Анонимная сеть «Hidden Lake»

Коваленко Геннадий Александрович

Аннотация. Сеть Hidden Lake, являясь по природе своей QB-сетью, представляет собой также ряд новых архитектурных решений, ранее не применявшихся в строении анонимных систем. Базируясь на принципе микросервисной архитектуры, таковая сеть может не только добавлять, но также и удалять функции по мере своей необходимости, никак не изменяя при этом общий механизм работы. Базируясь на слепой маршрутизации и полном шифровании сообщений, таковая сеть связывает всех узлов в системе, не позволяя применять долговременное наблюдение за связями и фактом коммуникации. Понимание общих принципов работы сети на базе её математических моделей способно дать не только оценку корректности функционирования всей системы, но и также возможный вектор развития будущих анонимных коммуникаций.

Ключевые слова: скрытые системы; анонимные сети; децентрализованные сети; теоретически доказуемая анонимность; qb-задача; микросервисная архитектура; сеть hidden lake;

Содержание

1. Введение

Анонимная сеть Hidden Lake (HL) - это децентрализованная F2F (friend-to-friend) анонимная сеть с теоретической доказуемостью [1]. В отличие от известных анонимных сетей, подобия Tor, I2P, Mixminion, Crowds и т.п., сеть HL способна противостоять атакам глобального наблюдателя. Сети Hidden Lake для анонимизации своего трафика не важны такие критерии как: 1) уровень сетевой централизации, 2) количество узлов, 3) расположение узлов и 4) связь между узлами в сети, что делает таковую систему абстрактной [2].

2. QB-задача

Задача на базе очередей [2] (Queue Based) представляет собой ядро анонимной сети Hidden Lake за счёт которого формируется теоретически доказуемая анонимность. QB-сети представляют собой одну из наиболее простых задач анонимизации в плане программной реализации, в сравнении с другими коллегами по теоретической доказуемости в лице DC (Dining Cryptographers) и EI (Entropy Increase) -сетей. Формально QB-сеть можно описать системой следующего вида:

$$QB\text{-}net = \Sigma^{n}_{i=1}(T = \{t_i\}, K = \{k_i\}, C = \{(c \in \{E_{kj}(m), E_r(v)\}) \leftarrow^{ti} Qi\})$$

где n - количество узлов в системе, *K* - множество ключей шифрования, *T* - множество периодов генерации, *C* - множество зашифрованных сообщений, *Q* - очередь шифрованных сообщений, *i,j* - идентификаторы отдельных узлов, *E* - функция шифрования, *m* - открытое сообщение, *v* - ложное сообщение, *r* - ключ шифрования не находящийся во множестве *K*.

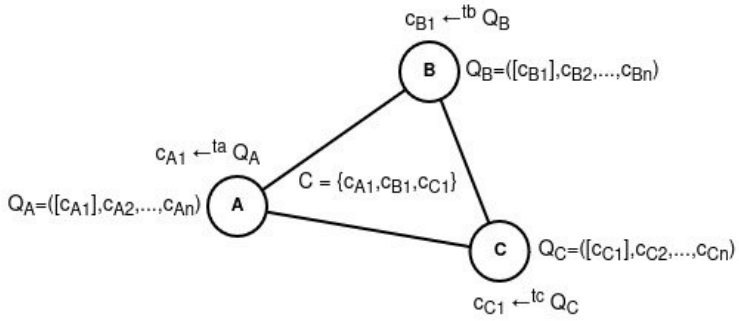

Рисунок 1. QB-сеть с тремя участниками A, B, C

Суть вышеописанной системы может быть легко представлена в виде следующего алгоритма:

1. $Q \leftarrow (c = E_{ki}(m))$, *где* $k_i \in K$, $c \in C$. Открытое сообщение *m* шифруется ключом получателя $k_i \in K$. Результат шифрования $c = E_{ki}(m)$ помещается в очередь *Q*,

2. $(c = E_{ki}(m)) \leftarrow^{t} Q$, *если* $Q \neq \emptyset$, *где* $t \in T$, $k_i \in K$, $c \in C$. В каждый период времени *t* из очереди *Q* берётся шифрованное сообщение *c* и отправляется всем участникам сети,

3. $(c = E_r(v)) \leftarrow^{t} Q$, *если* $Q = \emptyset$, *где* $t \in T$, $r \notin K$, $c \in C$. Если на период времени *t* очередь *Q* остаётся пустой, то создаётся ложное сообщение *v*, которое далее шифруется ключом без получателя $r \notin K$. Результат шифрования $c = E_r(v)$ отправляется всем участникам сети,

4. $m' = D_k^{-1}(c)$, *где* $c \in C$. Каждый участник пытается расшифровать полученное шифрованное сообщение *c* из сети своим ключом k^{-1}. Если сообщение не поддаётся расшифрованию $m' \neq m$, то это значит, что получателем является либо кто-то другой, либо никто.

Анонимность QB-сетей определяется не только разрывом связи между отправителем и получателем для глобального наблюдателя, но также и полным отсутствием связи в самом

факте отправления и получения информации. Иными словами, для пассивных наблюдателей, включающих в себя и глобального наблюдателя, ставится непосильной задача определения состояния субъекта, а именно:

1. Отправляет ли участник i в период равный t_i истинное сообщение $E_{kj}(m)$?
2. Получает ли участник i в периоды равные $T\backslash\{t_i\}$ какое-либо сообщение $D_{ki}^{-1}(c)$?
3. Бездействует ли участник $i \rightarrow E_r(v)$ в анализируемом периоде t_i?

При всех таких сценариях, не будучи одним из узлов участвующих непосредственно в коммуникации и не проявляющим какое-либо влияние на очередь Q_i анализируемого участника i, т.е. не будучи узлом проявляющим активное наблюдение, задача считается невыполнимой, если алгоритм шифрования E и ключи k,r являются надёжными.

В QB-сетях есть также ряд интересных и не совсем однозначных моментов. Так например, период t_i каждого отдельного участника i не обязательно должен иметь константное значение. Период может изменяться по времени или вовсе иметь случайное значение. Такое поведение никак не отразится на анонимности узлов до тех пор, пока будет существовать сам факт отложенности сообщений c в лице очереди сообщений Q. Если сообщение можно будет отправлять в обход очерёдности, тогда анонимность будет постепенно ухудшаться в зависимости от количества отправляемых таким образом сообщений.

Таким образом, в отличие от DC-сетей, где период T представлен только одним общим значением $T\backslash\{t\} = \varnothing$, QB-сети делают период не только субъективно (индивидуально) настраиваемым $T = \{t_1, t_2, ..., t_n\}$, но также и не обязательно статичным для каждого генерируемого сообщения $t \in [l;k]$, *где* $l \leq k$**.** Такое свойство позволяет QB-сетям не кооперировать с отдельными узлами за период, а также более качественно скрывать закономерность принадлежности пользователя к анонимизирующему трафику.

Далее, в QB-сетях предполагается использование асимметричной криптографии по умолчанию, и как следствие ключи $k_i \neq k_i^{-1}$ не связаны между собой напрямую. Тем не менее QB-сети вполне способны руководствоваться исключительно симметричной криптографией при которой $k_i = k_i^{-1}$. В таком случае ключи будут представлять не каждого отдельного участника системы из n возможных, а непосредственно связь между её участниками из $[n(n-1)]/2$ возможных рёбер графа (системы), что приводит также к появлению общих ключей вида $k_i = k_j$ для некоторых i, j связей. Если в системе заложен механизм маршрутизации, то для расшифрования получатель должен будет использовать уже не один конкретный ключ со стороны отправителя, а все ему известные ключи посредством метода их перебора.

Хоть использование симметричной криптографии и становится возможным в QB-сетях, всё же следует отдать предпочтение асимметричным алгоритмам, потому как:

1. В таком случае упростится общая система количественного хранения ключей с $[n(n-1)]/2$ до $2n$,

2. Расшифрование информации станет более доступным в маршрутизирующей системе, исключив тем самым перебор известных ключей k_i^{-1} до использования одного ключа k^{-1},

3. Аутентификация пользователей станет более качественной, за счёт частичного решения проблемы отказа от авторства при помощи цифровых подписей.

2.1. Недостатки QB-сетей

К сожалению QB-сети неидеальны и также обладают, свойственными своему классу, проблемами и недостатками, ряд из которых приводит к ограничению прикладного использования, другой ряд приводит к проблемам доступности сети:

1. Линейная нагрузка на сеть. В QB-сетях каждый отправляет сообщение всем с той лишь целью, чтобы невозможно было сузить область реальной коммуникации участников системы. Алгоритмом маршрутизации становится слепая (заливочная) маршрутизация, вследствие чего увеличение количества узлов сказывается линейно $O(n)$ на увеличение нагрузки всей системы,

2. Привязанность к очерёдности. Каждый узел в QB-сети так или иначе завязан на собственной очерёдности сообщений Q, где каждый период времени равный t шифрованное сообщение отправляется в сеть. Это значит, что повысить пропускную способность узла возможно лишь в трёх сценариях, каждый из которых будет приводить к увеличению нагрузки на всю сеть:

    1. Повысить размер передаваемых сообщений $m_1, m_2, ..., m_n$,
    2. Повысить количество отправляемых сообщений за раз $c_1, c_2, ..., c_n \leftarrow^t Q$,
    3. Понизить период генерации сообщений t,

3. Связность абонентов коммуникации. QB-сети не предполагают анонимности между узлами непосредственно участвующих в общении. Связано это в первую очередь с тем, что в QB-сетях отсутствует такое понятие как полиморфизм информации, то есть состояние информации в системе при котором её внешний вид постоянно меняется от узла к узлу. Такое свойство позволяет разрывать связь между отправителем и получателем посредством передаваемого объекта, т.е. самой информации. Полиморфизм в анонимных сетях чаще всего достигается множественным шифрованием, где при передаче от одного узла к другому постепенно снимаются наложенные слои шифрования, как например в Tor и I2P.

$$E_{k3}(E_{k2}(E_{k1}(m))) \rightarrow E_{k2}(E_{k1}(m)) \rightarrow E_{k1}(m) \rightarrow m.$$

Так например, если в QB-сети будет существовать сущность в виде ретрансляторов, скрывающая сетевые адреса абонентов (IP-адреса) друг от друга посредством перенаправления трафика, и будет при этом существовать кооперация одного из абонентов с глобальным наблюдателем, то задача связывания $IP \leftrightarrow k_i$ будет тривиальной, т.к. абоненту достаточно будет получить одно истинное сообщение $m = D_k^{-1}(c)$ от другого абонента, а далее по полученному шифрованному тексту $c = E_k(m)$ глобальному наблюдателю остаётся определить первое его появление.

Ситуацию можно усложнить для наблюдателей при помощи добавления канального шифрования, как например в Crowds. В таком случае глобальный наблюдатель не сможет явно связать отправленное и полученное сообщение, потому как оно будет постоянно менять свой вид при передаче от одного узла к другому.

$E_{k3}(\mathrm{m}) \rightarrow E_{k2}(\mathrm{m}) \rightarrow E_{k1}(\mathrm{m}) \rightarrow m$

Тем не менее это не является полиморфизмом информации в чистом виде, т.к. не выполняет функцию разграничения узлов между собой к маршрутизирующей информации *m*. Вследствие этого, задачей глобального наблюдателя станет вживление своих подчинённых узлов в систему рядом с каждым другим узлом. При таком сценарии он также легко сможет связать $IP \leftrightarrow k_i$.

Чисто технически в QB-сеть можно внедрить и множественное шифрование, чтобы разграничивать абонентов друг от друга, но в таком случае:

1. Уменьшится скорость передачи информации, т.к. каждый следующий узел в маршрутизирующей цепочке должен будет сохранять полученное ранее сообщение в свою очередь для последующей ретрансляции,

2. Усложнится система анонимизации в целом, т.к. вместо одной задачи анонимизации = *QB* будет использоваться уже композиция задач = *QB* + *OnionRouting*,

3. Композиция *QB* + *OnionRouting* обладает рядом тонкостей [2] с более сложными активными атаками, но всё также подрывающими анонимность связи между отправителем и получателем.

Вследствие всех вышеприведённых недостатков область применения QB-сетей становится более ограниченной:

1. Из-за линейной нагрузки на сеть и привязанности к очерёдности QB-сети плохо масштабируются и могут работать лишь в малых группах до *N* участников. Предел количества участников ограничен пропускной способностью самой сети, а также мощностью узлов постоянно шифрующих и расшифровывающих исходящий / входящий трафик. Следствием этого недостатка также является отсутствие существования поточной связи в лице стриминг-сервисов и видео/аудио-звонков,

2. Из-за связности абонентов коммуникации ограничивается ряд прикладных решений в которых важна анонимность узлов друг к другу. Вследствие этого, появляется наиболее релевантная композиция QB-сетей с F2F-сетями (friend-to-friend), где установление коммуниципирующей связи происходит двумя абонентами системы, а не одним из. Это не решает проблему отсутствия анонимности между связываемыми узлами, но даёт дополнительную защиту от несогласованного автоматического связывания и более явную связь доверительных коммуникаций, предполагающую что ни один из абонентов не будет пытаться деанонимизировать другого.

Теперь, если предположить наихудший сценарий атаки на QB-сеть при котором в кругу друзей $F = \{f_1, f_2, ..., f_n\}$ участника *i* будет существовать злоумышленник f_j в роли активного наблюдателя способного отправлять запросы и получать ответы от *i*, тогда модель атаки будет сводиться к анализу состояния очереди Q_i. Предположим, что участник *i* выставил статичный период генерации сообщений равный t_i. В таком случае f_j сможет в определённые интервалы времени $\{t'_i, 2t'_i, ..., nt'_i\}$, зависимые от периода времени $t_i \Rightarrow (kt'_i = kt_i + x)$, *где x*

$\in [0;t_i]$, отправлять запрос $R_{kt'i}$ участнику i с целью анализа времени ответа. Если ответ, полученный после запроса $R_{kt'i}$, будет генерироваться в диапазоне dt_i, *где* $d > 1$, то это будет означать факт реальной коммуникации участника i с кем либо в сети в периоды $\{(1+k)t_i, (2+k)t_i, ..., (d-1+k)t_i\}$, т.к. для ответа потребовался более чем один период. Если же $d = 1$, тогда участник i ни с кем не кооперировал в период $(1+k)t_i$.

Таким образом, вышеописанная атака снижает качество анонимности QB-сетей с сокрытия факта активности до сокрытия коммуникационной связи между абонентами. Иными словами, при таком активном наблюдении теперь становится возможным определение состояния субъекта в лице отправления или получения истинных сообщений, но до сих пор остаются под вопросов следующие моменты:

1. Кто является инициатором действий (запросов): сам прослушиваемый участник i или какой-то другой участник j из множества N?

2. С кем из множества N общался прослушиваемый участник i в периоды $\{(1+k)t_i, (2+k)t_i, ..., (d-1+k)t_i\}$?

2.2. Сравнение с другими задачами

	QB	EI	DC	Onion	Proxy
Теоретическая доказуемость	+	+	+	-	-
Накопительный эффект анонимности	-	+	-	-	-
Полиморфизм информации	-	+	+	+	-
Вероятностная маршрутизация	-	+	-	+/-	+/-
Периодичность генерации сообщений	+/-	-	+	-	-
Простота масштабирования	-	-	-	+	+
Простота программной реализации	+	-	-	+	+
Стадия анонимности	5^	6	1^	4 или 6	3
Сеть-представитель	Hidden Lake	-	Herbivore	Tor	Crowds

Таблица 1. Сравнение задач анонимности

3. Функция шифрования

Как было ранее показано, QB-сети зависимы от качества функции E и ключей k,r шифрования. Качество ключей шифрования определяется в первую очередь качеством ГСЧ (генератором случайных чисел) и/или КСГПСЧ (криптографически стойким генератором псевдослучайных чисел). Анализ таковых генераторов сложен по причине разных сред, в которых они исполняются, и средств, которые они задействуют в ходе своего выполнения. Поэтому исходя из логики абстрагирования мы будем далее предполагать, что ключи

генерируются качественным и безопасным образом, фокусируясь тем самым исключительно на логике исполнения функции шифрования.

$$E_{(k,privA,pubB)}(m) = E''_k(E'_{(privA,pubB)}(m))$$

Функция шифрования в сети Hidden Lake состоит из двух этапов, каждый из которых выполняет свою точно заданную роль. Первый этап $E'_{(privA,pubB)}$ сводится к непосредственному и первичному шифрованию данных с целью их сокрытия от посторонних лиц, используя для этого гибридную схему шифрования [3] (асимметричная + симметричная криптография). Второй этап E''_k сводится к разделению нескольких сетей посредством применения разных ключей шифрования (сетевых ключей).

3.1. Первый этап шифрования

$$E'_{(privA,pubB)}(m) = (E_{pubB}(k') \,||\, E_{k'}(pubA \,||\, s \,||\, h \,||\, S_{privA}(h) \,||\, f(m)))$$

$$k' = [RNG],\ s = [RNG],\ h = H_{mac(s)}(pubA \,||\, pubB \,||\, f(m)))$$

где k'- сеансовый ключ шифрования рассчитанный на одно сообщение, s - криптографическая соль рассчитанная на одно сообщение, m - открытое сообщение, $pubA$, $pubB$ - публичные ключи участников A, B соответственно, $privA$ - приватный ключ участника A, h - результат хеширования, S - функция подписания, f - функция дополнения сообщения до константной величины. В этой схеме предполагается, что A - есть отправитель информации m, B - есть получателя данной информации.

Безопасность данной функции зависит непосредственно от публичного ключа шифрования $pubB$, которым шифруется последующий сеансовый ключ k', от качества ГСЧ / КСГПСЧ которым был сгенерирован k', и также от безопасности самих функций шифрования E_{pubB}, $E_{k'}$.

Данная схема интересна тем, что она скрывает всю информацию в шифрованной оболочке, непозволяющей осуществлять атаки на идентификацию отправителя или получателя. Так например, если бы хеш-значение h и подпись $S_{privA}(h)$ не находились в шифрованном блоке $E_{k'}$, тогда была бы возможна атака анализа шифрованных сообщений по уже имеющемуся списку публичных ключей $\{pub_1, pub_2, ..., pub_n\}$, проверяющих их аутентичность $V_{pubi}(S_{privA}(h)) = h$.

Далее, если была бы известна криптографическая соль s и хеш-значение h, то можно было бы составить радужную таблицу наиболее часто встречающихся сообщений $\{m_1, m_2, ..., m_n\}$ с различными комбинациями участников i, j из множества всех узлов сети N по равенству $H_{mac(s)}(pub_i, pub_j, f(m_k)) = h$.

Плюс к этому, данная схема является самодостаточной на сетевом уровне работы QB-сетей в контексте заливочной маршрутизации, потому как позволяет обеспечивать идентификацию субъектов лишь и только при помощи асимметричной криптографии. Определить отправителя сообщения становится возможным посредством корректного расшифрования, т.е. при условии, когда получатель шифрованного сообщения располагает нужным приватным ключом.

Данный этап также предполагает, что сообщение $f(m)$ имеет статичную величину. Иными словами, при каждом вызове функции шифрования $E'_{(privA,pubB)}$, для всех m_i, m_j из $\{m_1,$

m_2, ..., m_n} соблюдается длина сообщения L от функции l, такая что $l \Rightarrow l(f(m_i)) = l(f(m_j)) = L$. Это становится возможным за счёт процедуры препроцессинга f, ограничивающей длину входного сообщения величиной L, и дополняющей длину входного сообщения до L. Целью такой процедуры становится защита от атак по анализу размера сообщений, при которых может выявляться структура передаваемого сообщения. Например, запросы чаще всего по размеру меньше чем ответы, передаваемые видео или аудио -файлы часто по размеру больше, чем обычные текстовые сообщения, системные / автоматические запросы меньше по размеру, чем вручную выполненные и т.д.

3.2. Второй этап шифрования

$$E''_k(m) = E_k(p(h) \,||\, (h = H_{mac(k)}(mv)) \,||\, (mv = (m \,||\, v))))$$

где k - ключ сети, p - функция доказательства работы, h - результат хеширования, m - открытое сообщение, v - пустые байты случайной длины полученные из *[RNG]*.

Второй этап шифрования придерживается подхода MAC-then-Encrypt (MtE), где сначала вычисляется MAC (Message Authentication Code), а далее сообщение $(m \,||\, v)$ и код h шифруются функцией E_k.

Данный этап шифрования выполняет несколько задач:

1. Разграничивает разные сети по ключу сети k, чтобы их нельзя было слить в одну общую систему. Это достигается преимущественно за счёт функции доказательства работы p, т.к. из-за неё становится более затратным перешифровывать весь трафик направленный из одной сети с ключом k_1 в другую сеть с ключом k_2,

2. В отличие от первого этапа шифрования, придающего информации m статичный вид при помощи функции f, второй этап напротив - придаёт информации случайный размер v с целью противодействия блокировок, направленных на анализ размерности сообщений.

Доказательство работы p определяется алгоритмом proof-of-work (PoW), где для конкретного хеш-значения h необходимо найти такое число i, чтобы результат $h_i = H(h \,||\, i)$ представлял собой битовый вектор с определённо заданным n-*ым* количеством нулей в качестве префикса *0000000000000000(n)...11001010*. Число n является сложностью работы. На текущий момент в сети Hidden Lake $n = 22$.

Из этого также стоит заметить, что ключ k может быть открытым параметром при условии, что нет необходимости противодействовать блокировкам. В таком случае, вторая задача становится побочной (опциональной), а ключ k - просто общеизвестной настройкой.

Подход Encrypt-then-MAC (EtM) не применяется в схеме шифрования по двум причинам:

1. Используется один ключ k для шифрования и аутентификации вместо двух ключей k_1, k_2 для этих задач. Если применить в этой ситуации подход EtM, тогда на один и тот же ключ k будет открыто два вектора нападения, вместо одного. Эту проблему можно было бы решить при помощи использования KDF (функции формирования ключей), которая бы позволила из одного ключа создать несколько. Тем не менее это усложнит общую схему шифрования, а также откроет дополнительный вектор нападения на саму KDF,

2. Учитывается принцип Хортона: “аутентифицировать нужно не то, что сказано, а то, что имеется в виду”. При успешной атаке на функцию вычисления MAC в подходе EtM, либо при неправильном распределении ключей k_1, k_2 может возникнуть ситуация когда аутентификация будет выдавать положительный результат на шифрованное сообщение, но сама процедура расшифрования будет некорректной. Если сообщение представляет собой хаотичный набор битов, то мы никогда не узнаем его истинность.

У подхода MtE безусловно существует недостаток в том, что перед тем как проверить целостность и аутентичность информации необходимо её расшифровать. У подхода EtM такой проблемы нет. Вследствие этого был также сформирован принцип криптографической обреченности Марлинспайком, который гласит: “если вы вынуждены выполнить любую криптографическую операцию до проверки имитовставки полученного сообщения, то это так или иначе, но неизбежно приведет к роковому концу”. Данный принцип противопоставляется принципу Хортона, когда речь заходит о выборе одного из двух подходов: MtE или EtM. Тем не менее выбор EtM обусловлен ещё тем, что принцип криптографической обреченности нарушается также на моменте первого, и куда более затратного, этапа шифрования.

4. Структурные параметры

Математические модели позволяют выявить корректность общей логики работы, но не позволяют определить безопасность конкретной реализации. Так например, мы можем лишь предполагать, что какая-либо функция или принимаемое ей значение будет безопасным. Тем не менее всё может зависеть от конкретной реализации и выбираемых параметров, поэтому на них также следует сделать упор.

1. Симметричная функция шифрования E_k определяется шифром AES-256 в режиме шифрования CFB, где $c_i = E_k(c_{i-1})$ *xor* m_i.

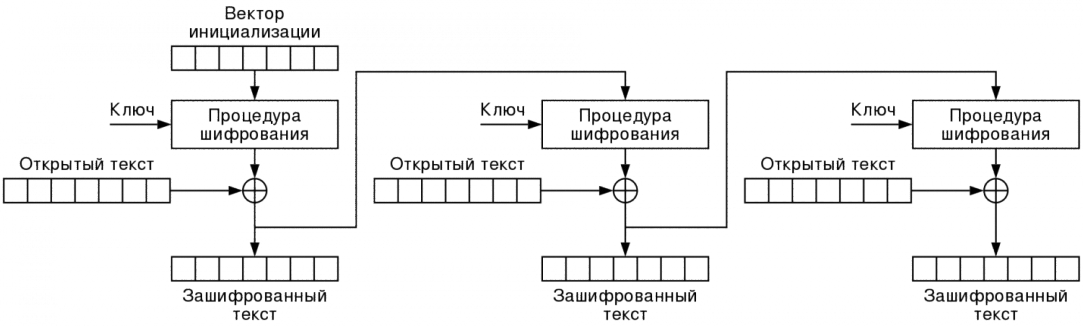

Рисунок 2. Режим шифрования CFB

Данный режим шифрования был выбран с учётом следующих моментов:

1. Режиму шифрования CFB не требуются дополнения (padding), как это требуется например режиму шифрования CBC. Вследствие этого

упрощается выставление статичного размера шифрованного сообщения, а также ликвидируются возможные атаки оракула,

2. Режим шифрования CFB не является поточным режимом шифрования, как например OFB или CTR. Вследствие этого, у CFB отсутствует проблема в лице повторяемости гаммы. Если IV (вектор инициализации) повторится, то это приведёт к куда меньшим проблемам безопасности, чем при OFB, CTR режимах,

3. Не использовался режим шифрования GCM по причине излишних операций аутентифицирования и хранения токенов. Первый и второй этап шифрования используют разные способы аутентификации сообщений. Вследствие этого, в плане гибкости использования, режим CFB становится более предпочтительным.

2. Асимметричная функция шифрования E_{pub} определяется алгоритмом RSA-4096 со схемой кодирования OAEP. Асимметричная функция подписания S_{priv} определяется также алгоритмом RSA-4096, но уже со схемой кодирования PSS, Для асимметричных функций расшифрования D_{pub} и подписания S_{priv} используется один ключ. По этой причине для одного ключа используются разные схемы кодирования,

3. Ключи размером в 256 для AES и в 4096 бит для RSA соответственно были выбраны с консервативной точки зрения. Ключ в 256 бит AES сможет успешно противостоять постквантовой криптографии. Ключ в 4096 бит RSA сможет противостоять постквантовой криптографии лишь на начальных этапах, потому как потребует от квантового компьютера примерно 8192 хорошо связанных между собой кубитов. Тем не менее размер в 4096 бит RSA был выбран также при сравнительном анализе безопасности с длиной ключа симметричного алгоритма к атакам полного перебора, где *4096bit RSA ≈ 140bit*. На текущий момент времени минимальной криптостойкостью считается длина ключа в 112 бит для симметричного алгоритма,

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Таблица 2. NIST: сравнение длин ключей при лобовой атаке

4. Алгоритм RSA был выбран за счёт его универсальности в плане шифрования и подписания, а также по причине того, что алгоритмы шифрования на базе эллиптической криптографии (ECIES) в настоящее время плохо стандартизированы и программно

реализованы. Плюс к этому, сеть Hidden Lake была написана на языке Go, где в стандартной библиотеке по умолчанию уже присутствует протестированный алгоритм RSA, что с точки зрения безопасности является более предпочтительным,

5. В новых версиях Go, начиная с go1.20, появился протокол Диффи-Хеллмана на эллиптических кривых ECDH, который чисто технически смог бы исполнять схожую первому этапу шифрования функцию. Тем не менее здесь есть три момента против:

1. Необходимо было бы использовать два алгоритма: один для распределения ключей ECDH, другой для подписания информации, например ECDSA. Как следствие, участник в сети идентифицировался бы двумя публичными ключами, вместо одного,

2. Публичный ключ ECDH отправителя передавался бы в открытом виде, что в некоторых прикладных использованиях являлось бы нежелательным свойством. У RSA такого недостатка нет. Связано это с тем, что в RSA приватным ключом информация способна расшифровываться, а в ECDH только генерироваться,

3. Пакет go-peer намеренно основывается на более ранней версии, а именно на go1.16, с целью поддержки наибольшего количества уже ранее написанных приложений.

6. Первый этап шифрования не представляет собой попытку какого-либо скрытия структуры сообщения. Такая задача возлагается лишь на второй этап шифрования,

7. Предполагается, что ключ сети на втором этапе шифрования редко меняется, обладает высокой энтропией и не является паролем. Следовательно, ключ сети не проходит сквозь какую бы то ни было KDF, но пропускается сквозь алгоритм хеширования SHA-256 для сжатия до фиксированного размера в 32 байт, чего требует алгоритм шифрования AES-256,

8. В качестве алгоритма MAC используется HMAC-SHA-256. Безопасность HMAC зависит полностью от используемой хеш-функции. Безопасность SHA-256 к коллизиям, как к наиболее сложной задаче, определяется 128 битами, исходя из атаки парадокса дней рождения.

9. Ни первый этап шифрования, ни второй не защищают от атаки повторных сообщений, где спустя определённый период времени *t* злоумышленником может быть транслировано вновь, сохранённое ранее им же, шифрованное сообщение. Такое сообщение будет полностью валидным, т.к. оно было сгенерированно истинным отправителем. Некоторые сети защищаются от данной атаки методом выставления меток времени. Hidden Lake поступает более радикально, сохраняя хеш сообщения в свою локальную БД. Такой подход имеет ряд плюсов:

1. Отпадает проблема в необходимости временной синхронизации отправителя и получателя. Проблема усложнялась бы ещё тем, что

большинство решений связанных с синхронизацией сводятся к централизованной архитектуре сети,

2. Отпадает проблема в проверка временных окон. Временные метки неминуемо порождают окна, зависимые от времени получателя, при которых сообщение может считываться. Окна порождают необходимость в дополнительной проверке одинаковых принятых сообщений в одном допускаемом периоде, а также в автоматическом удалении ранее принятых сообщений из старых периодов. Такие случаи могут усложнить конечную логику принятия сообщений

Подход сохранения хешей всех ранее принятых сообщений также имеет и свой минус в лице стремления к постоянному увеличению объёма базы данных. Как только появляется БД хешей, её ни в коем случае нельзя удалять или очищать, иначе все ранее принятые сообщения вновь смогут повториться, если злоумышленник будет применять атаку повторных сообщений.

Теперь, в такой парадигме вопрос ставится исходя из объёма и частоты постоянно генерируемой информации. Хеш сообщения в сети Hidden Lake - это хеш-значение $h = H_{mac(k)}$ полученное из информации на втором этапе шифрования E''_k. Размер хеша определяется криптографической хеш-функцией SHA-256, т.е. 32 байтами. Таким образом, одно принятое, либо отправленное сообщение будет увеличивать БД на 32 байта. Период генерации одного сообщения одним узлом равен 5 секундам. Если предположить, что в сети существует 10 узлов, каждый из которых в период равный 5 секундам генерирует одно шифрованное сообщение, тогда за 5 секунд будет сгенерировано и сохранено 320 байт, либо 64 байт в секунду. Далее, если умножить данный результат на количество секунд в минуте, минут в часе, часов в дне, дней в неделе, недель в месяце + 2 (30 дней), месяцев в годе, то получим, что за год 10-ью узлами БД будет увеличина на 1.73GiB информации, что по современным меркам является вполне допустимым значением.

5. Микросервисная архитектура

Анонимная сеть HIdden Lake представляет собой набор сервисов, каждый из которых выполняет свою конкретную задачу. Ядром сети Hidden Lake является сервис HLS (service), который непосредственно исполняет QB-задачу и все функции шифрования / расшифрования соответственно.

$$HLS = QB\text{-}net\ [\ E_{(k,privA,pubB)}(m) = E''_k(E'_{(privA,pubB)}(m))\ ]$$

Помимо сервиса HLS, в сети HL также существует ряд прикладных сервисов, по типу HLM (messenger), HLF (filesharer), ряд сервисов-помощников, по типу HLT (traffic), HLL (loader), HLE (encryptor) а также ряд адаптеров HLA, привязанных на текущий момент к сервисам common (тестовый сервис) и chatingar. В результате этого, сеть Hidden Lake можно представить как композицию нескольких сервисов.

$$Hidden\text{-}Lake = \Sigma^n_{i=1} APP_i \times HLS \times (HLT \times \Sigma^m_{j=1} HLA_j)^t$$

где APP - множество прикладных сервисов, *HLA* - множество адаптеров к сторонним сервисам, *t = 0 or 1* - параметр определяющий отсутствие / существование ретрансляции сообщений.

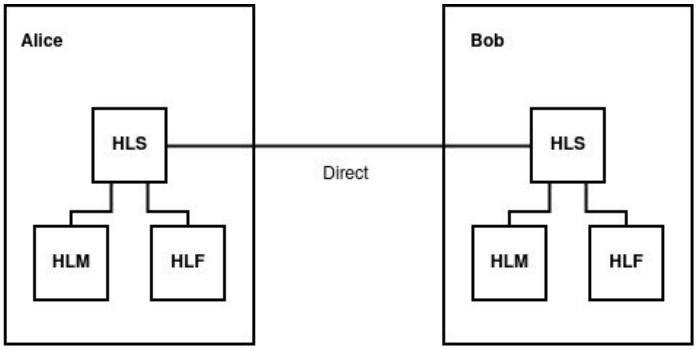

Рисунок 3. Классический p2p режим. Hidden-Lake = (HLM+HLF) × HLS

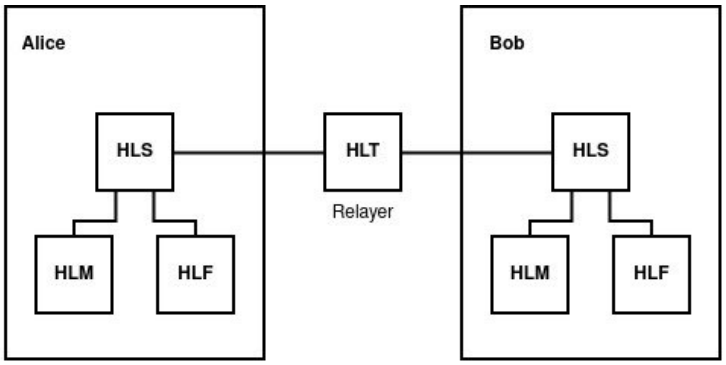

Рисунок 4. Режим ретрансляции. Hidden-Lake = (HLM+HLF) × HLS × HLT

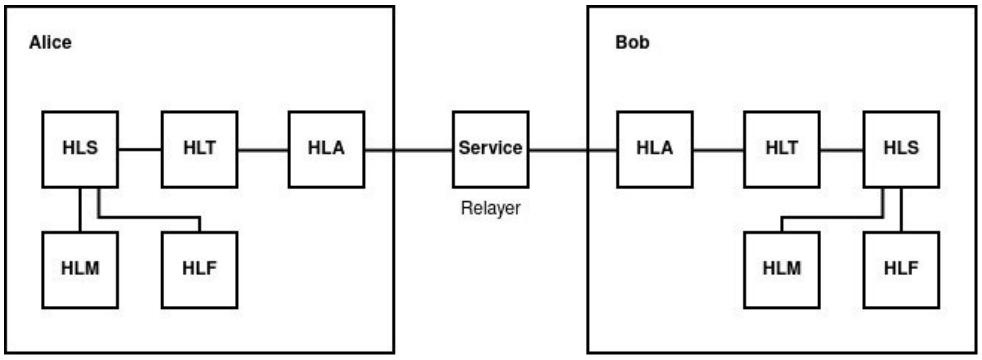

Рисунок 5. Режим сервиса связи. Hidden-Lake = (HLM+HLF) × HLS × HLT × HLA=Service

При отсутствии прикладных приложений, факта ретрансляции, и как следствие, адаптеров, сеть Hidden Lake становится равной сервису HLS: *Hidden-Lake = HLS*. Это есть минимальная характеристика, при которой HL ещё остаётся собой. При удалении же сервиса HLS, система перестаёт являться Hidden Lake сетью.

6. Заключение

В ходе нашей работы было разобрано функционирование анонимной сети Hidden Lake на основе её математических моделей. Были также представлены критерии выбора тех или иных криптографических примитивов в моменты проектирования и реализации.

Список литературы

1. Коваленко, Г. Теория строения скрытых систем [Электронный ресурс]. — Режим доступа: https://github.com/number571/go-peer/blob/master/docs/theory_of_the_structure_of_hidden_systems.pdf (дата обращения: 22.04.2024).
2. Коваленко, Г. Абстрактные анонимные сети [Электронный ресурс]. — Режим доступа: https://github.com/number571/go-peer/blob/master/docs/abstract_anonymous_networks.pdf (дата обращения: 22.04.2024).
3. Коваленко, Г. Монолитный криптографический протокол [Электронный ресурс]. — Режим доступа: https://github.com/number571/go-peer/blob/master/docs/monolithic_cryptographic_protocol.pdf (дата обращения: 22.04.2024).