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) [1] анонимная сеть с теоретической доказуемостью [2, с.49]. 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, с.144].

2. QB-задача

Задача на базе очередей (QB - Queue Based) [2 с.149] 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 представителями теоретической доказуемости в лице DC (Dining Cryptographers) [3, с.225] и EI (Entropy Increase) [2, с.165] - сетей. Формально QB-сеть можно описать системой следующего вида:

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

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*.

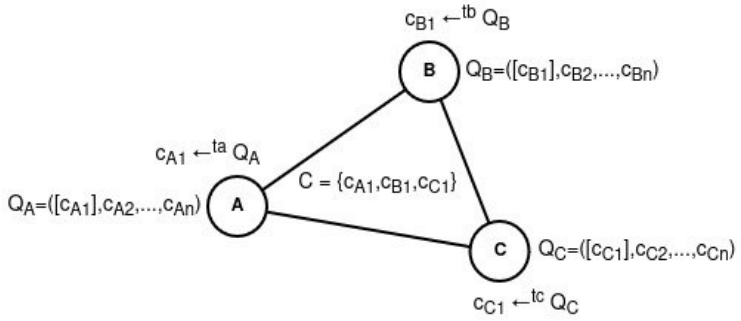

Рисунок 1. QB-сеть с тремя участниками A, B, C

Вышеуказанная система может быть представлена четырьмя состояниями:

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

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

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

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

Теорема 1. В множестве ключей $r \in R$, для любого шифртекста $c = E_r(v)$, существует такое r^{-1}, которое приводит к расшифрованию ложного сообщения $v = D_r^{-1}(c)$.

Доказательство 1. Множество R определяется разностью двух множеств $U \setminus K$, при $U = K \bigcup R$, представляющим собой конечное множество над входными значениями функции шифрования. Для любого $u \in U$ выполняется отображение в множество шифртекстов: $\bigcup E_u \rightarrow C$, при котором также будет существовать такое u^{-1}, выполняющее обратное отображение в множество открытых текстов: $\bigcup D_u^{-1} \rightarrow M$, исходя из того, что система $\sum(M, C, U, E, D)$ является шифром [4, с.75].

Анонимность QB-сетей базируется на сложности определения состояния шифрованного сообщения c, а именно чем оно является: $E_{ki}(m)$ *или* $E_r(v)$. Очерёдность сообщений Q в свою очередь гарантирует, что всегда будет существовать такое сообщение c, которое будет сгенерировано системой в период времени равный t, независимо от природы самого сообщения. При отсутствии истинных сообщений очередь Q можно рассматривать как очередь исключительно ложных сообщений $E_r(v)$. При появлении истинного сообщения $E_{ki}(m)$, таковое начинает заменять собой ложное $E_r(v)$ в определённый период времени t. Вследствие этого, QB-сеть становится генератором шума с функцией кратковременной замены случайного трафика на действительный, а неразличимость шифрованных сообщений друг от друга становится ключевым фактором анонимности.

Теорема 2. При наличии двух ключей шифрования k, r (порождающих соответственно истинные и ложные шифртексты), определение истинности выбранного шифртекста c_i из конечного множества $C = \{c_1, c_2, ..., c_n\}$ сводится к сложной задаче, если соблюдаются условия безопасности E, k, r параметров.

Доказательство 2. Задача истинности шифртекста c_i из множества C сводится к определению его принадлежности к двум состояниям: $E_k(m_i)$ и $E_r(v_i)$. При неизвестных параметрах k, r верхняя граница поиска определяется количеством итераций полного перебора равным $|K \bigcup R| = |U| = |\{u_1, u_2, ..., u_{|U|}\}|$, при $K \bigcap R = \varnothing$ соответственно, т.к. неизвестной переменной в данном случае остаётся принадлежность перебираемых значений u_i к множествам K и R (*Теорема 1*). При известном u, но неизвестном u^{-1} соответственно, проблема определения истинности шифртекста c_i сводится к одной из асимметричных задач [3, с.378][3, с.386].

Итого, анонимность QB-сетей определяется не только разрывом связи между отправителем и получателем для глобального наблюдателя, но и также 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 $T\{t_i\}$ 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.

Алгоритм 1. Функционирование участника *i* в системе *QB-net* на языке псевдокода

ВВОД: очередь Q_i, период t_i, функции E, D, ключи шифрования k_i, r_i
ВЫВОД: подмножество шифртекстов $C_i \in C$
thread-1. *(* Генерация истинных шифртекстов *)*
for (;;) {
    $k_j \leftarrow$ INPUT_STREAM (Ввод ключа получателя *)*
    $m \leftarrow$ INPUT_STREAM (Ввод открытого текста *)*
    $Q_i \leftarrow (c = E_{kj}(m))$
}
thread-2. *(* Генерация ложных шифртекстов *)*
for (;;) {
    if ($Q_i = \varnothing$) {
        $v \leftarrow$ RANDOM_STREAM (Получение ложного текста от КСГПСЧ *)*
        $Q_i \leftarrow (c = E_{ri}(v))$
    }
}
thread-3*. (* Отправление шифртекстов в сеть *)*
for (a = 1; ; a = a + 1) {
    sleep(t_i) (Ожидание периода *)*
    $c_a \leftarrow Q_i$, где $c_a \in C_i$
    QB-net $\leftarrow c_a$ (Запись шифртекста в сеть *)*
}
thread-4*. (* Принятие шифртекстов из сети *)*
for (b = 1; ; b = b + 1) {
    $c_b \leftarrow$ QB-net, где $c_b \in C \backslash C_i$ (Чтение шифртекста из сети *)*
    if valid($m = D_{ki}^{-1}(c_b)$) { (Проверка корректности расшифрования *)*
        OUTPUT_STREAM $\leftarrow$ m (Вывод полученного открытого текста *)*
    }
}

В QB-сетях есть также ряд интересных и не совсем очевидных 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 подобным способом 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.

Теорема 3. Теоретически доказуемая анонимность *QB-net* системы основывается на систематичности порождения множества шифртекстов $C = C^r \bigcup C^k = \{c_1, c_2, ..., c_n\}$ и на неразличимости его подмножеств относительно истинных C^k и ложных C^r шифртекстов.

Доказательство 3. Пусть $C^r = \{c_1^r, c_2^r, ..., c_m^r\}$ есть множество ложных шифртекстов, а $C^k = \{c_1^k, c_2^k, ..., c_n^k\}$ напротив есть множество истинных шифртекстов при $m, n \geqslant 0$ соответственно, тогда при их объединении создаётся множество всех шифртекстов: $C^r \bigcup C^k = \{c_1^r, c_2^r, ..., c_m^r, c_1^k, c_2^k, ..., c_n^k\} = \{c_1, c_2, ..., c_{m+n}\} = C$. Пусть под задачей деанонимизации $P_{(d)}$ будет далее пониматься однозначное нахождение факта существования или отсутствия истинного сообщения ($n > 0$?) в множестве шифртекстов C. Задача деанонимизации, в свою очередь, опирается на две другие задачи: неразличимости $P_{(i)}$ и систематичности $P_{(s)}$. При решении одной из двух подзадач, задача деанонимизации будет считаться также выполнимой, что можно выразить дизъюнктивной формой: $P_{(d)} = P_{(i)} \vee P_{(s)}$.

1. Задача неразличимости $P_{(i)}$ может быть определена двумя возможными ситуациями: 1) либо способом нахождения истинного шифртекста $c_i = E_k(m_i) \in C^k$; 2) либо, напротив, способом доказательства отсутствия истинных шифртекстов $c_i \notin C^k = \varnothing$. Задача становится тривиальной при отсутствии шифртекстов в общем, т.к. $C = \varnothing \rightarrow C^k = \varnothing$. Если же $C \neq \varnothing$, тогда задача сводится к проблеме соотношения шифртекстов $c_i \in C$ к их первоначальным подмножествам: C^r или C^k , что, как было показано ранее (*Теорема 2*), является сложной задачей.

2. Задача систематичности $P_{(s)}$ может быть определена нахождением дополнительных связей в механизме генерации шифртекстов $C = \{c_1, c_2, ..., c_n\}$. Если взять частный случай $C = \varnothing$, то связность шифртекстов будет априори отсутствовать, что, в свою очередь, показывает вводное отличие задачи систематичности от неразличимости. Далее, пусть $|C| = 1$, т.е. $C = \{c\}$, тогда, задача систематичности будет сводиться к вопросу: «вследствие какого события x был получен шифртекст c?». Если событие x не имеет связей с каким-либо открытым текстом m (полученным, получаемым, отправленным или отправляемым), т.е. шифртекст c не был создан вследствие появления m как события, тогда x следует рассматривать как *независимое* событие. Если предположить, что существует некий алгоритм A, генерирующий шифртексты c_i посредством независимого события x, т.е. $c_i \leftarrow A(x)$, где $c_i \in C$, и при этом $C = C^r$, тогда в механизме генерации априори будут отсутствовать дополнительные связи кроме основной связи в лице события x, т.к. $c_i \notin C^k = \varnothing$. Если же предположить обратное: $C = C^r \bigcup C^k$, где $C^k \neq \varnothing$, то сохранение единой связи, в лице независимого события x, становится возможным тогда и только тогда, когда шифртексты $c_i \in C^k$ будут генерироваться посредством того же алгоритма A, что и шифртексты подмножества C^r. Иными словами, получение истинных шифртекстов на базе алгоритма A с независимым событием x: $(c_i = E_k(m_i)) \leftarrow A(x)$ должно определяться точно также, как и получение ложных шифртекстов $(c_i = E_r(v_i)) \leftarrow A(x)$. В результате

приверженность механизма генерации к общему независимому событию x не позволяет истинному событию m сформировать собственную (дополнительную) связь генерации.

Следствие 3.1. Конечное множество шифртекстов C можно рассматривать как результат поэтапной генерации подмножеств $C_1 = \{c_1\}$, $C_2 = \{c_1, c_2\}$, ..., $C_n = \{c_1, c_2, ..., c_n\}$ и как завершение сетевой коммуникации в общем. Таким образом, при выполнении условий неразличимости и систематичности множеством $C = C_n$, подмножества $C_i \subseteq C_n$ продолжают в равной степени наследовать их выполнение.

Следствие 3.2. Пусть t есть период генерации порождаемых подмножеств C_i. (*Следствие 3.1*). В таком случае, t есть также вводное условие и независимое событие для алгоритма генерации шифртекстов: $c_i \leftarrow A(t)$. Раз период t базируется на алгоритме A и при этом порождение шифртекстов не связано с открытыми текстами, тогда его варьируемая характеристика в лице статичности ($t = T$) или динамичности ($t \in [l;k]$) не способна воздействовать на качество анонимности.

Следствие 3.3. Теоретически доказуемая анонимность QB-задачи не зависит от какого-либо одного конкретного алгоритма генерации шифртекстов A, и как следствие, множество участников системы $\{1, 2, ..., n\}$ способно использовать множество различных алгоритмов $\{A_1, A_2, ..., A_n\}$, по причине достаточности условия в лице независимости событий при генерации шифртекстов.

Пример 3.1. В QB-задаче под алгоритмом A наиболее часто понимается периодичность генерации шифртекстов на базе очередей с вводным условием t (периодом). Хоть это и наиболее практичный алгоритм, он всё же не единственен. Так например, под алгоритмом генерации A' может пониматься также формирование шифртекста $c_i \leftarrow A'(n)$ на основе *n-ого* количества принятых шифртекстов системой от других участников. Отсутствие зависимости от открытых текстов закономерно приводит к аналогичной теоретической доказуемости и указывает на возможность применения системой нескольких различных алгоритмов.

Пример 3.2. Отправление всех шифртекстов $C = \{c_1, c_2, ..., c_n\}$ сети за один раз также является алгоритмом $C \leftarrow A(C)$, хоть и специфичным по своей природе, т.к. исключает какую бы то ни было интерактивность в лице запросов и ответов. Данный пример интересен и тем, что с некоторой вероятностью все шифртексты во множестве C обязательно должны оставаться ложными. В противном случае, будут нарушаться сразу два условия: неразличимости, где известным будет становиться неравенство $C^k \neq \emptyset$, и систематичности, где алгоритм генерации будет зависим от открытых текстов: $A(C) = A(m)$.

Пример 3.3. Небольшое различие алгоритма A при генерации истинных и ложных шифртекстов может нарушить систематичность генерации. Предположим, что задан статичный период t, как условие алгоритма, но при этом генерация шифртекстов c_i занимает продолжительное время x, такое что $0 < x < t$. Далее, если предположить, что ложные шифртексты генерируются на моменте отправления $(c_i = E_r(v_i)) \leftarrow A(t)$, в обход очерёдности и по причине отсутствия в очереди каких-либо сообщений, тогда шифртекст c_i отправится спустя время равное $x+t$. В свою очередь, если истинные сообщения генерируются до

момента отправления, т.е. сначала помещаются в очередь $(c_i = E_k(m_i)) \leftarrow A(tQ)$, то время их отправления будет равно *t* на следующем этапе генерации. Таким образом, систематичность алгоритма нарушается разными вводными условиями (независимыми событиями) при генерации истинных и ложных шифртекстов соответственно.

Пример 3.4. Различие между алгоритмами *A(t)* и *A(tQ)* определяется лишь отсутствием или существованием очереди сообщений. Теоретически оба алгоритма основаны на независимых событиях, но практически они отличаются тем, что при *A(t)* генерация истинных шифртекстов будет более затруднительной, т.к. она приводит к ручному их созданию на конкретном периоде времени *t* по причине отсутствия механизма отложенности сообщений до заданного условия. Алгоритм с очередью *A(tQ)*, напротив, позволяет генерировать и сохранять сообщения в любой момент времени, вне зависимости от заданного периода *t*. Именно поэтому QB-задача именуется как Queue, а не Time Based.

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 $2n$ [3, с.278],

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 [5, с.46].

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

[6, с.398], 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 [2, с.62], 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, I2P, Mixminion [7][8][9]:

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

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 [10]. 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 с более сложными активными атаками [2, с.159], 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. Вследствие этого недостатка реализация стриминговых сервисов и видео / аудио звонков становится либо очень затруднительной задачей, либо вовсе невозможной.

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.

Алгоритм 2. Фильтрация сообщений в F2F-сетях на языке псевдокода

ВВОД: множество друзей F, отправитель s, функция-обработчик h, сообщение m
ВЫВОД: обработанное сообщение h(m) ИЛИ завершение алгоритма
if (s ∉ F) {
    return (Завершение алгоритма *)*
}
return h(m) (Обработка сообщения *)*

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 множестве периодов $D = \{(1+k)t_i, (2+k)t_i, ..., (d-1+k)t_i\}$, 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. Являлся ли прослушиваемый участник i инициатором запросов в множестве D?
2. С какими узлами общался прослушиваемый участник i при множестве периодов D?
3. Может ли участник i намеренно генерировать ложные сообщения в роли истинных?

2.2. Сравнение с другими задачами

По своим характеристикам QB-задача наиболее близка к DC-задаче из-за следующих особенностей: теоретически доказуемая анонимность, периодичность генерации сообщений, принадлежность к второму вектору развития анонимных коммуникаций [2, с.71], сложность масштабирования. Отличия QB от DC-сетей, с положительной точки зрения, присутствуют следующие: периодичность генерации может иметь динамичную величину, анонимность не зависит от выстроенных связей с другими участниками, более простая программная реализация. Негативное отличие определяется отсутствием полиморфизма информации. Более детальное и общее сравнение QB-задачи с другими задачами анонимизации как теоретическими, так и практическими представлено в *Таблице 1*.

	QB	**EI**	**DC**	**Onion**	**Proxy**
Теоретическая доказуемость	+	+	+	-	-
Накопительный эффект анонимности	-	+	-	-	-
Полиморфизм информации	-	+	+	+	-
Вероятностная маршрутизация	-	+	-	+/-	+/-
Периодичность генерации сообщений	+/-	-	+	-	-
Независимость анонимности от связей	+	-	-	-	-
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 анонимизации

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 [5, с.190]. 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 (асимметричная + симметричная криптография) [2, с.125]. 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 получатель 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 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 [11].

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 блокировкам, направленных на анализ размерности сообщений и их структуру.

Доказательство работы p определяется алгоритмом proof-of-work (PoW) [12], 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 именуется сложностью работы.

Стоит также заметить, что ключ шифрования k может являться открытым параметром, если отсутствует необходимость противодействовать блокировкам. В таком случае, вторая задача становится побочной (опциональной), а ключ k - просто общеизвестной настройкой для разграничения сетей.

Подход Encrypt-then-MAC (EtM) не применяется в схеме шифрования второго этапа 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” [5, с.130]. 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” [13, с.93]. 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. Микросервисная архитектура

Анонимная сеть HIdden Lake представляет собой набор сервисов, каждый из которых выполняет свою конкретную задачу [19]. 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), HLR (remoter), 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.

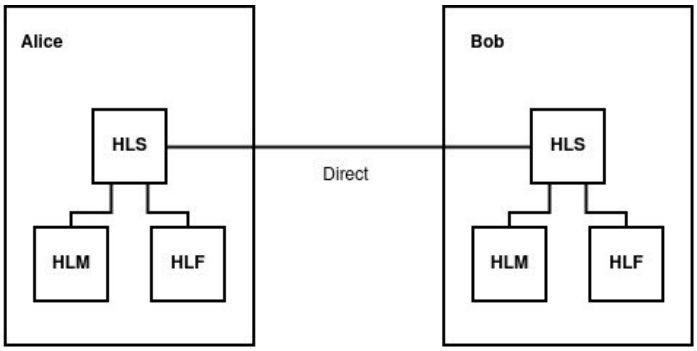

Рисунок 4. Классический p2p режим. Hidden-Lake = (HLM+HLF) × HLS

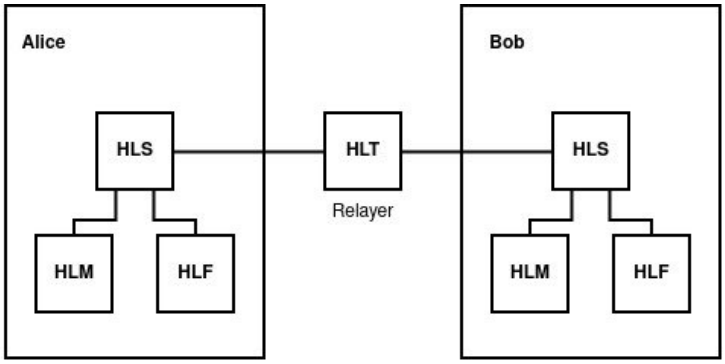

Рисунок 5. Режим ретрансляции. Hidden-Lake = (HLM+HLF) × HLS × HLT

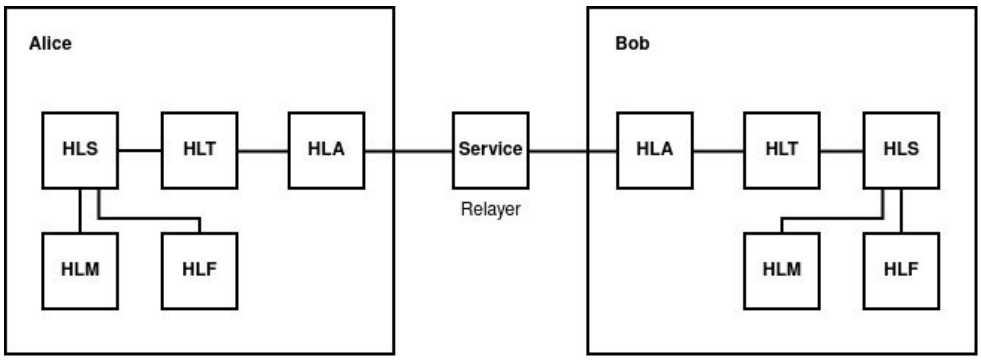

Рисунок 6. Режим сервиса связи. Hidden-Lake = (HLM+HLF) × HLS × HLT × HLA=Service

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.

5. 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 это ничего не говорит о конкретной реализации, выбираемых параметрах, модели угроз и подходах проектирования. Таким образом, необходимо уделить внимание не только общему описанию работы сети, но и подробному изложению её структурных параметров.

1. Реализация сети Hidden Lake придерживается принципа наименьшего количества зависимостей: «Если какую-то функцию можно реализовать относительно легко стандартной библиотекой языка, то следует отдать предпочтение именно такому подходу, даже если итоговое решение не будет являться лучшим из множества возможных». Данный принцип позволяет уменьшить количество используемых зависимостей, тем самым сужая вектор атак на сеть через сторонние реализации библиотек и приложений,

2. Симметричная функция шифрования E_k определяется блочным алгоритмом AES с длиной ключа 256 бит в режиме шифрования CFB (режим обратной связи по шифртексту), где $c_0 = IV$, $c_i = E_k(c_{i-1})$ *xor* m_i.

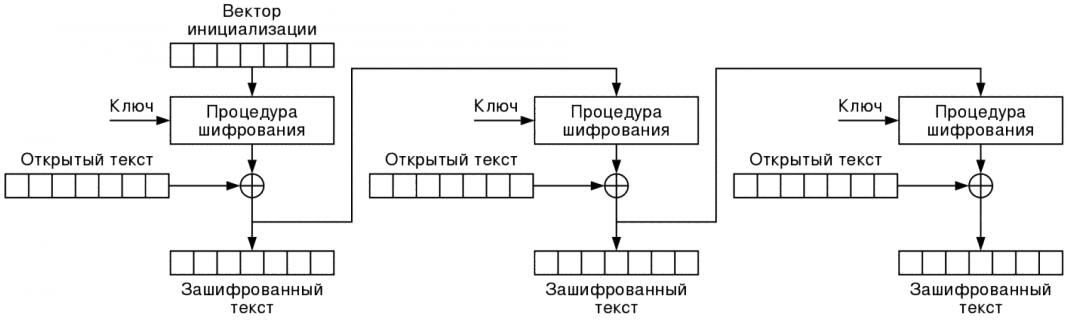

Рисунок 2. Режим шифрования CFB

Данный режим шифрования был выбран с учётом следующих моментов:

1. Режиму шифрования CFB не требуются дополнения (padding), как это требуется например режиму шифрования CBC (режим сцепления блоков шифртекстов). Вследствие этого упрощается выставление статичного размера шифрованного сообщения, а также ликвидируются возможные атаки оракула [14],

2. Режим шифрования CFB не является поточным режимом шифрования, как например OFB (режим обратной связи по выходу) или CTR (режим счётчика). 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 [5, с.93],

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

3. Асимметричная функция шифрования E_{pub} определяется алгоритмом RSA с длиной ключа 4096 бит со схемой кодирования OAEP (оптимальное асимметричное шифрование с дополнением). Асимметричная функция подписания S_{priv} определяется также алгоритмом RSA-4096, но уже со схемой кодирования PSS (схема вероятностной подписи), Для асимметричных функций расшифрования D_{pub} и подписания S_{priv} используется один ключ. По этой причине для одного ключа используются разные схемы кодирования [5, с.257],

4. Алгоритм RSA был выбран за счёт его универсальности в плане шифрования и подписания, а также по причине того, что алгоритмы шифрования на базе ECIES (интегрированная схема шифрования на эллиптической кривой) в настоящее время плохо стандартизированы и программно реализованы. Плюс к этому, сеть Hidden Lake была написана на языке программирования Go [17], где в стандартной библиотеке по умолчанию уже присутствует протестированный алгоритм RSA, что с точки зрения безопасности и принципа наименьшего количества зависимостей является более предпочтительным,

5. 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 [15, с.131]. Ключ в 4096 бит RSA сможет противостоять постквантовой криптографии лишь на начальных этапах, потому как потребует от квантового компьютера примерно $2n+3 = 8195$ хорошо связанных между собой кубитов [15, с.134]. 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* [16, с.54]. 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

6. 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[1] намеренно основывается на более ранней версии, а именно на go1.16, с целью поддержки наибольшего количества уже ранее написанных приложений,

7. Первый этап шифрования не представляет собой попытку какого-либо скрытия структуры сообщения. Такая задача возлагается лишь на второй этап шифрования в целях разделения логики на сохранение статичного и приобретение динамичного размера в пределах одного сообщения,

8. 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 функцию формирования ключа (KDF), но пропускается сквозь алгоритм хеширования SHA-256 для сжатия до фиксированного размера в 32 байт, чего требует алгоритм шифрования AES-256,

9. В качестве алгоритма имитовставки (MAC) используется HMAC-SHA-256. Безопасность HMAC зависит от используемой им хеш-функции [18, с.168]. Безопасность SHA-256 может быть определена стойкостью к коллизиям в 128 бит, исходя из атаки парадокса дней рождения [5, с.52],

10. В отличие от классического описания QB-сетей, в реализации проекта go-peer структура очередей Q представлена двумя очередями: очередью истинных Q_k и очередью ложных Q_r сообщений далее сливающихся в одну. Необходимость в двух очередях обуславливается нуждой в фоновой генерации ложных сообщений, чтобы при достижении периода t очередь Q_r преимущественно была непустой. В свою очередь такая нужда связана непосредственно с алгоритмом доказательства работы, который значительно замедляет шифрование сообщений, вследствие чего выставленный период генерации t может быть расширен. Хоть ситуация в таком случае и схожа с *Примером 3.3*, тем не менее, систематичность алгоритма здесь не нарушается, т.к. генерация истинных и ложных сообщений происходит не на моменте их отправления, а на моменте помещения их в очередь,

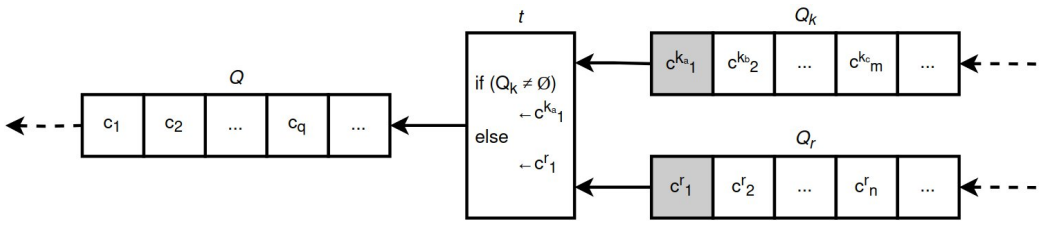

Рисунок 3. Схема двойной очереди сообщений в проекте go-peer

11. Ни первый этап шифрования, ни второй не защищают от атаки воспроизведения повторных сообщений [5, с.279], где спустя определённый период времени t

[1] Исходный код анонимной сети Hidden Lake содержится в репозитории проекта go-peer: https://github.com/number571/go-peer/tree/master/cmd/hidden_lake. Для сети Hidden Lake проект go-peer является фреймворком по причине их общей принадлежности к кодовой базе и релизным версиям.

злоумышленником может быть транслировано вновь, сохранённое ранее им же, шифрованное сообщение. Такое сообщение будет полностью верным, т.к. оно было сгенерировано 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 проверке 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 воспроизведения 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 уникальное 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 полученный 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 увеличена на ~1.73GiB[2] информации, что по современным меркам является вполне допустимым значением,

12. Сообщения в первом этапе шифрования имеют статичный размер равный 8192 байт, из которых 1658 байт уделяется заголовочным данным: вектор инициализации (16B), хеш (32B), подпись (512B), публичный ключ (526B), соль (32B), ключ шифрования (512B), размеры в байтах: хеша, подписи, публичного ключа, соли, ключа шифрования, шифрованного блока, данных, полезной нагрузки (28B). Второй этап шифрования добавляет минимум 68 байт (при отсутствии пустых байт), из которых: вектор инициализации (16B), доказательство работы (8B), хеш (32B), маска (4B), количество пустых байт (4B), размер сообщения после первого этапа шифрования (4B). Размер сообщения после полного

[2] (64 [байт] * 60 [секунд] * 60 [минут] * 24 [часов] * 7 [дней] * 4 [недель] + 2 [дня]) * 12 [месяцев] / 1024^3 [GiB] = 1.7303467020392418 ~= 1.73GiB

шифрования становится равен 8260 байт из которых 6534 байт являются полезной нагрузкой. Таким образом, если период генерации сообщений равен 5 секундам, тогда за одну секунду сеть может передать ~1306 значимых байт. Если прикладное приложение предполагает коммуникацию типа «запрос-ответ», тогда вследствие произведённого запроса неминуемо начнётся этап ожидания ответа, что приведёт к двойному уменьшению пропускной способности до ~653 значимых байт в секунду из-за увеличенного интервала ожидания равного ~10 секундам. Итого, если будет присутствовать задача передачи файла размером 1MiB по сети, то передача файла будет занимать приблизительно от ~13.4 до ~26.8 минут (~803 и ~1606 секунд соответственно),

13. Модель угроз анонимной сети Hidden Lake ограничивается исключительно защитой сетевых коммуникаций между её участниками, что, в свою очередь, предполагает отсутствие каких-либо дополнительный мер, действий и условий направленных на защиту приватных ключей, баз данных, конфигурационных файлов или взаимодействий сервисов между собой в условиях локальной среды исполнения. Вследствие этого, предпринимаемые меры для обеспечения безопасности локального окружения, в условиях существования чувствительной информации, должны быть возложены на более низкие уровни взаимодействия, как например: выставление прав доступа к файлами и процессам, изолирование программ посредством использования виртуального окружения, программное и аппаратное полнодисковое шифрование, ограничение физического доступа к аппаратным средствам для третьих лиц и т.д.,

14. Сеть Hidden Lake имеет множество настроек, в первую очередь направленных на её адаптацию в других системах. Так например, настройка f2f_disabled=true отключает опцию F2F, позволяя тем самым принимать сообщения от всех узлов, настройка queue_period_ms=0 способна отключать QB-задачу, позволяя тем самым не генерировать много трафика, настройки rand_queue_period_ms>0 и rand_message_size_bytes>0 делают период генерации и размер передаваемых сообщений динамичным параметром, что позволяет скрывать закономерные особенности генерируемого трафика от блокировочных систем. Все эти настройки, помимо положительных свойств, привносят и ряд негативных последствий, из-за чего по умолчанию отключены в конфигурационных файлах.

6. Заключение

В ходе нашей работы было разобрано функционирование анонимной сети Hidden Lake на основе её математических моделей. Были приведены основные сервисы сети, а также их взаимодействия друг с другом в парадигме микросервисной архитектуры. Были представлены критерии выбора алгоритмов шифрования, подписания хеширования, их параметров и настроек в моменты проектирования и реализации сети.

Список литературы

1. Popescu, B., Crispo, B., Tanenbaum, A. Safe and Private Data Sharing with Turtle: Friends Team-Up and Beat the System [Электронный ресурс]. — Режим доступа: https://web.archive.org/web/20070316085325/http://www.turtle4privacy.org/documents/sec_prot04.pdf (дата обращения: 04.07.2024).

2. Коваленко, Г. Общая теория анонимных коммуникаций. Второе издание / Г. Коваленко. — [б. м.]: Издательские решения, 2023. - 208 с.
3. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы и исходные коды на языке С / Б. Шнайер. — СпБ.: ООО «Альфа-книга», 2018. - 1040 с.
4. Алферов, А., Зубов, А., Кузьмин, А., Черемушкин, А. Основы криптографии: Учебное пособие / А. Алферов, А. Зубов, А. Кузьмин, А. Черемушкин. — М.: Гелиос АРВ, 2002. - 480 с.
5. Шнайер, Б., Фергюсон, Н. Практическая криптография / Б. Шнайер, Н. Фергюсон. - М.: Издательский дом «Вильямс, 2005. - 420 с.
6. Таненбаум, Э., Уэзеролл, Д. Компьютерные сети / Э. Таненбаум, Д. Уэзеролл. — СПб.: Питер, 2017. - 960 с.
7. Perry, M. Securing the Tor Network [Электронный ресурс]. — Режим доступа: https://www.blackhat.com/presentations/bh-usa-07/Perry/Whitepaper/bh-usa-07-perry-WP.pdf (дата обращения: 04.07.2024).
8. Astolfi, F., Kroese, J., Oorschot, J. I2P - Invisible Internet Project [Электронный ресурс]. — Режим доступа: https://staas.home.xs4all.nl/t/swtr/documents/wt2015_i2p.pdf (дата обращения: 04.07.2024).
9. Danezis, G., Dingledine, R., Mathewson, N. Mixminion: Design of a Type III Anonymous Remailer Protocol [Электронный ресурс]. — Режим доступа: https://web.archive.org/web/20170312061708/https://gnunet.org/sites/default/files/minion-design.pdf (дата обращения: 04.07.2024).
10. Reiter, M., Rubin, A. Crowds: Anonimity for Web Transactions [Электронный ресурс]. — Режим доступа: https://www.cs.utexas.edu/~shmat/courses/cs395t_fall04/crowds.pdf (дата обращения: 04.07.2024).
11. Ишкуватов, С. Способ и алгоритм определения типа трафика в шифрованном канале связи [Электронный ресурс]. — Режим доступа: https://cyberleninka.ru/article/n/sposob-i-algoritm-opredeleniya-tipa-trafika-v-shifrovannom-kanale-svyazi (дата обращения: 04.07.2024).
12. Накамото, С. Биткойн: система цифровой пиринговой наличности [Электронный ресурс]. — Режим доступа: https://bitcoin.org/files/bitcoin-paper/bitcoin_ru.pdf (дата обращения: 04.07.2024).
13. Хлебников, А. OpenSSL 3: ключ к тайнам криптографии / А. Хлебников. — М.: ДМК Пресс, 2023. - 300 с.
14. Heaton, R. The Padding Oracle Attack [Электронный ресурс]. — Режим доступа: https://robertheaton.com/2013/07/29/padding-oracle-attack/ (дата обращения: 04.07.2024).
15. Граймс, Р. Апокалипсис криптографии / Р. Граймс. — М.: ДМК Пресс, 2020. - 290 с.
16. Barker, E. Recommendation for Key Management [Электронный ресурс]. — Режим доступа: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf (дата обращения: 23.07.2024).
17. Донован, А., Керниган, Б. Язык программирования Go / А.А. Донован, Б.У. Керниган. — М.: ООО «И.Д. Вильямс», 2018. - 432 с.
18. Омассон, Ж. О криптографии всерьез. Практическое введение в современное шифрование / Ж. Омассон. — М.: ДМК Пресс, 2021. - 328 с.
19. Introduction into Microservices [Электронный ресурс]. — Режим доступа: https://web.archive.org/web/20160611033551/https://specify.io/concepts/microservices (дата обращения: 04.07.2024).