Анонимная сеть «Hidden Lake»

Коваленко Геннадий Александрович (@number571)

Июнь 15, 2025

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; стек протоколов gp/12; сеть hidden lake; постквантовая криптография;

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[1], в сравнении с другими представителями теоретической доказуемости в лице DC (Dining Cryptographers) [3, с.225] и EI (Entropy Increase) [2, с.165] - сетей.

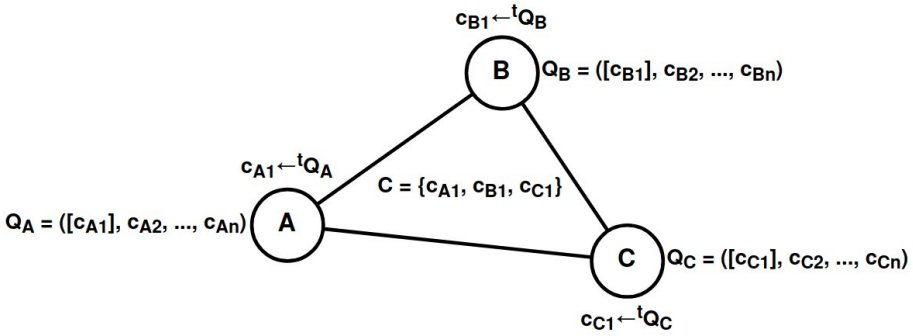

Рисунок 1. QB-сеть с тремя участниками A, B, C

Определение 1. Под задачей анонимизации следует понимать криптографический протокол, целью которого является сокрытие истинной связи между $(s \in S, r \in R)$ от $o \in O$ в системе $\Sigma(S, R, O)$, *где* S – множество отправителей, R – множество получателей, O – множество наблюдателей. В частных случаях, наблюдатель $o \in O$ может быть представлен одним из абонентов непосредственной коммуникации, т.е. $o \stackrel{?}{=} s$, $o \stackrel{?}{=} r$.

Формальным языком 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.

Вышеуказанная система может быть представлена четырьмя состояниями:

[1] Анонимная сеть M-A занимает 100 строк кода на языке программирования Go, используя исключительно стандартную библиотеку. Репозиторий: https://github.com/number571/micro-anon.

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. $Q_i \leftarrow (c = E_{ri}(v))$, *если* $Q_i = \varnothing$, *где* $r_i \notin K$, $c \in C$. Ложное сообщение v шифруется ключом без получателя $r_i \notin K$. Результат шифрования $c = E_{ri}(v)$ помещается в очередь Q_i,

3. $c \leftarrow^t Q_i$, *где* $t \in T$, $c \in C$. В каждый период времени t из очереди Q_i берётся шифрованное сообщение $c \in \{E_{kj}(m), E_{ri}(v)\}$ и отправляется всем участникам сети,

4. $m' = D_{ki}^{-1}(c)$, *где* $c \in C$. Каждый участник пытается расшифровать принятое им шифрованное сообщение c своим ключом k_i^{-1}. Если шифртекст 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 $r_j \notin K$).

Лемма 1 [Л1]. Для любого элемента $r \in R$, при шифртексте $c = E_r(v)$, существует всегда такое r^{-1}, которое приводит к расшифрованию ложного сообщения $v = D_r^{-1}(c)$.

Доказательство Л1. Множество R определяется разностью двух множеств $U \setminus K$, при $U = K \bigcup R$, представляющим собой множество ключей 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 $\Sigma(M, C, U, E, D)$ является шифром [4, с.75].

Анонимность QB-сетей базируется на сложности определения состояния шифрованного сообщения c, а именно чем оно является: $E_k(m)$ *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 $E_k(m)$, 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.

Лемма 2 [Л2]. При наличии в системе 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 задаче неразличимости шифртекстов по открытым текстам и ключам шифрования.

Доказательство Л2. Задача определения истинности шифртекста c_i из множества C сводится к установлению его принадлежности к одному из двух состояний: $E_k(m_i)$ или $E_r(v_i)$. В такой концепции, функция шифрования E должна обладать свойством неразличимости шифртекстов: 1) по открытым текстам m_i, v_i; 2) по используемым ключам шифрования[2] k, r.

[2] У некоторых представителей асимметричных алгоритмов шифрования отсутствует гарантия

Первое свойство гарантирует, что даже если наблюдателю будут известны тексты или шаблоны истинных / ложных сообщений, то связать их с результатом шифрования (шифртекстом) будет вычислительно неосуществимо. Второе свойство гарантирует, что даже если наблюдателю будут известны публичные ключи шифрования, то связать их с результатом шифрования будет также вычислительно неосуществимо. В итоге, при использовании надёжной функции E и параметров $k \in K$, $r \in R$, верхняя граница поиска начинает определяться количеством итераций полного перебора равным $|K \bigcup R| = |U| = |\{u_1, u_2, ..., u_{|U|}\}| \rightarrow D_u(c)$, при $K \bigcap R = \emptyset$ соответственно, за счёт принадлежности всех перебираемых значений u_i к общей шифр-системе *(Лемма 1*).

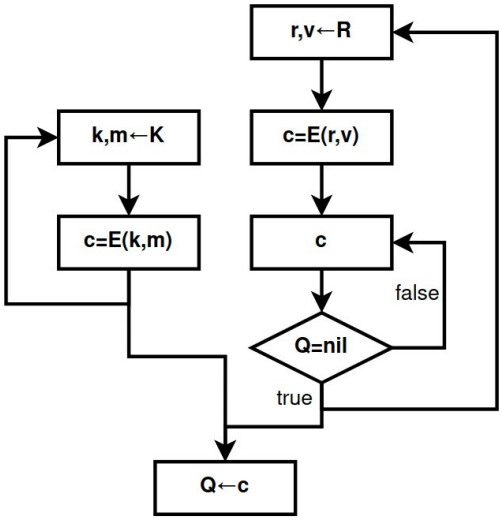

Рисунок 2. Алгоритм заполнения очереди в QB-задаче, где
R – генератор ложных сообщений, K – генератор истинных сообщений

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 $T\backslash\{t_i\}$ 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

неразличимости шифртекстов по ключам шифрования, например – RSA. Публичный ключ в RSA представлен двумя значениями $\{e, N\}$, где e – публичная экспонента (чаще всего являющаяся константой $2^{16}+1$), N – модуль, представленный произведением двух простых чисел. Предположим, что существует два публичных ключа $pub_1 = \{e, N_1\}$ и $pub_2 = \{e, N_2\}$. Один из двух модулей всегда будет больше второго, т.е. $N_1 > N_2$ (в противном случае, ключи одинаковы). В результате, если наблюдателем был получен шифртекст $c = m^e \bmod N_?$ и при этом, $c > N_2$, то это будет означать, что шифртекст c был создан на основе модуля N_1, т.е. на основе публичного ключа pub_1.

участника *i*, т.е. не будучи узлом проявляющим активное наблюдение, задача считается невыполнимой, если алгоритм шифрования *E* и ключи *k, r* являются надёжными.

Определение 2. Задача анонимизации считается теоретической[3], если наличие любого пассивного наблюдателя $\forall o \in O$ в системе $\Sigma(S, R, O)$, включая глобального, никоим образом не влияет на качество скрытия истинной связи между $(s \in S, r \in R)$.

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

ВВОД: очередь Q, период t, функции E, D, ключи шифрования k, r
ВЫВОД: подмножество шифртекстов $X \in C$
thread-1. *(* Генерация истинных шифртекстов *)*
while (true) {
    $k_j \leftarrow$ *INPUT_STREAM (* Ввод ключа получателя *)*
    $m \leftarrow$ *INPUT_STREAM (* Ввод открытого текста *)*
    $Q \leftarrow (c = E_{kj}(m))$
}
thread-2. *(* Генерация ложных шифртекстов *)*
while (true) {
    $v \leftarrow$ *RANDOM_STREAM (* Получение ложного текста от КСГПСЧ *)*
    $c = E_r(v)$
    while $(Q \neq \emptyset)$ *{} (* Ожидание пустой очереди *)*
    $Q \leftarrow c$
}
thread-3*. (* Отправление шифртекстов в сеть *)*
while (true) {
    sleep(t) (Ожидание периода *)*
    $c \leftarrow Q$, *где* $c \in X$
    QB-net $\leftarrow$ *c (* Запись шифртекста в сеть *)*
}
thread-4*. (* Принятие шифртекстов из сети *)*
while (true) {
    $c \leftarrow$ *QB-net, где* $c \in C \backslash X$ *(* Чтение шифртекста из сети *)*
    if valid$(m = D_k^{-1}(c))$ *{ (* 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

[3] Одним из возможных способов достижения теоретической доказуемости становится сведение задачи скрытия связи $(s \in S, r \in R)$ к задаче сокрытия источника этой связи $s \in S$. На базе данного подхода формируются DC, EI, QB сети.

значение. Период может изменяться по времени или вовсе иметь случайное значение. Такое поведение никак не отразится на качестве анонимности узлов до тех пор, пока будет существовать сам факт отложенности шифртекстов *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 $T = \{t\}$, 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 1 [T1]. 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 T1. 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: $P_{(d)} = P_{(i)} \vee P_{(s)}$.

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, является вычислительно сложной задачей (*Лемма 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 задача становится невыполнимой, т.к. связность шифртекстов будет априори отсутствовать. 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. $C^k = \varnothing \rightarrow c_i \notin C^k$. Теперь, если предположить обратное: $C = C^r \bigcup C^k$, где $C^k \neq \varnothing$, то

сохранение единой связи генерации, в лице независимого события x, становится возможным тогда и только тогда, когда шифртексты $c_i \in C^k$ будут создаваться вместе и вследствие того же алгоритма A, что и шифртексты подмножества C^r, посредством исполнения условия: $(c = E_k(m))\ if\ (m \neq null)\ else\ (c = E_r(v)) \leftarrow A(x)$. В результате этого, истинная связь $(m \neq null)$ инкапсулируется в независимой связи x, и не проявляет, тем самым, свою природу генерации.

Следствие T1:1. 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 целом. 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 T1:2. Пусть t есть период генерации порождаемых подмножеств C_i. 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 T1:3. Независимые события $x_1, x_2, ..., x_n$, привязанные к участникам $1, 2, ..., n$, необязательно должны быть представлены общим константным / статичным значением $= x$, т.к. данные события принадлежат в равной степени общему классу независимых событий X. В результате этого, допускается существование неравенства $x_i \neq x_j$, при любых i, j значениях.

Пример 1. В QB-задаче под алгоритмом A понимается Q – структура «очередь», а под независимым событием x наиболее часто понимается параметр $x=t$ – период генерации. Хоть t и является простым условием формирования независимости, оно всё же не является единственным. Так например, под независимым событием $x=n$ может пониматься формирование шифртекста $c \leftarrow A(n)$ 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 событий.

Пример 2. Отправление всех шифртекстов $C = \{c_1, c_2, ..., c_n\}$ сети за один раз также является независимым событием $x=a$: $C \leftarrow A(a)$, хоть и специфичным, т.к. 1) исключает какую бы то ни было интерактивность в лице запросов и ответов; 2) не располагает информацией о природе генерации самого множества C. Данный пример интересен с теоретической точки зрения тем, что с некоторой вероятностью все шифртексты во множестве C обязательно должны оставаться ложными. В противном случае, будут нарушаться сразу два условия: 1) неразличимости, где известным будет становиться неравенство $C^k \neq \emptyset$; 2) систематичности, где событие a будет зависимо от открытых текстов: $A(a) = A(m)$.

Пример 3. Небольшое различие алгоритма *A* при генерации истинных и ложных шифртекстов может нарушить систематичность генерации. Предположим, что задан статичный период *t*, как условие двух алгоритмов *P, Q*, и при этом сама процедура генерации шифртекстов c_i занимает продолжительное время *r*, такое что $0 < r < t$. Далее, если предположить, что ложные шифртексты генерируются сразу на моменте отправления $(c_i = E_r(v_i)) \leftarrow P(t)$, тогда шифртекст c_i отправится спустя время равное $t+r$. В свою очередь, если истинные сообщения генерируются до момента отправления, т.е. сначала помещаются в очередь $(c_i = E_k(m_i)) \leftarrow Q(t)$, а лишь потом отправляются, то время их отправления будет задано константой *t* на следующем этапе генерации. Таким образом, из-за использования разных алгоритмов генерации, *P* и *Q* соответственно, систематичность генерации шифртекстов c_i нарушается за счёт создаваемой разницы *r*, хоть при этом и используется общее, для двух алгоритмов, независимое событие *t*.

Пример 4. Различие между алгоритмами *P(t)* и *Q(t)* определяется лишь отсутствием или существованием очереди сообщений. Теоретически оба алгоритма основаны на независимом событии *t*, но практически они отличаются тем, что при *P(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 *Q(t)*, напротив, позволяет генерировать и сохранять сообщения в любой момент времени, вне зависимости от заданного периода *t*.

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* участников. Если в системе заложен механизм маршрутизации, то для расшифрования получатель должен будет использовать уже не один конкретный ключ k^{-1}, а все ему известные ключи $k_1, k_2, ..., k_n$ посредством метода их перебора.

Использование симметричной криптографии может повысить криптостойкость *QB-net* системы в условиях подготовки к постквантовой криптографии или в результате её наступления, т.к. известно, что современные симметричные алгоритмы с большой длиной ключа (256 бит и более) являются квантовоустойчивыми [5, с.131], в то время как новые разрабатываемые асимметричные алгоритмы, с консервативной точки зрения, могут быть слишком новы, чтобы считаться безопасными, в том числе даже для классических компьютеров.

Помимо всего этого, использование симметричных алгоритмов может сделать *QB-net* систему не только квантовоустойчивой, но и абсолютно криптостойкой, если в качестве симметричного алгоритма будет использоваться шифр Вернама [3, с.65]. В таком случае необходимо будет рассчитать максимально возможное количество используемой гаммы *Г* на необходимый промежуток времени *P* (число периодов генерации) при имеющемся количестве собеседников *N* (включая ложного получателя) и длине сообщения *L*.

$$Г = P \times (1+N) \times L$$

Тем не менее, применение исключительно симметричной криптографии затрудняет также прикладное использование и приносит дополнительный ряд издержек:

1. Усложняется общая система количественного хранения и распространения ключей: *2n* в асимметричной криптографии, *n(n - 1)/2* в симметричной криптографии [3, с.278],

2. Расшифрование данных будет приводить к линейному перебору всех известных симметричных ключей k_i, что при наличии асимметричных ключей требовало лишь одного ключа k^{-1},

3. Понизится модель угроз с возможностей активного посредника подменять публичные ключи до возможности пассивного посредника просматривать секретные ключи [3, с.80],

4. Имитовставки, в отличие от цифровых подписей асимметричного раздела криптографии, не позволяют однозначным образом подтверждать авторство сообщений [6, 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 [7, 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 открытых сообщений $m_1, m_2, ..., m_n$,
   2. Повысить количество отправляемых шифртекстов 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, как для внутренних, так и для внешних наблюдателей. Такое свойство позволяет разрывать связь между отправителем и получателем посредством передаваемого объекта, т.е. самой информации.

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 невыполнимой,

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 $(s \notin F)$ *{*
    return (Завершение алгоритма *)*
}
return h(m) (Обработка сообщения *)*

2.2. Активные наблюдения

Из всего вышеописанного ранее было доказано, что QB-задача невосприимчива к любым пассивным наблюдениям. Это говорит о том, что данная проблема принадлежит классу задач анонимизации с теоретической доказуемостью. В свою очередь, теоретическая доказуемость не является абсолютной, т.к. сводится лишь к решению проблем пассивных наблюдений, игнорируя и обходя активные. Как будет показано далее, QB-задача не обладает абсолютной анонимностью, и может быть уязвима к конкретным активным наблюдениям.

Предположим, что в QB-сети существует роль ретранслятора, скрывающая сетевые адреса абонентов (IP-адреса) друг от друга посредством перенаправления трафика, и при этом существует кооперация одного из абонентов с глобальным наблюдателем. В таком случае, задача связывания $IP \leftrightarrow k_i$ становится тривиальной, т.к. абоненту достаточно будет получить одно истинное сообщение $m = D_k^{-1}(c)$ от собеседника, а далее по полученному шифртексту $c = E_k(m)$ глобальный наблюдатель сможет определить первое его появление, тем самым деанонимизировав отправителя или получателя (его сетевое местоположение). Это говорит о сильной связности абонентов коммуникации, и, как следствие, об отсутствии анонимности друг к другу.

Ситуацию можно усложнить для наблюдателей при помощи добавления канального шифрования между узлами, как это сделано в Crowds [8]. 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:

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

Во всяком случае, такое свойство плохо исполняет функцию разграничения узлов между собой к маршрутизирующей информации *m*. Вследствие этого, задачей глобального наблюдателя становится вживление подконтрольных узлов в систему рядом с каждым другим узлом. При таком сценарии наблюдатель вновь сможет легко решить задачу $IP \leftrightarrow k_i$.

Определение 3. Задача анонимизации считается абсолютной[4], если наличие любого активного наблюдателя $\forall o \in O$ в системе $\Sigma(S, R, O)$ никоим образом не влияет на качество скрытия истинной связи между $(s \in S, r \in R)$.

Другим способом решения проблемы является придание информации свойства полиморфизма. 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 [9], I2P [10], Mixminion [11]. Такое свойство позволяет разграничивать связь абонентов друг к другу, тем самым, анонимизируя их:

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

Технически, в QB-сеть можно внедрить множественное шифрование, т.е. придать информации свойство полиморфизма, чтобы иметь возможность далее разграничивать абонентов коммуникации друг от друга, и, как следствие, исключить атаку связывания. Но, в таком случае:

1. Уменьшится скорость передачи информации, т.к. каждый маршрутизирующий узел должен будет сохранять полученное им ранее сообщение в свою очередь,

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

3. Композиция задач = *QB* + *Onion* обладает рядом тонкостей с более сложными активными наблюдениями [2, с.159], но всё также деанонимизирующими абонентов сети.

---

[4] На данный момент времени только DC-задача, в своём классическом определении, способна обеспечить абсолютную доказуемость. Отключение или замедление одного из участников, со стороны внешнего активного наблюдателя, приведёт всю сеть в состояние блокировки. Отправление большого количества пакетов, будет приводить лишь к формированию коллизий и к сопутствующим Dos/DDoS атакам, но не к выявлению источника сообщения. Отправление множества запросов со стороны внутреннего активного наблюдателя к одному из участников системы, с дальнейшим анализом полученных ответов, аналогично ни к чему не приводит.

Теорема 2 [T2]. При наличии в *QB-net* системе активного внутреннего наблюдателя f_i, в роли собеседника узла i, задача анонимизации всегда будет сводиться к сокрытию связи между данными абонентами коммуникации.

Доказательство T2. Предположим, что QB-задача может быть определена фактом скрытия коммуникации при наличии активного внутреннего наблюдателя f_i, в роли собеседника узла i. В таком случае, мы приходим к противоречию, потому как атакующий f_i, при отправлении или получении сообщений от собеседника i, априори будет знать информацию о том, что в конкретные промежутки времени $t_1, t_2, ..., t_n$, присутствовал обмен открытыми текстами $m_1, m_2, ..., m_n$, а потому был нарушен сам факт скрытия коммуникации собеседника i. В результате, задача деанонимизации начинает сводиться к поиску конкретной связи[5], а не к наличию факта существования этой связи.

Далее, если предположить 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 далее, 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 вопросом следующие моменты:

1. С какими узлами общался прослушиваемый участник i в множестве периодов D?
2. Являлся ли прослушиваемый участник i инициатором запросов при множестве D?
3. Может ли участник i намеренно генерировать ложные сообщения в роли истинных?

Далее, если допустить ситуацию при которой у каждого узла $i \in \{1, 2, ..., n\}$, не относящегося к наблюдателям, в друзьях будет присутствовать как минимум один активный наблюдатель $f_i \in \{f_1, f_2, ..., f_n\}$, тогда первая задача может быть решена тривиальным образом, при условии, что атакующие будут находиться в кооперации, т.е. $\forall\, i, j,\ f_i = f_j$, а участники будут использовать тип связи «запрос-ответ». В таком случае близкая к одновременной

[5] Данное утверждение не распространяется на другие задачи анонимизации с теоретической доказуемостью. Так например, даже если в DC-задаче будет существовать связь $i \leftrightarrow f_i$, то она не будет приводить к снижению модели угроз со скрытия инициатора связи до сокрытия связи между абонентами коммуникации. Связано это с тем, что в DC-задаче присутствует полиморфизм информации, который не позволяет наблюдателям, включающими в себя собеседника, однозначно определить точку отправления.

загруженность очереди $(d+x)t_i$, *где* $d > 1$, $x \in \{0, 1, 2\}$ у нескольких участников сети i будет свидетельствовать об ограничении первоначального множества наблюдения за счёт разделения узлов i, j по заполненным $Q_i \neq \varnothing$ и пустым $Q_j = \varnothing$ очередям.

Решение второй задачи может базироваться на условиях первой, когда будет существовать множество активных и кооперирующих наблюдателей $f_i \in \{f_1, f_2, ..., f_n\}$, а участники будут использовать тип связи «запрос-ответ». В таком случае, для поиска ответа наблюдателям потребуется выявлять факт начала коммуникации у всех узлов $i \in \{1, 2, ..., n\}$ посредством перехода состояния очереди из пустого в заполненное: $Q_i = \varnothing \rightarrow Q_i \neq \varnothing$. Первый узел, осуществивший такой переход, с большей вероятностью становится инициатором запроса.

Решение третьей задачи является наиболее трудоёмким с точки зрения наблюдателей, т.к. во-первых, оно связано с решением задачи неразличимости шифртекстов $P_{(i)}$, что является вычислительно трудной задачей, во-вторых, отсутствие решения третьей задачи усложняет выявление закономерностей первых двух задач, посредством скрытия точного состояния очередей у абонентов коммуникации «$Q_i = \varnothing$?». Хоть такая задача и связывает себя с задачей неразличимости, она ей не тождественна, т.к. помимо неё присутствует и задача систематичности $P_{(s)}$. Вследствие этого, чтобы доказать безопасность третьей задачи, необходимо свести её сложность к задаче деанонимизации $P_{(d)}$. Но это невозможно, т.к. ответ наблюдателю сам же и становится процедурой нарушения независимости события x в задаче систематичности. За счёт этого, наблюдатели могут быть уверены, что в момент ответа dt_i участник i не мог параллельно отвечать другому абоненту. Таким образом, третья задача не способна дать гарантий анонимности и посредством более длительного наблюдения всё также может проявлять паттерны, свойственные решению первой задачи.

Проблему можно решить расширением количества очередей $Q \rightarrow \{Q_1, Q_2, ..., Q_n\}$ на одном узле с их привязкой к абонентам коммуникации $i \in \{1, 2, ..., m\}$. При этом, количество очередей должно быть априори задано статичным значением n, чтобы невозможно было выявить количество связанных друзей m, т.е., количество друзей m всегда должно иметь сравнение: $m \leqslant n$. В результате, расширение очередей приведёт либо к повышению количества отправляемых шифртекстов за раз: $c_1, c_2, ..., c_n \leftarrow^t Q_1, Q_2, ..., Q_n$, либо к повышению периода генерации шифртекстов: $c_i \leftarrow^{nt} Q_i$. Это, в свой черёд, будет являться гарантией / доказательством того, что активный наблюдатель не сможет повлиять на очередь сообщений других абонентов, т.к. все его действия будут привязаны и ограничены одной непересекающейся очередью.

2.3. Сравнение с другими задачами

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

Простота масштабирования	-	-	-	+	+
Простота программной реализации[6]	+	-	-	+/-	+
Стадия анонимности	5^	6	1^	4 или 6	3
Сеть-представитель[7]	Hidden Lake	-	Herbivore	Tor	Crowds

Таблица 1. Сравнение задач анонимизации

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 и отсутствием абсолютной доказуемости. Более детальное и общее сравнение QB-задачи с другими задачами анонимизации как теоретическими, так и практическими представлено в *Таблице 1*.

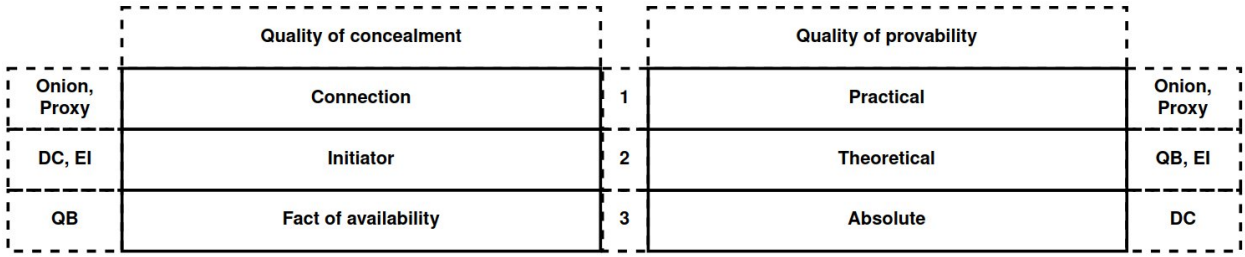

Рисунок 3. Сравнение качеств задач анонимизации

Качество анонимности нескольких задач может рассматриваться в двух плоскостях: 1) со стороны качества скрытия; 2) со стороны качества доказуемости. В зависимости от выбранного метода лидирующей задачей может стать либо QB, либо DC. Так например, QB-задача хоть и скрывает сам факт существования коммуникации, но при этом она не является

[6] Характеристика сложности может быть определена количественным соотношением суммы успешных пассивных / активных наблюдений к сумме необходимых процедур для предотвращения от таковых наблюдений. Если сумма процедур определяется небольшим числом, тогда простота программной реализации будет минимальна, что, тем не менее, не говорит об итоговой безопасности сети, т.к. малое количество процедур может свидетельствовать и об отсутствии принимаемых мер при ликвидации успешных наблюдений. В таком случае, простота программной реализации может быть не только следствием высокой безопасности, но и следствием пониженной модели угроз.

[7] Сеть-представитель необязательно должен наследовать все характеристики задачи анонимности. Так например, Herbivore, хоть и представляет собой DC-сеть, но не является ни абсолютно доказуемой, ни теоретически доказуемой. Связано это с тем, что Herbivore идёт на компромисс между анонимностью и эффективностью использования, понижая, тем самым, модель угроз. Сеть Hidden Lake напротив, придерживается концепции QB-задачи в полной мере, в том числе и в наследии всех её недостатков. Сеть Crowds и вовсе повышает модель угроз, наследуя не только классическую Proxy-задачу, но и добавляя аспект вероятностной маршрутизации.

абсолютно доказуемой. В совершенно другой плоскости находится DC-задача, которая, являясь абсолютно доказуемой задачей, всё же не скрывает факт существования связи[8].

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 [6, 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

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

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

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, H – функция хеширования, H_{MAC} – функция вычисления имитовставки на базе хеш-функции H. 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}, $E_{k'}$.

---

[8] Качество анонимности той или иной задачи также может рассматриваться со стороны простоты / лёгкости перехода между близлежайшими состояниями. Технически, DC-задача может обладать скрытием факта связи (хоть и не идеальным), если вероятность возникновения коллизий будет пренебрежимо мала, а само шифрование станет сквозным. В это же время QB-задача не может достичь уровня абсолютной доказуемости, т.к. не способна выдержать ряда активных наблюдений. Таким образом, качество анонимности DC-задачи, в сумме, способно превышать качество QB-задачи.

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 $H_{MAC(s)}(pub_i, pub_j, f(m_l)) = h$.

Плюс к этому, данная схема является самодостаточной [2, с.121] 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.

Этап 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 [12].

Предполагается также, что у получателя шифртекста существует список публичных ключей и их хешей в формате словаря, т.е. $H(pub_i) \leftrightarrow pub_i$. При успешном расшифровании пользователь получает переданное значение $H(pubA)$, после чего пытается соотнести его с имеющимся публичным ключом в словаре. Если публичного ключа $pubA$ в словаре не находится, тогда сообщение игнорируется. Иначе берётся публичный ключ отправителя $pubA$, собственный публичный ключ $pubB$, криптографическая соль s, сообщение m' и ими проверяется корректность полученной хеш-суммы $h = H_{MAC(s)}(pubA \,||\, pubB \,||\, m')$. Если сумма неверна, тогда сообщение игнорируется. Иначе по публичному ключу $pubA$ и полученному значению $S_{privA}(h)$ начинается проверка корректности цифровой подписи. Если подпись валидна, тогда сообщение m' декодируется $f^{-1}(m') \rightarrow m$ и принимается. В противном случае, сообщение игнорируется.

Одновременно с особенностью скрытия информации в шифрованной оболочке, возникает и проблема проверки корректности содержимого шифртекста, т.к. хеш-значение и подпись внутренне инкапсулированы. Для того, чтобы успешно провалидировать шифрованное сообщение, необходимо произвести дополнительно две операции расшифрования – приватным и сеансовым ключом, что является ресурсозатратным действием, преимущественно из-за использования асимметричного алгоритма.

3.2. Второй этап шифрования

$$E''_k(m) = E_k(p(h) \,||\, h \,||\, m),$$

$$h = H_{MAC(k)}(m),$$

где k - ключ сети, *p* - функция доказательства работы, *h* - результат хеширования, *m* - открытое сообщение. Функция придерживается MtE подхода (MAC-then-Encrypt), где сначала вычисляется MAC (Message Authentication Code), а далее сообщение *m*, полученный код *h* и доказательство *p(h)* шифруются функцией E_k.

Данный этап шифрования выполняет одну задач – разграничивать 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,

Функция доказательства работы *p* определяется алгоритмом proof-of-work (PoW) [13], 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, пример *00000000(n)...11001010*. Число *n* именуется сложностью работы.

Стоит также заметить, что ключ шифрования *k* может быть открытым параметром, если отсутствует необходимость в формировании конкретной группы узлов с общим секретом. В таком случае, сетевой ключ *k* становится 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» [6, 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» [14, с.93]. Данный принцип противопоставляется принципу Хортона, когда речь заходит о выборе одного из двух

подходов: MtE или EtM. Тем не менее выбор MtE обусловлен ещё и тем, что принцип криптографической обреченности нарушается также на моменте первого, и куда более затратного, этапа шифрования.

4. Сетевое взаимодействие

Наличие QB-задачи предполагает, что каждое отправляемое сообщение в сети будет достигать всех её участников. При этом в такой задаче ничего не говорится о том, как сообщение будет достигать узлов при отсутствии связи «все-ко-всем», какой протокол передачи данных будет использоваться, как должны работать прикладные приложения в такой модели, как децентрализованная структура будет обходить NAT в сети Интернет и т.д. Все эти вопросы требуют дополнительного пояснения исходя из конкретной реализации сетевого взаимодействия.

4.1. Микросервисная архитектура

Анонимная сеть HIdden Lake, как приложение[9], представляет собой набор сервисов, каждый из которых выполняет свою конкретную задачу [15]. Выбор микросервисной архитектуры, в противовес монолитной, был сделан с учётом следующих аспектов:

1. Микросервисная архитектура позволяет упростить и децентрализовать разработку прикладных сервисов, благодаря чему становится возможным применение различных языков программирования и технологий при реализации собственных приложений,

2. Микросервисная архитектура позволяет разделять ответственность сервисов к обрабатываемой или хранимой информации, благодаря чему при усложнении системы, корреляция самого усложнения будет минимально распространяться на сервисы,

3. Микросервисная архитектура позволяет добавлять новые функции, редактировать их или удалять вовсе без необходимости перекомпиляции и перезагрузки всех сервисов, что может положительно сказаться как на тестировании, так и на отказоустойчивости.

Ядром сети Hidden Lake является сервис HLS (service), который непосредственно исполняет QB-задачу и все функции шифрования / расшифрования соответственно. Помимо приложения HLS, в сети Hidden Lake также существует ряд прикладных сервисов и адаптеров. В результате этого, сеть Hidden Lake можно представить как композицию[10] нескольких сервисов.

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

[9] Репозиторий анонимной сети Hidden Lake: https://github.com/number571/hidden-lake.

[10] Символ “+” (сложение) определяет композицию функций в пределах одного приложения, в то время как символ “×” (умножение) устанавливает композицию функций посредством использования нескольких приложений. Символ “Π” объединяет приложения со схожей логикой функционирования в одну общую композиционную группу. При $m = 0 \rightarrow u(m) = 0$, $HLA{=}http$ становится равен единице, что приводит к следующему равенству: $Hidden\text{-}Lake = \Pi^n_{i=1}APP_i \times HLS \times 1 = \Pi^n_{i=1}APP_i \times HLS$.

$$Hidden\text{-}Lake = \Pi^{n}_{i=1} APP_i \times HLS \times (HLA{=}http)^{u(m)} \times \Pi^{m}_{j=1} HLA_j,$$

где APP - множество прикладных сервисов, *HLA* - множество сетевых адаптеров, *D* – доставщик открытых сообщений к прикладным сервисам, *HLA=http* – сетевой адаптер на базе протокола HTTP, $u(m) = \lceil (m-1)/(m+2) \rceil$, что эквивалентно выражению: *(1 if m > 1 else 0)*, при $m \in \{0\} \bigcup \mathbb{N}$. Адаптер HLA=http является связывающим, т.к. позволяет соединять HLS приложение с любым существующим адаптером. Отделённый HLA=http адаптер от HLS, в роли сервиса, необходим для объединения множества адаптеров под один общий интерфейс.

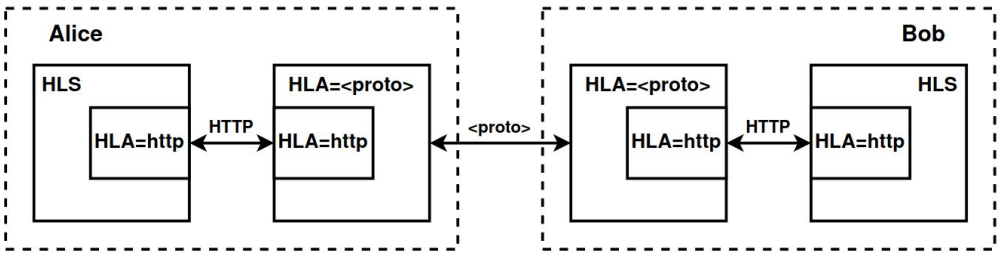

Рисунок 4. Взаимодействие двух узлов HLS посредством HLA=http и HLA=<proto>

При отсутствии прикладных приложений, а также сетевых адаптеров (за исключением HLA=http), 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, т.к. лишается своего ядра, даже при существовании прикладных сервисов и сетевых адаптеров.

Алгоритм 3. Обработка запросов в HLS приложении на языке псевдокода

```
ВВОД: запрос req, отправитель s, отображение сервисов M, функция-фильтр H
ВЫВОД: ответ rsp ИЛИ завершение алгоритма
host, method, path, head, body ← req (* Получение запроса *)
if (host ∉ M) { (* Сервис не найден в отображении *)
    return (* Завершение алгоритма *)
}
intReq ← M(host), method, path, (head || s), body (* Обогащение запроса *)
intRsp ← do(intReq) (* Запрос к сервису *)
if (noResponse(intRsp)) { (* Ответ не получен ИЛИ Ответ не требуется *)
    return (* Завершение алгоритма *)
}
status, head, body ← intRsp (* Получение ответа *)
rsp ← status, H(head), body (* Фильтрация ответа *)
return rsp (* Отправление ответа *)
```

Вследствие всего вышеописанного реализация сети Hidden Lake может представлять три различных режима коммуникации: классический, адаптационный и мультипликативный.

1. Под классическим режимом коммуникации понимаются соединения на базе стандартного HTTP адаптера, имплементированного непосредственно в HLS приложение. Данный режим предполагает, что узлы или ретрансляторы (на базе HLA=http) заранее выставляют список возможных соединений, чтобы далее иметь возможность не только принимать, но и успешно отправлять сообщения, тем самым формируя дуплексную связь. Рекомендуется применять в ситуациях, когда использование HTTP-протокола является достаточным условием, а сетевые адреса участников являются доступными, статичными и заранее известными (внесёнными в список соединений),

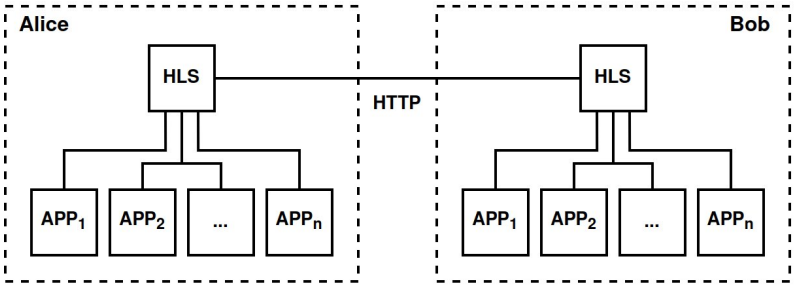

Рисунок 5. Классический режим:
Hidden-Lake = $\Pi^{n}_{i=1} APP_i \times HLS$

2. Под адаптационным режимом коммуникации понимаются соединения на базе протокола (или платформы связи) отличного от стандартного HTTP адаптера. Такой режим коммуникации позволяет отделять реализацию от конкретной используемой технологии (HTTP протокола), и далее приспосабливать её к множеству других протоколов (TCP, UDP, QUIC и т.д.) или платформ связи (социальные сети, мессенджеры, форумы и т.д.). Рекомендуется применять в ситуациях, когда у коммуникационной системы присутствует ряд ограничений, как например, невозможность установки статичных сетевых адресов, список участников сети заранее не может быть известен, применение базового HTTP протокола невозможно в выбранной коммуникационной среде и т.д.,

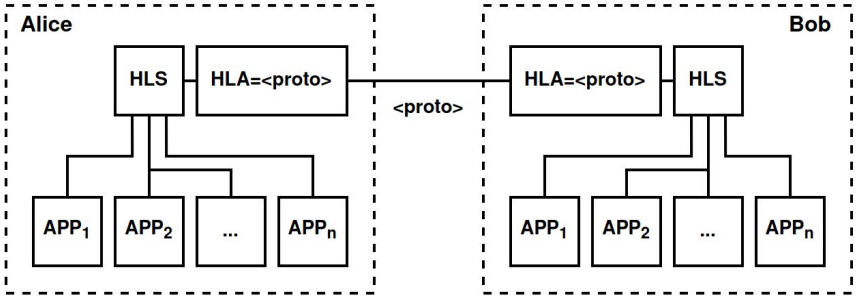

Рисунок 6. Адаптационный режим:
Hidden-Lake = $\Pi^{n}_{i=1} APP_i \times HLS \times HLA=<proto>$

3. Под мультипликативным режимом коммуникации понимаются соединения на базе нескольких протоколов и/или платформ связи объединённых между собой единым HTTP адаптером. Стандартный HTTP адаптер, находящийся в HLS приложении, не способен выполнять ретранслирующую функцию, т.е. не способен передавать сообщения принятые из адаптера *HLA=<$proto_x$>* на множество других адаптеров *HLA=<$proto_i$>, для i=1, 2, ..., x-1, x+1, ..., m*. По этой причине становится необходимым существование дополняющего

HTTP адаптера, который бы придерживался как базового протокола, так и ретранслирующей функции. Рекомендуется применять в ситуациях, когда существует несколько платформ связи, большинство из которых являются нестабильными (слабое оборудование, некорректное функционирование, частые обновления API) или враждебно настроенными к автоматизации (необходимость авторизации, блокирование аккаунтов, установка таймаутов между операциями, проверка выполнения задачи – решение каптч, выполнение JS-скриптов).

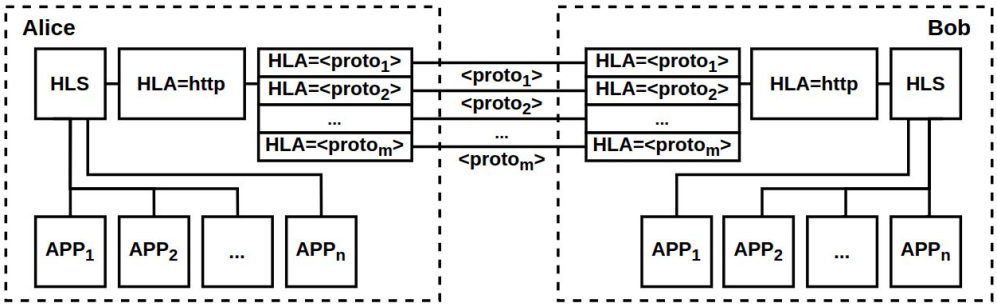

Рисунок 7. Мультипликативный режим:
$Hidden\text{-}Lake = \Pi^{n}_{i=1} APP_i \times HLS \times HLA{=}http \times \Pi^{m}_{j=1} HLA{=}\langle proto_j \rangle$

4.2. Стек протоколов «GP/12»

Абстрактный характер анонимной сети Hidden Lake формируется стеком протоколов GP/12 (сокращение от go-peer[11] и 1,2 – слои шифрования[12]), аналогичным по своей сути стеку протоколов TCP/IP. В нём также присутствуют четыре уровня: канальный (CL), сетевой (NL), транспортный (TL) и прикладной (AL), но при этом имеется ряд следующих отличий:

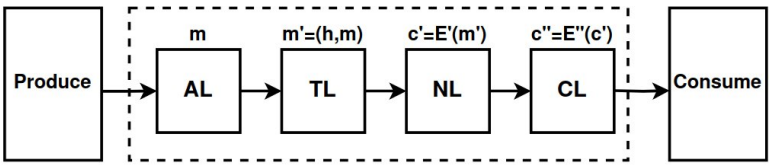

Рисунок 8. Стек протоколов GP/12

1. GP/12 идентифицирует узлы по публичным ключам, а не по IP адресам,
2. GP/12 может использовать задачу теоретически доказуемой анонимности,
3. GP/12 не зависит от сетевых протоколов и систем коммуникаций,
4. GP/12 использует схему сквозного (end-to-end) шифрования.

[11] Репозиторий проекта go-peer: https://github.com/number571/go-peer. В go-peer содержатся все основные модули: функции шифрования, работа с очередью сообщений, сетевое взаимодействие. Hidden Lake, в свою очередь, активно применяет данные компоненты для реализации таких сервисов, как HLS, HLA=tcp. Для сети Hidden Lake проект go-peer ранее являлся фреймворком по причине их общей принадлежности к кодовой базе и релизным версиям. Сейчас данные проекты разделены.

[12] Слои шифрования идут в обратном порядке относительно этапов шифрования, т.е. первый слой шифрования = второй этап шифрования, второй слой шифрования = первый этап шифрования. Различность порядков связана с различностью схем на которых базируются данные определения. Слои шифрования исходят из того – как сообщение будет получено, этапы шифрования исходят из того – как сообщение будет отправлено. Чтобы не было путаницы, в данной работе используется преимущественно определение «этапы шифрования», а исключением является лишь модель «GP/12».

Канальный уровень в сетевой модели GP/12 представляет собой распространение сообщений методом заливочной маршрутизации. Данный уровень характеризуется применением второго этапа шифрования, когда важен сам факт успешной передачи сообщения всем своим соединениям. Сетевой уровень представляет собой распространение сообщений по конкретным узлам сети, используя для этого публичные ключи в роли идентификаторов. Данный уровень характеризуется применением первого этапа шифрования, когда важна конфиденциальность и аутентичность сообщений. Транспортный уровень представляет собой маршрутизацию открытых сообщений (успешно расшифрованных) по конкретно заданным прикладным сервисам. Прикладной уровень представляет собой принятие открытого сообщения с дальнейшей его обработкой. Таким образом, стек протоколов GP/12 может быть изображён как композиция четырёх уровней с позиции принятия сообщений: CL × NL × TL × AL.

Вследствие вышеописанного, исполнение полного стека протоколов GP/12 от канального до прикладного уровней возможно лишь с использованием минимум двух сервисов в сети Hidden Lake, т.к. HLS, являясь ядром сети, покрывает лишь первые три уровня: CL, NL, TL, в то время как последний уровень AL может быть покрыт лишь использованием прикладных приложений по типу: HLM (messenger), HLF (filesharer), HLR (remoter), HLP (pinger) и т.д.

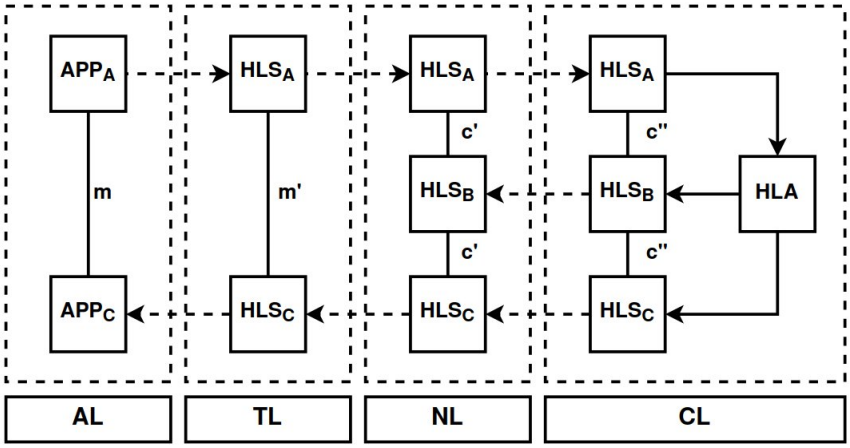

Рисунок 9. Модель GP/12 на примере сети Hidden Lake, где A – отправитель, C – получатель

Также, сетевая модель GP/12 необязательно должна обладать свойством анонимности из-за чего сервис HLS, при удалённой QB-задаче с целью сохранения GP/12 совместимости, может быть заменён композицией из трёх сервисов: D × Encryptor × HLA=<proto>, предоставляющих третий, второй и первый уровни соответственно. На базе данного подхода было создано приложение secpy-chat[13].

Анонимизация трафика формируется на втором уровне стека GP/12, когда функция генерации сообщений начинает быть неразрывно связана с какой-либо анонимизирующей задачей. Это может быть как QB, так и DC, EI –задачи. Основное ограничение в таком выборе заключается в необходимости существования теоретической доказуемости, без которой невозможно будет далее формировать абстрактную систему со свойством анонимности. Если будут применяться Proxy или Onion –задачи, то в таком случае сеть на базе сетевой

[13] Репозиторий приложения secpy-chat: https://github.com/number571/secpy-chat.

модели GP/12 не станет более анонимной, потому как этим задачам, для своего корректного исполнения, требуется немонолитная система, что противоречит определению абстрактных анонимных сетей, для которых наличие централизации никак не сказывается на качество итоговой анонимизации.

Для своей реализации стеку протоколов GP/12 достаточно лишь существования сетевых интерфейсов Produce и Consume, вследствие чего он может адаптироваться далее к конкретной среде коммуникации. Так например, модель GP/12 в сети Hidden Lake по умолчанию базируется на протоколе прикладного уровня HTTP, а при помощи сервисов HLA (адаптеров) может базироваться как поверх протоколов транспортного уровня: TCP, UDP, QUIC, прикладного уровня: SSH, FTP, SMTP и т.п, так и вовсе вне модели TCP/IP, как пример, использовав радиовещание или систему видимого света.

5. Программная реализация

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 Hidden Lake, но и подробному изложению её структурных и конфигурационных параметров.

5.1. Структурные параметры

1. Симметричная функция шифрования E_k определяется блочным алгоритмом AES с длиной ключа 256 бит в режиме шифрования CFB, где $c_0 = IV$, $c_i = E_k(c_{i-1}) \; xor \; m_i$.

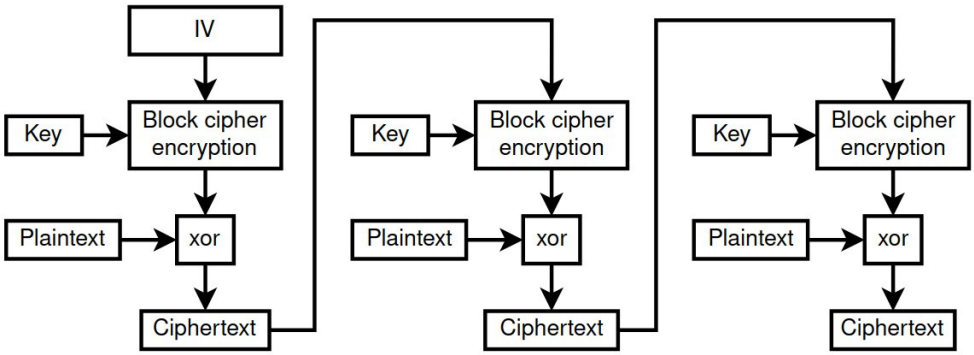

Рисунок 10. Режим шифрования CFB (режим обратной связи по шифртексту)

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 [16],

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 [6, с.93],

3. Режим шифрования CFB не требует от алгоритмов шифрования соблюдения точных параметров, как например, 128-битные блоки при режиме GCM (режим счётчика с аутентификацией Галуа) [17, с.190], а также не требует от алгоритмов функции расшифрования, что позволяет не только упростить заменяемость небезопасных шифров, но и использовать в качестве процедуры шифрования односторонние функции [18, с.282],

4. Не использовался режим шифрования GCM также и по причине излишних операций аутентификации и хранения токенов. Первый и второй этапы 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 ML-KEM-768 (до стандартизации – Kyber-768) [19]. Асимметричная функция подписания S_{priv} определяется алгоритмом ML-DSA-65 (до стандартизации – Dilithium-M3) [20]. Выбор таковых алгоритмов был сделан с учётом подготовки сети Hidden Lake к постквантовой криптографии. Установление F2F-соединения к абоненту коммуникации определяется фактически двумя публичными ключами – ML-KEM-768 для шифрования сообщений E_{pub} (ключом получателя) и ML-DSA-65 для проверки подписи сообщений V_{pub} (ключом отправителя),

3. Предполагается, что ключ сети на втором этапе шифрования редко меняется, обладает высокой энтропией и не является паролем. Но так как ключ сети не имеет фиксированного размера, он проходит сквозь функцию формирования ключа PBKDF2 для фиксации размера в 32 байт, чего требует алгоритм шифрования AES-256. Криптографическая соль и количество итераций в PBKDF2 не задаются, т.е. они по умолчанию равны пустой строке и числу 0. Хеш-функцией является SHA-512,

4. В качестве алгоритма имитовставки (MAC) используется HMAC-SHA-384. Безопасность HMAC зависит от используемой им хеш-функции [17, с.168]. Безопасность SHA-384 может быть определена стойкостью в 384 бит при задаче поиска первого и второго прообразов, и в 192 бит к поиску коллизий, исходя из атак парадокса дней рождения [6, с.52],

5. Количественные и неизменяемые параметры алгоритмов AES-256, SHA-384, ML-KEM-768, ML-DSA-64 были выбраны с консервативной точки зрения для сохранения достаточного уровня безопасности в реалиях постквантовой криптографии [5, с.131]. Алгоритм Гровера теоретически позволяет уменьшить безопасность симметричных шифров к атакам перебора и хеш-функций к поиску коллизий, сократив необходимое количество вычислений до $2^{n/2}$ и $2^{m/3}$, где n – длина ключа симметричного шифра, m – размер результирующего блока хеш-функции. Таким образом, при использовании AES-256 и SHA-384, минимальная безопасность будет определяться 128 битами, что является устойчивым

значением к атакам перебора. При отсутствии квантовых компьютеров, минимальная безопасность будет определяться алгоритмами ML-KEM-768 и ML-DSA-64, безопасность которых сравнима с 192 битным значением,

6. Ни первый этап шифрования, ни второй не защищают от атаки воспроизведения повторных сообщений [6, с.279], где спустя определённый период времени *t* злоумышленником может быть транслировано вновь, сохранённое ранее им же, шифрованное сообщение. Такое сообщение будет полностью верным, т.к. оно было сгенерировано одним из участников сети. Для защиты от подобного вида атак, сеть Hidden Lake поступает наиболее простым и радикальным способом, сохраняя хеши сообщений в свою локальную БД. Данный подход позволяет полностью исключить возможность успешного принятия дубликатов, в том числе и при перезагрузке узла, а также упростить в общем процедуру дедупликации, посредством исключения дополнительных проверок сообщений в пределах временных окон.

К сожалению, защита узла от принятия и последующей обработки дублирующих сообщений никак не защищает от воспроизведения злоумышленником сообщений, которые ещё не были приняты узлом непосредственно владеющим БД. Такое событие становится возможным, если узел отсутствовал в периоды генерации шифртекстов оставшимися участниками системы. Вследствие этого, злоумышленник может нагрузить новоприбывший узел старыми и давно забытыми сообщениями сети. Защититься от этой атаки возможно, если подключиться к давно работающему анонимизирующему узлу или ретранслятору, сохранившему историю принятых сообщений. В таком случае он начнёт выступать в роли межсетевого экрана, разграничивая старые и новые шифртексты. Ещё одним возможным решением может стать обновление ключа сети *k*, если в системе было сгенерировано много трафика, а давно функционирующие узлы неизвестны или недоступны.

Добавление же метки времени *t* на каждое сообщение второго этапа шифрования напротив, способно нарушить анонимность QB-задачи, т.к. при неправильной реализации может создать дополнительную связь в генерации шифртекстов и сделать задачу систематичности $P_{(s)}$ легко решаемой. Так например, при создании истинного сообщения метка времени будет фиксироваться не на моменте отправления, а на моменте генерации. По причине того, что сообщения из очереди отправляются последовательно в сеть, то сгенерированный заранее ложный шифртекст будет обладать меткой времени приближенной к метке открытого сообщения. Иными словами, будет прослеживаться связь генерации истинных сообщений посредством существования двух меток времени, разница между которыми будет меньше одного периода. Ситуацию можно изменить, заменив ложный шифртекст истинным, а не просто отодвинув ложный шифртекст в очереди. Но, помимо данной проблемы, в механизме генерации шифртекстов должен быть также учтён момент указания метки времени *t* в зависимости от периода генерации *T*. Иными словами, для *n-ого* сообщения в очереди метка времени должна иметь следующий вид: $t = now + nT$, *где now* – текущее время. Данное решение может обладать также и негативным эффектом, если скорость генерации шифртекстов, например при большой загруженности узла, не будет поспевать за меткой времени. В таком случае может возникнуть рассинхронизация очередей по метке времени между истинными и ложными сообщениями. Ситуацию можно изменить постоянным удалением старых сгенерированных шифртекстов из очереди, но в таком случае появится риск потери истинных сообщений, которые даже не отправлялись в сеть,

7. 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,

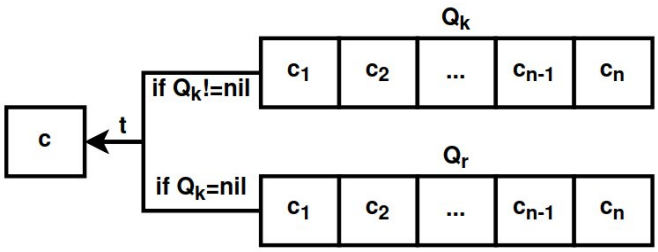

Рисунок 11. Схема двойной очереди сообщений в проекте go-peer

8. Анонимная сеть Hidden Lake может быть уязвима к активным кооперирующим наблюдателям, если таковые будут присутствовать в списке друзей у нескольких абонентов системы с отсутствующими мульти-очередями. В таком случае, помимо ухудшения качества анонимности: с задачи скрытия факта коммуникации до сокрытия связи между абонентами коммуникации, также появится и риск решения / деанонимизации самой QB-задачи. Проблему можно решить созданием нескольких очередей $Q_{k1}, Q_{k2}, ..., Q_{kn}$ для каждого отдельного абонента (с использованием настройки qbp_consumers=n), но в таком случае увеличится общее время ответа в n раз. Другим способом решения данной проблемы может стать изменение модели угроз, в которой друг априори будет равен доверенному узлу, а потому не будет заинтересован и задействован в процедурах деанонимизации. На таком конструкте F2F-систем преимущественно базируется сеть Hidden Lake,

9. Модель угроз анонимной сети Hidden Lake ограничивается исключительно защитой сетевых коммуникаций между её участниками, что, в свою очередь, предполагает отсутствие каких-либо дополнительных 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 файлам 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.

5.2. Конфигурационные параметры

1. Подход сохранения хешей всех ранее принятых сообщений также имеет минус в лице стремления объёма БД к своему постоянному увеличению. Как только появляется БД хешей, её ни в коем случае нельзя удалять или очищать (без изменения сетевого ключа), иначе все ранее принятые сообщения вновь смогут повториться, если злоумышленник будет

применять атаку воспроизведения повторных сообщений. В такой парадигме вопрос начинает базироваться на объёме и частоте 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-384, т.е. 48 байтами. Таким образом, одно принятое, либо отправленное уникальное сообщение будет увеличивать БД на 48 байт. 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 480 байт, либо 96 байт в секунду. Далее, если расширить полученный результат до года, то БД будет увеличена на ~2.6ГиБ[14] информации, что является вполне допустимым значением,

2. С каждым сохранённым хешем в базе данных уменьшается общая область допустимых шифртекстов из-за сопутствующих коллизий, свойственных всем хеш-функциям. Вследствие этого, чем больше будет приниматься шифртекстов, тем меньше их можно будет принимать в будущем. Вероятность нахождения коллизии в более чем 50% случаев начинается после преодоления порога в 2^{192} принятых шифртекстов для хеш-функции SHA-384 с использованием классических компьютеров. После этого сеть начнёт работать нестабильно, отбрасывая половину всех принимаемых системой сообщений, а единственным способом «перезапуска» сети станет удаление БД с последующей сменой сетевого ключа. Но такое событие является неосуществимым, т.к. требует сохранения 24923024920967172616946382380256460 8 ЙиБ[15] информации для 2^{192} хешей типа SHA-384. При существовании в сети 8000-ти узлов с паттерном генерации шифртекстов 1.6KiB/c, т.е. при достижении лимита пропускного канала связи в 100мбит/с, за год будет сохраняться ~2076ГиБ в локальной БД. Такими темпами, чтобы преодолеть порог нахождения коллизий >50% случаев, потребуется более чем 10^{47} лет[16].

3. Сообщения на первом этапе шифрования имеют статичный размер равный 8192 байт, из которых 4569 байт уделяется заголовочным данным: вектор инициализации (16B), хеш данных (48B), подпись ML-DSA-65 (3309B), хеш публичного ключа отправителя (48B), соль (32B), инкапсулированный ключ шифрования ML-KEM-768 (1088B), а также размеры данных в байтах: хеша (4B), подписи (4B), публичного ключа (4B), соли (4B), шифрованного блока данных (4B), полезной нагрузки (4B), заполняющих байт (4B). Второй этап шифрования добавляет 76 байт, из которых: вектор инициализации (16B), доказательство работы (8B), хеш шифртекста (48B), сетевая маска (4B). Размер сообщения после полного шифрования становится равен 8268 байт из которых 3623 байт являются полезной нагрузкой. Таким образом, если период генерации сообщений равен 5 секундам, тогда за одну секунду узел может передать ~724.6 значимых байт. Если прикладное приложение предполагает коммуникацию типа «запрос-ответ», тогда вследствие произведённого запроса и существующей последовательности (очереди) неминуемо начнётся этап ожидания ответа, что приведёт к двойному уменьшению пропускной способности до ~362.3 значимых байт в

---

[14] ((48 [байт] × 10 [узлов] / 5 [секунд]) × 60 [секунд] × 60 [минут] × 24 [часов] × 7 [дней] × 4 [недель] + 2 [дня]) × 12 [месяцев] / 2^{30} [ГиБ] = 2.595520041882992 ≈ 2.6ГиБ.

[15] (2^{192}[хешей] × 48[байт] / 2^{80}[ЙиБ]) = 24923024920967172616946382380256460 8 ЙиБ.

[16] (2^{192}[хешей] × 48[байт]) / ((48 [байт] × 8000 [узлов] / 5 [секунд]) × 60 [секунд] × 60 [минут] × 24 [часов] × 7 [дней] × 4 [недель] + 2 [дня]) × 12 [месяцев] = 135140700251269152632028 72779249622921588462718 9 > 10^{47} лет.

секунду из-за увеличенного интервала ожидания равного ~10 секундам. Итого, если будет присутствовать задача передачи файла размером 1МиБ по сети, то транспортировка будет занимать приблизительно от ~24.1 до ~48.2 минут (~1447 и ~2894 секунд соответственно),

4. Сложность работы сети Hidden Lake определяется параметром work_size_bits, который в стандартной (рекомендуемой) конфигурации равен 22 битам. Данный параметр ограничивает пропускную способность каждого узла сети, противодействуя тем самым DoS и DDoS атакам, а также атакам нацеленным на дублирование сообщений по нескольким подсетям. Данный параметр способен определять примерную количественную границу участников сети.

Предположим, что пропускная способность каждого узла в сети равна 100мбит/с, размер генерируемого сообщения равен 8КиБ, а период генерации равен 5 секундам. В таком случае, перегрузка сети будет возможна при наличии более чем 8000 узлов[17]. Это, в свою очередь, является верхней границей не учитывающей возможности злонамеренных действий со стороны самих узлов, способных генерировать трафик более чем за 8КиБ/5с = 1.6КиБ/с. Без учёта параметра work_size_bits (второго этапа шифрования), естественным ограничением для атакующих становится время шифрования сообщений (первый этап шифрования). На процессоре «Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz» шифрование одного 8КиБ сообщения занимает приблизительно 0.254мс, что в среднем требует 0.4 секунд[18] для достижения предельной величины в 12.5МиБ, с дальнейшей возможностью переполнения сетевого канала в 252мбит/с = 12.5/0.4. В результате, 8000 честно работающих узлов становятся равны менее чем одному зловредному ≈ 0.4.

Для предотвращения такой разницы вводится параметр, определяющий сложность работы. При work_size_bits=22 шифрование одного 8КиБ сообщения занимает приблизительно 1.98с на том же процессоре без учёта времени первого этапа шифрования, что в среднем требует 52.8 минут[19] (3168 секунд) для достижения предельной величины в 12.5МиБ. В результате, для достижения предела злоумышленнику потребуется ~3168 параллельных вычислений такой же процессорной мощности, чтобы иметь возможность преодолевать пропускную способность канала в 100мбит/с. Беспрерывное шифрование сообщений при work_size_bits=22 расходует ~15% процессорной мощности, что позволяет увеличить параллельность в ~6.(6) раз для достижения лимита в 100%. Таким образом, ~3168 параллельных вычислений сравнимы с использованием 3168 / 6.(6) ≈ 475.2 физических процессоров, работающих на пределах своих мощностей.

Ситуация для злоумышленника упрощается, если вместо CPU будет использоваться GPU. Хешрейт SHA-384 для «Nvidia GTX 1080 Founders Edition» составляет ~1050 MH/s[20], в то время как хешрейт «Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz» равен ~26.6 MH/s[21]. В результате, разница между процессорами определяется 1050 / 26.6 ≈ 39.5 единицами, что равносильно соотношению 475.2 CPU к 475.2 / 39.5 ≈ 12 GPU при решении PoW задачи.

---

[17] (100[мбит/с] / 8[бит] = 12.5[МиБ/с]) / ((8[КиБ]/5[с] = 1.6[КиБ/с]) / 1024[байт] = 0.0015625[МиБ/с]) ≈ 8000 узлов, генерирующих каждый по 1.6КиБ/с.

[18] (8[КиБ] / 0.254[мс]) → (1.6[КиБ] / 0.0508[мс]), 0.0508[мс] × 8000[пакетов по 1.6КиБ] = 406.4мс ≈ 0.4с.

[19] (8[КиБ] / 1.98[с]) → (1.6[КиБ] / 0.396[с]), 0.396[с] × 8000[пакетов по 1.6КиБ] = 3168с = 52.8м.

[20] «8x Nvidia GTX 1080 Hashcat Benchmarks»: https://gist.github.com/epixoip/a83d38f412b4737e99bbef804a270c40.

[21] При использовании 15% мощностей процессора, хешрейт вычисления SHA-384 составляет ~4 MH/s. При 100% нагрузке хешрейт будет повышен в 6.(6) раз, что составит уже 26.6 MH/s.

6. Заключение

В ходе работы было разобрано функционирование анонимной сети Hidden Lake на основе её математических моделей, с дальнейшим указанием преимуществ и ограничений, следующих из QB-задачи. Были приведены основные сервисы сети, а также их сетевые взаимодействия друг с другом в парадигме микросервисной архитектуры. Был описан стек протоколов GP/12, позволяющий сетям с теоретически доказуемой анонимностью становиться абстрактными. Были представлены критерии выбора алгоритмов шифрования, подписания, хеширования, их параметров и настроек в моменты проектирования и реализации сети.

6.1. Прикладное применение

Анонимная сеть Hidden Lake, обладая всем вышеописанным рядом преимуществ и недостатков, выстраивает параллельным образом также и границы своего прикладного использования. В отличие от сетей Tor и I2P, область применения которых достаточно обширна, как за счёт отсутствия связности абонентов коммуникации, так и за счёт пониженной модели угроз (отсутствие глобального наблюдателя), сеть Hidden Lake напротив, сильно ограничена в способах использования, т.к. низкая пропускная способность канала связи (периоды генерации) и проблема масштабируемости (слепая маршрутизация) не позволяют HL становиться поистине монолитной системой. Вследствие этого, единственным способом расширения сети становится горизонтальное масштабирование за счёт создания множества обособленных друг от друга подсетей. Несмотря на все недостатки, ограничения и неудобства использования в конкретных ситуациях, сеть Hidden Lake всё же обладает потенциальным списком полезных использований:

1. Защита локальных сетей от прослушивания. Проблема масштабируемости наиболее остро проявляет себя в глобальных сетях и наименее ощутимо в локальных. За счёт этого, а также за счёт отсутствия какой-либо монолитности, сеть Hidden Lake может легко запускаться и применяться в малых коммуникационных системах. В свою очередь, вероятность присутствия глобального наблюдателя в локальных сетях несравненно выше, чем в глобальных, а потому возможность анонимной сети противодействовать глобальным наблюдениям становится одним из важных условий,

2. Защита военных сетей от прослушивания. Абстрактный характер анонимной сети Hidden Lake позволяет адаптироваться под конкретные коммуникационные условия передачи информации. В свою очередь, правильное выстраивание friend-to-friend связей (например, иерархическая структура) позволяет снизить риски компрометации информации между несколькими участниками сети. Анонимизация трафика позволяет скрыть реальную активность и действующую структуру связи её абонентов. Существование конкретных периодов генерации и знание точного количества узлов в сети позволяет избежать DoS и DDoS атаки, если отдельная часть системы однажды начнёт генерировать трафика больше, чем было задано ограничивающей константой,

3. Формирование тайных каналов связи. Адаптеры в сети Hidden Lake позволяют не только приспосабливаться к конкретным сетевым протоколам, но также и встраивать (вживлять) анонимизирующий трафик в уже существующие централизованные и/или

децентрализованные системы. За счёт увеличения накладных расходов в лице удлинения периодов генерации, уменьшения частоты пропускного канала связи, дополнительного применения криптографических или стеганографических примитивов, генерируемый трафик может скрываться не только от внешних сторонних наблюдателей, но и от самой системы.

Данный список полезных прикладных использований не является исчерпывающим, т.к. он представляет собой только те случаи в которых Hidden Lake находится в выигрышной позиции в сравнении с большинством других представителей анонимных систем. Помимо такого списка существуют также пересекающиеся области. Так например, Hidden Lake вполне корректно может использоваться для выполнения определённо заданных функций и алгоритмов в глобальной сети, таких как удалённый доступ, просмотр сайтов, отправление сообщений, обмен файлами и т.д. Ограничением в данном случае является лишь количество участников в сети, из-за существующей проблемы масштабируемости, и поточная связь, из-за периодов генерации шифртекстов.

Модель угроз сети Hidden Lake строится таким образом, что наблюдателями могут быть только внешние и внутренние злоумышленники, но не друзья – т.е. непосредственные абоненты коммуникации. Из-за этого усложняется и ограничивается ряд прикладных использований в которых абоненты коммуникации должны оставаться анонимами друг к другу. Использование ретрансляторов / маршрутизаторов, в совокупности с канальным или множественным шифрованием, позволяет усложнить анализ сетевых коммуникаций, разграничивая связь абонентов друг к другу. Но данный подход не решает проблему основательно, т.к. в кооперации с пассивным или активным глобальным наблюдателем (в зависимости от предпринимаемых мер) один из абонентов коммуникации всё же имеет возможность деанонимизировать[22] своего собеседника.

6.2. Для разработчиков

Анонимная сеть Hidden Lake может дополняться новыми функциями и приложениями тремя допустимыми способами относительно уровня разработки:

1. Высокоуровневый метод. Написание приложений и адаптеров при помощи микросервисной архитектуры. С помощью данного подхода появляется возможность писать сервисы на любом удобном языке программирования или технологии. Необходимо лишь придерживаться HLS API[23],

2. Среднеуровневый метод. Использование пакета pkg/network[24], располагаемого внутри проекта Hidden Lake. С помощью данного подхода появляется возможность писать приложения, не используя при этом микросервисную архитектуру. Недостатком является зависимость разработки от языка программирования Go,

---

[22] В данном случае под деанонимизацией понимается нахождение сетевого адреса абонента, но не его реальной активности с другими участниками сети. В концепте QB-задачи, это не может называться деанонимизацией, т.к. роль ретрансляторов или маршрутизаторов отсутствует в нём по определению. Из этого следует, что QB-задача не пытается скрывать абонентов коммуникации друг от друга, а все предпринимаемые меры по анонимизации связности абонентов начинают сводиться к другим задачам.

[23] HLS API: https://github.com/number571/hidden-lake/tree/master/cmd/hls#hls-api

[24] Пакет pkg/network: https://github.com/number571/hidden-lake/tree/master/pkg/network

3. Низкоуровневый метод. Использование библиотеки go-peer. С помощью данного подхода появляется возможность существенно изменять работу и специфику сети, в том числе, располагая возможностью исключать анонимизацию трафика. Данный метод следует выбирать с осторожностью и в тех ситуациях, когда выстроенная архитектура Hidden Lake ограничивает ряд возможных использований.

Сеть Hidden Lake разрабатывается полностью открыто. Весь её исходный код доступен на GitHub. Внести изменения в репозиторий проекта можно при помощи создания соответствующего Pull Request[25]. Задать интересующие вопросы, оставить предложения или указать на существующие ошибки, неточности, уязвимости можно в разделе Issues[26].

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. Граймс, Р. Апокалипсис криптографии / Р. Граймс. — М.: ДМК Пресс, 2020. - 290 с
6. Шнайер, Б., Фергюсон, Н. Практическая криптография / Б. Шнайер, Н. Фергюсон. - М.: Издательский дом «Вильямс, 2005. - 420 с.
7. Таненбаум, Э., Уэзеролл, Д. Компьютерные сети / Э. Таненбаум, Д. Уэзеролл. — СПб.: Питер, 2017. - 960 с.
8. 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).
9. 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).
10. 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).
11. 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).
12. Ишкуватов, С. Способ и алгоритм определения типа трафика в шифрованном канале связи [Электронный ресурс]. — Режим доступа: https://cyberleninka.ru/article/n/sposob-

[25] Pull Requests: https://github.com/number571/hidden-lake/pulls

[26] Issues: https://github.com/number571/hidden-lake/issues

i-algoritm-opredeleniya-tipa-trafika-v-shifrovannom-kanale-svyazi (дата обращения: 04.07.2024).
13. 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).
14. Хлебников, А. OpenSSL 3: ключ к тайнам криптографии / А. Хлебников. — М.: ДМК Пресс, 2023. - 300 с.
15. Lewis, J., Fowler, M. Microservices [Электронный ресурс]. — Режим доступа: https://martinfowler.com/articles/microservices.html (дата обращения: 07.09.2024).
16. Heaton, R. The Padding Oracle Attack [Электронный ресурс]. — Режим доступа: https://robertheaton.com/2013/07/29/padding-oracle-attack/ (дата обращения: 04.07.2024).
17. Омассон, Ж. О криптографии всерьез. Практическое введение в современное шифрование / Ж. Омассон. — М.: ДМК Пресс, 2021. - 328 с.
18. Мао, В. Современная криптография: теория и практика / В. Мао. - М.: Издательский дом «Вильямс, 2005. - 768 с.
19. Module-Lattice-Based Key-Encapsulation Mechanism Standard [Электронный ресурс]. — Режим: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf (дата обращения: 19.10.2024).
20. Module-Lattice-Based Digital Signature Standard [Электронный ресурс]. — Режим: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf (дата обращения: 19.10.2024).