентифицированных данных. Более того, цель этой спецификации состоит в абстрагировании от этих вопросов. Интерфейс и рекомендации по его использованию согласуются с рекомендациями [EEM04].

1.3. Преимущества

Модель AEAD позволяет приложениям, которым нужны криптографические услуги защиты, более просто реализовать такие службы. Разработчики приложений в результате такого преимущества могут сосредоточиться на таких важных вопросах, как защитные службы, канонизация и сортировка данных, освобождаясь от необходимости разработки криптографических механизмов, соответствующих задачам защиты. Важно отметить, что безопасность алгоритма AEAD можно проанализировать независимо от его использования в конкретном приложении. Это освобождает пользователей AEAD от необходимости анализа аспектов защиты, связанных с относительным порядком применения аутентификации и шифрования, безопасности конкретной комбинации шифра и MAC (например, утраты конфиденциальности через MAC). Разработчикам приложений, использующих интерфейс AEAD, не требуется на этапе разработки выбирать конкретный алгоритм AEAD. Кроме того, интерфейс AEAD сравнительно прост, поскольку ему нужен лишь ключ в качестве входных данных и один идентификатор, указывающий алгоритм, применяемый в конкретном случае.

Модель AEAD обеспечивает преимущества при реализации криптоалгоритмов, предоставляя недоступную в других случаях оптимизацию для снижения объёма вычислений, стоимости реализации и/или требуемых ресурсов. Простой интерфейс упрощает тестирование, это весьма важно для реализации криптоалгоритмов. За счёт предоставления унифицированного интерфейса для доступа к криптографическим службам модель AEAD позволяет множеству приложений использовать одну реализацию криптосистемы. Например, аппаратный модуль с интерфейсом AEAD может обеспечивать ускорение криптографических операций любому приложению, которое использует этот интерфейс, даже если такое приложение было разработано до появления этого модуля.

1.4. Используемые соглашения

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не нужно** (SHALL NOT), **следует** (SHOULD), **не следует** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе должны интерпретироваться в соответствии с [RFC2119].

2. Интерфейс AEAD

Алгоритм AEAD включает две операции - аутентифицированное шифрование и аутентифицированную расшифровку. Входные и выходные данные этих алгоритмов определены ниже в терминах строк октетов.

Реализации **могут** принимать на входе дополнительные данные. Например, на входе может использоваться информация, позволяющая пользователю выбирать между разными стратегиями реализации. Однако **недопустимо** влияние таких расширений на взаимодействие с другими реализациями.

2.1. Аутентифицированное шифрование

Операция аутентифицированного шифрования принимает на входе 4 элемента в форме строк октетов.

Секретный ключ K, который **должен** генерироваться так, чтобы быть однородно случайным или псевдослучайным.

Одноразовое значение (nonce) N. Каждое значение nonce при разных вызовах операции Authenticated Encryption **должно** быть другим (новым) для любого конкретного значения ключа, если не используются значения nonce нулевого размера. Приложениям, которые могут генерировать разные значения nonce, **следует** использовать метод формирования nonce, описанный в параграфе 3.2, и **можно** применять любой другой метод, соответствующий требованиям уникальности значений. Остальным приложениям **следует** использовать значения nonce размером 0.

Открытый текст P, представляющий собой данные, которые будут шифроваться и аутентифицироваться.

Связанные данные A, которые будут аутентифицироваться без шифрования.

На выходе операции будет единственное значение.

[1]Advanced Encryption Standard - расширенный стандарт шифрования.
[2]Galois/Counter Mode.
[3]Counter and CBC MAC Mode.

Шифротекст C, размер которого не меньше размера открытых данных, или индикация невозможности выполнить запрошенную операцию шифрования.

Все входные и выходные данные представляют собой строки октетов переменного размера с приведёнными ниже ограничениями.

Размер строки октетов ключа K может составлять от 1 до 255. Для каждого алгоритма AEAD размер K **должен** быть фиксированным.

Для любого конкретного ключа должно выполняться одно из условий: 1) все значения nonce при разных вызовах операции Authenticated Encryption **должны** быть разными или 2) все значения nonce **должны** иметь размер 0. При использовании с конкретным ключом значений nonce нулевого размера, все значения для этого ключа **должны** иметь размер 0. В остальных случаях размер nonce **следует** делать равным 12. Для конкретного ключа **можно** использовать значения nonce разных размеров. Некоторые алгоритмы не могут работать с nonce нулевого размера (см. раздел 4). Приложениям, соблюдающим рекомендации по размеру nonce, не потребуется создавать значения nonce разного размера в зависимости от используемого алгоритма. Это позволяет вынести связанную с алгоритмом логику за пределы приложения.

Размер открытых данных P **может** быть нулевым.

Размер связанных данных A **может** быть нулевым.

Размер зашифрованных данных C **может** быть нулевым.

Эта спецификация не задаёт максимальный размер nonce, открытых и шифрованных данных, а также дополнительных данных аутентификации. Однако конкретный алгоритм AEAD **может** ограничивать размеры этих входных и выходных значений. Такие ограничения **может** вносить и конкретная реализация AEAD. Если реализации алгоритма AEAD предлагается обрабатывать входные данные, размер которых выходит за допустимый предел, она **должна** возвращать соответствующий код ошибки, выдавать какую-либо информацию на выходе алгоритма в этом случае **недопустимо**. В частности, **недопустимо** возвращать частично зашифрованные или частично расшифрованные данные.

Защита конфиденциальности и аутентификация сообщения обеспечиваются для открытых данных P. Если размер P равен 0, алгоритм AEAD действует, как код аутентификации сообщения (MAC) на входе A.

Связанные данные A являются данными, которым требуется аутентификация, но не нужна защита конфиденциальности. При использовании AEAD для защиты, например, сетевого протокола эти входные данные могут включать адреса, номера портов, порядковые номера, версии протокола и другие поля, указывающие как шифрованные или открытые данные следует обрабатывать, пересылать и обслуживать. Во многих случаях желательно проверять подлинность этих полей, но при этом они должны сохраняться в открытом виде для обеспечения корректной работы сети или системы. Когда эти данные включаются во вход A, их аутентификация выполняется без копирования в открытые данные (plaintext).

Секретный ключ K **недопустимо** включать в любой из других входов (N, P, A) (это не означает, что значения этих входов должны проверяться на предмет наличия в них субстрок, совпадающих с ключом, просто ключ недопустимо явно копировать в эти входы).

Для значения nonce алгоритмом выполняется внутренняя аутентификация и его не требуется включать во вход AD. Значение nonce **можно** включить в P или A, если это удобно для приложения.

Значение nonce **может** сохраняться или транспортироваться вместе с шифрованными данными, а также **может** восстанавливаться непосредственно перед операцией аутентифицированной расшифровки. Достаточно предоставить модулю расшифровки информацию, позволяющую воссоздать значение nonce (например, система может использовать значение nonce, содержащее порядковый номер в определённом формате и в этом случая значение может быть восстановлено из порядка следования шифрованных блоков). Поскольку процесс аутентифицированной расшифровки обнаруживает некорректные значения nonce, при некорректном воссоздании nonce и передаче значения процессу аутентифицированной расшифровки отказов защиты не возникает. Любому методу воссоздания значений nonce требуется принимать во внимание возможность потери или изменения порядка доставки шифрованных блоков между шифратором и дешифратором.

Приложениям **недопустимо** предполагать наличие любой конкретной структуры или формата в шифрованных данных.

2.2. Аутентифицированная расшифровка

Операция аутентифицированной расшифровки принимает на входе 4 значения K, N, A и C, определённые выше. На выходе операции имеется единственное значение, которым могут быть открытые (расшифрованные) данные P или специальный символ FAIL, показывающий, что входные данные недействительны. Зашифрованные данные C, nonce N и связанные данные A будут действительны для ключа K, когда значение C было создано операцией шифрования с входными данными K, N, P и A при тех же значениях N, P и A. Операция аутентифицированной расшифровки с высокой вероятностью будет возвращать результат FAIL, если входные данные N, C[1] и A были обработаны соблюдающим nonce нарушителем, которому не известен секретный ключ (предполагается, что алгоритм AEAD безопасен).

2.3. Форматирование данных

Этот документ не задаёт какого-либо конкретного представления входных или выходных данных AEAD, поскольку это представление не оказывает влияния на услуги защиты, обеспечиваемые алгоритмом AEAD.

При выборе приложением формата данных ему **следует** разместить зашифрованную информацию C так, чтобы она появлялась после всех прочих данных, которые нужны для создания других входных параметров алгоритма аутентифицированной расшифровки. Например, если в пакете присутствуют шифрованные данные и nonce, сначала должно размещаться значение nonce, а за ним - шифротекст. Это правило повышает эффективность работы AEAD и упрощает аппаратную реализацию.

[1] В оригинале ошибочно сказано P. См. https://www.rfc-editor.org/errata_search.php?eid=4008. *Прим. перев.*

3. Руководство по использованию алгоритмов AEAD

В этом разделе приведены рекомендации, следование которым поможет в безопасно применять алгоритмы AEAD.

Если приложение не может обеспечить генерацию уникальных одноразовых значений nonce, он **должно** использовать nonce нулевого размера. Для работы с такими приложениями подходят описанные ниже алгоритмы с учётом состояния или внесением случайности. В остальных случаях приложениям **следует** использовать значения nonce размером 12 октетов. Поскольку в алгоритмах поощряется использование такого размера, приложениям следует выбирать его для надёжного взаимодействия с другими приложениями.

3.1. Требования к генерации одноразовых значений

Для безопасности важно создавать одноразовые значения nonce так, чтобы выполнялось требование применять новое значение nonce при каждом вызове операции аутентифицированного шифрования для любого фиксированного ключа. В этом параграфе рассматриваются некоторые следствия этого требования для разных ситуаций.

При использовании одного ключа шифрования множеством устройств эти устройства должны координировать процессы обеспечения уникальности значений nonce. Простым способом реализации этого является использование формата nonce, включающего поле, которое различается для каждого из устройств, как описано в параграфе 3.2. Отметим отсутствие необходимости координировать детали формата nonce между шифрующей и дешифрующей стороной, поскольку значение nonce передаётся и сохраняется вместе с зашифрованными данными и, благодаря этому, доступно дешифратору. Если дешифратору недоступно полное значение nonce, он должен знать его структуру для того, чтобы восстановить значение. Приложениям **следует** давать модулю шифрования относительную свободу выбора значений nonce. Например, nonce может включать значение счётчика и поле, устанавливаемое шифратором, но не обрабатываемое на приёмной стороне. Такая свобода позволяет множеству шифраторов более легко координировать обеспечение уникальности их значений nonce.

Если секретный ключ будет использоваться долго (например, в течение периода, когда возможна перезагрузка системы), значение nonce потребуется сохранять в энергонезависимой памяти. В таких случаях важно использовать для nonce контрольные точки, т. е., следует сохранять текущее значение nonce для восстановления информации о состоянии, которая потребуется для возобновления шифрования в случае неожиданного отказа. Одним из простых способов обеспечить высокую вероятность того, что значение nonce не будет использовано повторно, является ожидание процессом шифрования сигнала от процесса хранения о том, что следующее значение nonce уже было сохранено. Поскольку такой метод может вызывать существенные задержки, желательно сохранять значения nonce с некоторым упреждением. Например, можно сохранить значение nonce 100 и после этого применять значения от 1 до 99 при шифровании. Значение nonce 200 может быть сохранено в то время, когда значения от 1 до 99 будут использоваться и т. д.

Многих проблем, связанных с повторным использованием nonce, можно избежать, меняя ключ в ситуации, когда координация значений nonce затруднена.

Для каждого алгоритма AEAD **следует** описывать влияние случайного повторного использования nonce на снижение уровня защиты.

3.2. Рекомендации по формированию Nonce

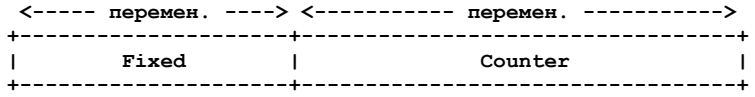

Рисунок 1. Рекомендуемый формат Nonce.

Рекомендуется применять описанный ниже метод создания значений nonce. Формат nonce показан на рисунке 1. Начальные октеты содержат фиксированное поле (Fixed), а в заключительных размещается значение счётчика (Counter). Для каждого фиксированного ключа размер каждого из этих полей и общий размер nonce фиксированы. Реализациям **следует** поддерживать 12-октетные значения nonce с полем Counter размером 4 октета.

Поля Counter последовательных nonce образуют монотонно возрастающую последовательность, когда значения этих полей рассматриваются как целые числа без знака с сетевым порядком байтов. Размер поля Counter **должен** сохраняться во всех значений nonce, генерируемых для данного шифровального устройства. Значение Counter **следует** делать нулевым для первого nonce и увеличивать на 1 для каждого последующего генерируемого nonce. Однако конкретное значение Counter **может** быть пропущено и отсутствовать в последовательности использованных значений, если это удобно. Например, приложение может пропустить начальное значение Counter=0 и начать нумерацию nonce с 1. Таким образом может быть создано до 2^{8*C} значений nonce при использовании поля Counter размером C октетов.

Поле Fixed **должно** оставаться постоянным во всех nonce, генерируемых для данного устройства шифрования. Если разные устройства выполняют шифрование с использованием одного ключа, каждое из таких устройств **должно** использовать своё значение поля Fixed для обеспечения уникальности значений nonce. Таким образом, один ключ может использовать до 2^{8*F}, если размер поля Fixed составляет F октетов.

3.2.1. Частично неявные значения Nonce

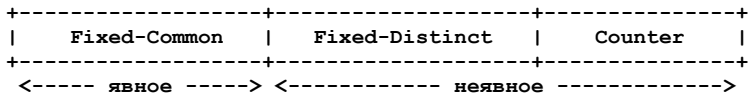

Рисунок 2. Формат Nonce с частичной неявностью.

В некоторых случаях желательно не передавать и не сохранять значения nonce целиком, а вместо этого восстанавливать значения из контекста непосредственно перед расшифровкой. Например, шифротекст может последовательно записываться на диск, а значение nonce для конкретного шифротекста определяется местоположением этого шифротекста и некими правилами создания nonce, известными на стороне расшифровки. Будем называть ту часть nonce, которая сохраняется или передаётся, явной, а восстанавливаемую часть nonce -

неявной. Когда часть nonce не задана явно, **рекомендуется** показанная ниже специализация описанного формата. Поле Fixed делится на две части - Fixed-Common и Fixed-Distinct, формат которых показан на рисунке 2. Если разные устройства выполняют шифрование с одним ключом, каждое из таких устройств **должно** использовать своё значение поля Fixed-Distinct, а поле Fixed-Common является общим для всех устройств. Поля Fixed-Distinct и Counter **должны** присутствовать в явной части nonce. Поле Fixed-Common **может** быть неявным. Эти соглашения обеспечивают простое восстановление nonce на основе явных данных.

Основания для использования частично неявных значений nonce приведены ниже. Этот метод создания nonce базируется на проверенном опыте - он применяется в GCM ESP[1] [RFC4106] и CCM ESP [RFC4309], где поле Fixed содержит значение «затравки» (Salt), а младшие восемь октетов nonce явно передаются в пакете ESP. В GCM ESP поле Fixed должно иметь размер не менее 4 октетов, чтобы в него можно было включить значение Salt. В CCM ESP поле Fixed должно быть размером не менее 3 октетов по той же причине. Этот метод создания nonce применяется также в некоторых вариантах шифрования в режиме counter, включая CTR ESP.

3.3. Подготовка входных данных AEAD

Если вход AD собран из множества элементов, важна возможность однозначного разбора на элементы без использования в процессе разбора неаутентифицированных данных (математически вход AD должен быть инъективной функцией элементов данных). Если приложение создаёт входные данные AD таким способом, что существуют два разных набора элементов, дающих одинаковое значение AD, атакующий может вынудить получателя к восприятию ложного набора, подменяя один набор другим. Требование однозначного разбора делает такие атаки невозможными. Это требование тривиально выполняется, если AD собирать из элементов фиксированного размера. Если же AD содержит строки переменного размера, это требование можно выполнить включением размера строк в AD.

Аналогично, если открытый текст собирается из множества элементов, важна возможность его однозначного разбора на элементы без привлечения неаутентифицированных данных. Отметим, что данные, включённые в AD, могут применяться при разборе открытого текста, хотя в результате отсутствия шифрования AD становится возможной утрата конфиденциальности при включении информации об открытом тексте в AD.

3.4. Пример использования

Для применения алгоритма AEAD приложение должно указать, как будет определяться ввод алгоритма шифрования в терминах данных приложений, а также способ передачи шифротекста и nonce. Простейший способ заключается в выражении каждого ввода в терминах формирующих его данных, а затем выражение данных приложения в терминах вывода операции шифрования AEAD.

Например, AES-GCM ESP [RFC4106] можно выразить следующим способом. Входом AEAD будет

```
P = RestOfPayloadData || TFCpadding || Padding || PadLength || NextHeader
N = Salt || IV
A = SPI || SequenceNumber
```

где символ || обозначает конкатенацию, поля RestOfPayloadData, TFCpadding, Padding, PadLength, NextHeader, SPI и SequenceNumber определены в [RFC4303], а Salt и IV - в [RFC4106]. Поле RestOfPayloadData содержит открытые данные, которые описаны полем NextHeader, и не включает других данных (напомним, что поле PayloadData содержит IV и RestOfPayloadData, см. рисунок 2 в [RFC4303]).

Формат пакета ESP можно представить в форме

```
ESP = SPI || SequenceNumber || IV || C
```

где C - шифротекст AEAD (который в данном случае включает тег аутентификации). Следует отметить, что здесь не описывается использование расширенных порядковых номеров ESP.

4. Требования к спецификации алгоритмов AEAD

Каждый алгоритм AEAD **должен** воспринимать лишь ключи фиксированного размера K_LEN и **недопустимо** требовать какого-либо конкретного формата данных для предоставляемых на входе ключей. Алгоритмы, которым требуются такие структуры (например, субключ с определенным форматом контроля чётности), должны обеспечивать их внутренними средствами.

Каждый алгоритм AEAD **должен** воспринимать любые открытые данные с размером от 0 до P_MAX октетов, включительно (значение P_MAX определяется алгоритмом). Значение P_MAX **должно** быть дольше 0 и **следует** поддерживать размер не менее 65536 (2^{16}) октетов. Этот размер обычно является предельным размером для сетевых пакетов данных. Другие приложения могут использовать большие значения P_MAX, поэтому для алгоритмов общего назначения важна поддержка больших значений.

Каждый алгоритм AEAD **должен** воспринимать любые связанные данные размером от 0 до A_MAX октетов, включительно (значение A_MAX определяется алгоритмом). Значение A_MAX **должно** быть больше 0 и **следует** поддерживать размер не менее 65536 (2^{16}) октетов. Другие приложения могут использовать большие значения A_MAX, поэтому для алгоритмов общего назначения важна поддержка больших значений.

Каждый алгоритм AEAD **должен** воспринимать любые значения nonce размером от N_MIN до N_MAX октетов, включительно (значения N_MIN и N_MAX определяются алгоритмом). Значения N_MAX и N_MIN **могут** совпадать. Каждому алгоритму **следует** воспринимать nonce размером 12 октетов. Алгоритмы, применяющие случайность или состояние, описанные ниже, **могут** применять N_MAX = 0.

Алгоритм AEAD **может** произвольно структурировать выходной шифротекст, например, шифрованные данные могут включать тег аутентификации. Каждому алгоритму **следует** выбирать структуру, обеспечивающую эффективную обработку.

Алгоритм аутентифицированного шифрования **может** включать или использовать источник случайных значений, например, для генерации внутреннего вектора инициализации, который встраивается в шифрованный вывод. Алгоритмы AEAD этого типа называются «недетерминированными» (randomized), хотя следует отметить, что

[1]Encapuslating Security Payload - инкапсуляция данных защиты.

случайным является лишь шифрование, а расшифровка всегда детерминирована. Такие алгоритмы **могут** иметь N_MAX = 0.

Алгоритм аутентифицированного шифрования **может** включать информацию о внутреннем состоянии, которое поддерживается между вызовами операции шифрования, например, для того, чтобы поддерживать создание разных значений, которые могут применяться в качестве внутренних nonce. Алгоритмы AEAD этого типа называют алгоритмами с учётом состояния (stateful). Этот метод может применяться алгоритмом для обеспечения лучшей защиты даже в тех случаях, когда от приложения приходят nonce нулевого размера. Такие алгоритмы **могут** иметь N_MAX = 0.

Спецификация алгоритма AEAD **должна** включать значения K_LEN, P_MAX, A_MAX, N_MIN и N_MAX, определённые выше. Кроме того, она **должна** задавать максимальное число октетов шифротекста, которое здесь обозначено C_MAX.

Каждый алгоритм AEAD **должен** обеспечивать описание, связывающее размер открытых данных с размером шифротекста. Для этого отношения размеров **недопустима** зависимость от внешних факторов, таких как параметр строгости аутентификации (например, размер тега аутентификации). Такая зависимость усложнит использование алгоритмов по причине возникновения ситуаций, в которых информации из реестра AEAD будет недостаточно для обеспечения корректного взаимодействия.

Спецификации **каждого** алгоритма AEAD **следует** описывать снижение уровня защиты, которое может возникать в результате непреднамеренного повтора значений nonce.

Спецификации **каждого** алгоритма AEAD **следует** включать ссылку на подробный анализ защищенности. В этом документе не задана определённая модель безопасности, поскольку в литературе используется несколько разных моделей. В анализе защищенности **следует** определять или указывать модель безопасности.

Алгоритмы, использующие случайность или состояние, как указано выше, **следует** описывать с использованием этих терминов.

5. Алгоритмы AEAD

В этом разделе определены 4 алгоритма AEAD, два из которых основаны на AES GCM, два других - на AES CCM. Каждая пара включает алгоритм с размером ключа 128 и 256 битов.

5.1. AEAD_AES_128_GCM

Алгоритм аутентифицированного шифрования AEAD_AES_128_GCM, как указано в [GCM], использует блочный шифр AES-128, обеспечивая ключ, nonce, открытые данные и связанные данные для этого режима работы. Используется тег аутентификации размером 16 октетов (128 битов). Шифротекст AEAD_AES_128_GCM формируется путём добавления тега аутентификации, обеспечиваемого на выходе операции шифрования GCM, в конец шифротекста, являющегося выводом этой операции. Тестовые примеры представлены в приложении к [GCM]. Размеры входных и выходных данных приведены ниже.

K_LEN - 16 октетов.

P_MAX - 2^{36} - 31 октетов.

A_MAX - 2^{61} - 1 октетов.

N_MIN и N_MAX - по 12 октетов каждое.

C_MAX - 2^{36} - 15 октетов.

Размер шифротекста AEAD_AES_128_GCM превышает размер исходных открытых данных в точности на 16 октетов.

Анализ защищенности алгоритма GCM представлен в [MV04].

5.1.1. Повторное использование Nonce

Непреднамеренное повторное использование значения nonce в двух вызовах операции шифрования GCM с одним ключом, но разными открытыми данными, подвергает риску конфиденциальность данных, защищённых при этих вызовах, и подрывает всю защиту целостности и аутентификацию, обеспечиваемые данным ключом. Поэтому GCM следует применять лишь в тех случаях, где гарантируется уникальность nonce. Особенность устройства GCM, используемая для минимизации задержки, создаёт уязвимость при последующем использовании ключа. Отметим, что допускается многократное применение одного значения nonce для операций расшифровки.

Последствия для безопасности могут быть достаточно серьёзными, если атакующий увидит два шифроблока, созданные с одним nonce и ключом, несмотря на различие открытых данных и значений AD при двух вызовах операции шифрования. Во-первых, теряется конфиденциальность, поскольку атакующий может восстановить значение XOR для двух блоков нешифрованных данных. Во-вторых, теряется защита целостности, поскольку атакующий может восстановить внутренних хэш-ключ, использованный для защиты целостности. Знание этого ключа позволяет ему легко подделать последующие данные.

5.2. AEAD_AES_256_GCM

Этот алгоритм идентичен AEAD_AES_128_GCM за исключением двух отличий:

K_LEN - 32 октета вместо 16;

AES-256 GCM вместо AES-128 GCM.

5.3. AEAD_AES_128_CCM

Алгоритм аутентифицированного шифрования AEAD_AES_128_CCM работает в соответствии с [CCM], используя блочный шифр AES-128 и предоставляя ключ, nonce, связанные данные и открытые данные для этого режима работы.

Функция форматирования и генерации счётчиков задана в приложении A к указанному документу, а значения параметров, указанные там же, приведены ниже.

Размер nonce - n = 12.

Размер тега - t = 16.

q = 3.

Используется тег аутентификации размером 16 октетов (128 битов). Шифротекст AEAD_AES_128_CCM формируется путём добавления тега аутентификации, обеспечиваемого на выходе операции шифрования CCM, в конец шифротекста, являющегося выводом этой операции. Тестовые примеры представлены в приложении к [CCM]. Размеры входных и выходных данных приведены ниже.

K_LEN - 16 октетов.

P_MAX - 2^{24} - 1 октетов.

A_MAX - 2^{64} - 1 октетов.

N_MIN и N_MAX - по 12 октетов каждое.

C_MAX - 2^{24} + 15 октетов.

Размер шифротекста AEAD_AES_128_CCM превышает размер исходных открытых данных в точности на 16 октетов.

Анализ защищенности алгоритма AES CCM представлен в [J02].

5.3.1. Повторное использование Nonce

Непреднамеренное повторное использование значения nonce в двух вызовах операции шифрования CCM с одним ключом подвергает риску защиту сообщений, обработанных в этих вызовах. Потеря конфиденциальности связана с тем, что атакующий может восстановить побитово значения XOR для двух блоков открытых данных.

5.4. AEAD_AES_256_CCM

Этот алгоритм идентичен AEAD_AES_128_CCM за исключением двух отличий:

K_LEN - 32 октета вместо 16;

AES-256 CCM вместо AES-128 CCM.

6. Взаимодействие с IANA

Агентство IANA создало реестр AEAD, описанный ниже. Разработчики **могут** регистрировать свои алгоритмы для облегчения их использования. Для добавления в реестр AEAD требуется документированная спецификация в виде RFC или иного документа с постоянным открытым доступом, достаточно подробно описывающая взаимодействие между независимыми реализациями алгоритма. Каждая запись реестра содержит перечисленные ниже элементы.

Краткое имя, вида AEAD_AES_128_GCM, которое начинается с префикса AEAD.

Целое положительное значение идентификатора.

Ссылку на спецификацию, которая полностью определяет алгоритм AEAD и содержит тестовые примеры, которые можно использовать для проверки корректности реализации.

Запрос на добавление записи в реестр **должен** включать имя и ссылку, а номер назначает IANA. При назначении номеров **следует** использовать наименьшее положительное число из доступных. Заявителю **следует** представить свой запрос на рецензию группе IRTF CFRG[1] по адресу cfrg@ietf.org. Заявителям, не знакомым с процедурами IANA, следует посетить сайт http://www.iana.org.

Номера от 32768 (двоичное 1000000000000000) до 65535 (1111111111111111), включительно, не будут назначаться IANA и резервируются для приватного использования. Не будет предприниматься никаких попыток предотвратить использование одного номера на множестве сайтов разными (и не совместимыми) способами [RFC2434].

Агентство IANA добавило перечисленные в таблице записи в реестр AEAD.

Имя	Описание	Идентификатор
AEAD_AES_128_GCM	Параграф 5.1	1
AEAD_AES_256_GCM	Параграф 5.2	2
AEAD_AES_128_CCM	Параграф 5.3	3
AEAD_AES_256_CCM	Параграф 5.4	4

Регистрация в IANA алгоритма AEAD не означает его одобрения или защищенности.

7. Другие вопросы

Непосредственное тестирование недетерминированного алгоритма AEAD с использованием тестовых примеров с фиксированными входными и выходными данными невозможно, поскольку процесс шифрования является недетерминированным. Однако можно проверить недетерминированный алгоритм AEAD с использованием описанного ниже метода. Алгоритм аутентифицированной расшифровки является детерминированным и может быть протестирован напрямую. Алгоритм аутентифицированного шифрования можно проверить путём шифрования открытых данных, расшифровки результата и сравнения исходных данных с расшифрованными. Комбинация этих тестов охватывает сразу алгоритмы шифрования и расшифровки.

Выбранные алгоритмы AEAD отражают их стандартизацию. Вопрос добавления других алгоритмов AEAD остаётся открытым. Возможно множество вариантов базовых алгоритмов, каждый из которых имеет свои преимущества.

[1]Crypto Forum Research Group - исследовательская группа криптофорума.

Несмотря на желание принять любые алгоритмы, которые могут быть полезны на практике, желательно ограничить общее число зарегистрированных алгоритмов. Текущая спецификация требует для регистрации алгоритма полную спецификацию и набор данных для проверки. Есть надежда, что это позволит задать верные критерии регистрации.

Может оказаться желательным определение алгоритма AEAD, который использует общую композицию с методом «шифрование, затем MAC»[BN00], объединяя базовый алгоритм шифрования (такой как CBC [MODES]) с базовым кодом аутентификации сообщений (таким как HMAC-SHA1 [RFC2104] или AES CMAC [CMAC]). Алгоритм AEAD этого типа будет отражать накопленный опыт и может легче поддерживаться криптомодулями, которые не поддерживают другие алгоритмы AEAD.

8. Вопросы безопасности

Этот документ описывает алгоритмы аутентифицированного шифрования и даёт рекомендации по их применению. Хотя эти алгоритмы в некотором смысле облегчают разработку криптографических приложений, следует помнить, что обеспечения надёжной криптографической защиты является сложной задачей. Алгоритмы AEAD достаточно полезны, но они не способствуют решению вопросов генерации ключей [RFC4086] и управления ими [RFC4107].

Алгоритмы AEAD, основанные на различных значениях nonce, могут не подходить для некоторых приложений или вариантов использования. Разработчикам приложений следует разобраться с требованиями, приведёнными в параграфе 3.1.

Программная реализация операции шифрования AEAD на виртуальной машине (VM[1]) может непреднамеренно повторно использовать nonce в результате «отката» VM к предыдущему состоянию [GR05]. Приложениям рекомендуется документировать возможные проблемы, чтобы помочь пользователям приложений и VM в предотвращении непреднамеренных ошибок такого рода. Существует возможность форсирования атакующим отката VM. Угрозы и их смягчение в таких ситуациях активно исследуются. С учётом перспективы следует отметить, что атакующий, способный инициировать такой откат, возможно уже преодолел защиту системы, например, вызвав ошибку учёта.

Регистрацию в IANA алгоритма AEAD **недопустимо** считать подтверждением защищенности алгоритма. Кроме того, уровень защищенности алгоритма может снижаться со временем в результате криптоанализа или по закону Мура (за счёт снижения стоимости вычислительных ресурсов с течением времени).

9. Благодарности

Многие люди представили существенные комментарии к предварительным версиям этого документа. Некоторые полезные обсуждения прошли в почтовой конференции Crypto Forum Research Group в 2006 году.

10. Литература

10.1. Нормативные документы

[CCM] Dworkin, M., "NIST Special Publication 800-38C: The CCM Mode for Authentication and Confidentiality", U.S. National Institute of Standards and Technology, <http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C.pdf>.

[GCM] Dworkin, M., "NIST Special Publication 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.", U.S. National Institute of Standards and Technology, November 2007, <http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Дополнительная литература

[BN00] Bellare, M. and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm", Proceedings of ASIACRYPT 2000, Springer-Verlag, LNCS 1976, pp. 531-545, 2002.

[BOYD] Boyd, C. and A. Mathuria, "Protocols for Authentication and Key Establishment", Springer 2003.

[CMAC] "NIST Special Publication 800-38B", <http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf>.

[EEM04] Bellare, M., Namprempre, C., and T. Kohno, "Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the Encode-then-Encrypt-and-MAC paradigm", ACM Transactions on Information and System Security, <http://www-cse.ucsd.edu/users/tkohno/papers/TISSEC04/>.

[GR05] Garfinkel, T. and M. Rosenblum, "When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments", Proceedings of the 10th Workshop on Hot Topics in Operating Systems, <http://www.stanford.edu/~talg/papers/HOTOS05/virtual-harder-hotos05.pdf>.

[J02] Jonsson, J., "On the Security of CTR + CBC-MAC", Proceedings of the 9th Annual Workshop on Selected Areas on Cryptography, 2002, <http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ccm/ccm-ad1.pdf>.

[MODES] Dworkin, M., "NIST Special Publication 800-38: Recommendation for Block Cipher Modes of Operation", U.S. National Institute of Standards and Technology, <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>.

[MV04] McGrew, D. and J. Viega, "The Security and Performance of the Galois/Counter Mode (GCM)", Proceedings of INDOCRYPT '04, December 2004, <http://eprint.iacr.org/2004/193>.

[1]Virtual Machine.

[R02] Rogaway, P., "Authenticated encryption with Associated-Data", ACM Conference on Computer and Communication Security (CCS'02), pp. 98-107, ACM Press, 2002, <http://www.cs.ucdavis.edu/~rogaway/papers/ad.html>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4106] Viega, J. and D. McGrew, "The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)", RFC 4106, June 2005.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, June 2005.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, December 2005.

[RFC4309] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)", RFC 4309, December 2005.

Адрес автора

David A. McGrew

Cisco Systems, Inc.

510 McCarthy Blvd.

Milpitas, CA 95035

US

Phone: (408) 525 8651

EMail: mcgrew@cisco.com

URI: http://www.mindspring.com/~dmcgrew/dam.htm

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

Полное заявление авторских прав

Интеллектуальная собственность