Anonymous network "Hidden Lake"

Kovalenko Gennady Alexandrovich

Annotation. The Hidden Lake network, being a QB network by nature, also represents a number of new architectural solutions that have not previously been used in the construction of anonymous systems. Based on the principle of microservice architecture, such a network can not only add, but also remove functions as needed, without changing the general mechanism of operation. Based on blind routing and full encryption of messages, such a network connects all nodes in the system, preventing long-term monitoring of connections and the fact of communication. Understanding the general principles of the network based on its mathematical models can provide not only an assessment of the correctness of the functioning of the entire system, but also a possible vector for the development of future anonymous communications.

Keywords: hidden systems; anonymous networks; decentralized networks; theoretically provable anonymity; qb-task; microservice architecture; gp/12 protocol stack; hidden lake network; post-quantum cryptography;

Content

1.Introduction

The Hidden Lake (HL) anonymous network is a decentralized F2F (friend-to-friend) [1] anonymous network with theoretical provability [2, p.49]. Unlike well-known anonymous networks, such as Tor, I2P, Mixminion, Crowds, etc., the HL network is capable of resisting global observer attacks. The Hidden Lake network does not care about such criteria as: 1) the level of network centralization, 2) the number of nodes, 3) the location of nodes, and 4) the connection between nodes in the network to anonymize its traffic, which makes such a system abstract [2, p.144].

2. QB task

The Queue Based (QB) task [2 p.149] is the core of the Hidden Lake anonymous network, which forms theoretically provable anonymity. QB networks are one of the simplest anonymization tasks in terms of software implementation.[1], in comparison with other representatives of theoretical provability in the form of DC (Dining Cryptographers) [3, p.225] and EI (Entropy Increase) [2, p.165] networks. Formally, a QB network can be described by a system of the following type:

$$QB\text{-}net = \Sigma^{ni=1}(T = \{ti\}, K = \{ki\}, C = \{(c \in \{Ekj(m), Eri(v)\}) \leftarrow^{ti} Qi\})$$

*Where*n is the number of nodes in the system, K is the set of encryption keys, T is the set of generation periods, C is the set of encrypted messages, Q is the queue of encrypted messages, i,j are the identifiers of individual nodes, E is the encryption function, m is the plaintext message, v is the false message, r is the encryption key not in the set K.

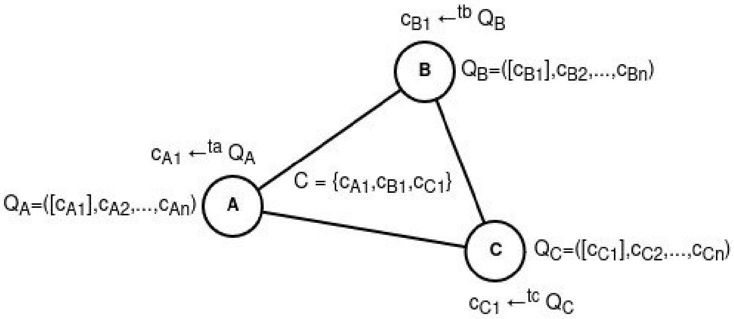
$c_{B1} \leftarrow^{tb} Q_B$
B
$Q_B=([c_{B1}],c_{B2},\dots,c_{Bn})$
$c_{A1} \leftarrow^{ta} Q_A$
$C = \{c_{A1},c_{B1},c_{C1}\}$
$Q_A=([c_{A1}],c_{A2},\dots,c_{An})$
A
C
$Q_C=([c_{C1}],c_{C2},\dots,c_{Cn})$
$c_{C1} \leftarrow^{tc} Q_C$

Figure 1.QB network with three participants A, B, C

The above system can be represented by four states:

1. *$Qi \leftarrow (c= Ekj(m))$, where $kj \in K$, $c \in C$.* The plain message m is encrypted with the recipient's key kj ∈ K. The encryption result is*c= Ekj(m)*is placed in the Qi queue,

2. *$Qi \leftarrow (c= Eri(v))$, if $Qi = \varnothing$, where ri*$\notin$ K, c ∈ C. The false message v is encrypted with the key without the recipient ri ∉ K. The encryption result is*c= Eri(v)*is placed in the Qi queue,

3. *$c \leftarrow t\ Qi$, where $t \in T$, $c \in C$.*At each time period t, an encrypted message c is taken from the queue Qi ∈*{Ekj(m), Eri(v)}*and is sent to all network participants,

[1] The MA anonymous network is written in 100 lines of Go programming language code, using only the standard library. Repository:https://github.com/number571/micro-anon.

4. $m' = Dki-1(c)$, *wherec* $\in C$.Each participant tries to decrypt the encrypted message it received with its key ki-1. If the ciphertext cannot be decrypted$m' \neq m$, then this means that the recipient is either someone else (the key kj $\in$ K was used) or no one (the key rj $\notin$ K was used).

Theorem 1. In a multitude of keys$r \in R$,for any ciphertext c =$Er(v)$,there exists an r-1 that results in the decryption of the false message v = Dr-1(c).

Proof 1. The set R is defined by the difference of two sets U \ K, when U = K$\bigcup R$, which is the set of keys of the encryption function. For any u$\in$*U is mapped into a set of ciphertexts:*$\bigcup Eu \rightarrow C$, for which there will also exist a u-1 that performs the inverse mapping to the set of plaintexts:$\bigcup Du\text{-}1 \rightarrow M$, based on the fact that the system$\sum(M, C, U, E, D)$is a cipher [4, p.75].

The anonymity of QB networks is based on the difficulty of determining the state of the encrypted message c, namely what it is:*Ek(m)*or*Er(v)*. The order of messages Q in turn guarantees that there will always be a message c that will be generated by the system in a time period equal to t, regardless of the nature of the message itself. In the absence of true messages, the queue Q can be considered as a queue of exclusively false messages.*Er(v)*. When a true message appears*Ek(m)*, this begins to replace the false*Er(v) at a certain time period t*As a result, the QB network becomes a noise generator with the function of short-term replacement of random traffic with real traffic, and the indistinguishability of encrypted messages from each other becomes a key factor of anonymity.

Theorem 2. Given two encryption keys k, r (generating true and false ciphertexts, respectively), determining the truth of a selected ciphertext ci from a finite set C = {c1, c2, ..., cn} is reduced to a computationally complex problem if the security conditions of the E, k, r parameters are met.

Proof 2. The problem of the truth of the ciphertext ci from the set C is reduced to determining its belonging to two states: Ek(mi) and Er(vi). For unknown parameters k, r, the upper bound of the search is determined by the number of iterations of the complete search equal to $|K\bigcup R| = |U| = |\{u1, u2, ..., u|U|\}|$, *for* $K\bigcap R = \varnothing$accordingly, since the unknown variable in this case remains the belonging of the values ui being sorted to the sets K and R (Theorem 1). With known u, but unknown u-1, accordingly, the problem of determining the truth of the ciphertext ci is reduced to one of the asymmetric problems [3, p. 378][3, p. 386].

In summary, the anonymity of QB networks is determined not only by the break in communication between the sender and the recipient for the global observer, but also by the absence of communication in the very fact of sending and receiving information. In other words, for passive observers, including the global observer, the task of determining the state of the subject is beyond their capabilities, namely:

1. Does participant i send a true message Ekj(m) in period ti??
2. Does participant i receive any message Dki-1(c) in periods equal to T\{ti}??
3. Is participant i → Er(v) inactive in the analyzed period ti??

In all such scenarios, without being one of the nodes directly participating in the communication and without exerting any influence on the queue Qi of the analyzed participant i, i.e. without being a node exerting active observation, the task is considered impossible if the encryption algorithm E and the keys k, r are reliable.

Algorithm 1. Functioning of participant i in the QB-net system in pseudocode language

```
INPUT: queue Qi, period ti, functions E, D, encryption keys ki, ri
OUTPUT: a subset of ciphertexts Ci∈C
thread-1.(*Generating true ciphertexts*)
for (;;) {
    $k_j$ ← INPUT_STREAM (*Input receiver key*)
    m ← INPUT_STREAM (* Plaintext Input *)
    Qi ← (c= Ekj(m))
}
thread-2.(*Generating false ciphertexts*)
for (;;) {
    if (Qi= Ø) {
        v ← RANDOM_STREAM (* Receiving fake text from the KSGPCH *)
        Qi ← (c= Eri(v))
    }
}
thread-3. (* Sending ciphertexts to the network *)
for (a = 1; ; a = a + 1) {
    sleep(ti) (* Wait for period *)
    ca ← Qi, where is ca∈ Ci
    QB-net ← ca(* Write ciphertext to the network *)
}
thread-4. (* Accepting ciphertexts from the network *)
for (b = 1; ; b = b + 1) {
    $c_b$ ← QB-net, where cb∈ C\Ci(* Reading ciphertext from the network *)
    if valid(m = Dki-1(cb)) { (* Checking the correctness of the decryption *)
        OUTPUT_STREAM ← m(* Output of the received plaintext *)
    }
}
```

There are also a number of interesting and not entirely obvious moments in QB networks. For example, the period ti of each individual participant i does not necessarily have to have a constant value. The period can change over time or even have a random value. Such behavior will not affect the quality of the nodes' anonymity as long as the fact of the ciphertexts' delay c exists in the form of the message queue Q. If a message can be sent bypassing the queue, then the anonymity will gradually deteriorate depending on the number of messages sent in this way.

Thus, unlike DC networks, where the period T is represented by only one common value T = {t}, QB networks make the period not only subjectively (individually) configurable T = {t1, t2, ..., tn}, but also not necessarily static for each generated message t ∈ [l;k], where l ≤ k. This property allows QB networks not to cooperate with individual nodes during the period, and also to better hide the pattern of user affiliation to the anonymizing traffic.

Theorem 3. The theoretically provable anonymity of the QB-net system is based on the systematic generation of the set of ciphertexts C = Cr⋃ *Ck = {c1, c2, ..., cn}*and on the indistinguishability of its subsets with respect to true Ck and false Cr ciphertexts.

Proof 3. Let Cr = {c1r, c2r, ..., cmr} be the set of false ciphertexts, and let Ck = {c1k, c2k, ..., cnk} be the set of true ciphertexts for m, n⩾*0*accordingly, then when they are combined, the set of all ciphertexts is created: Cr⋃ *Ck = {c1r, c2r, ..., cmr, c1k, c2k, ..., cnk} = {c1, c2, ..., cm+n} = C.*Let the deanonymization problem P(d) be further understood as the unambiguous finding of the fact of existence or absence of the true message (n > 0?) in the set of ciphertexts C. The deanonymization problem, in turn, is based on two other problems: indistinguishability P(i) and systematicity P(s). When solving one of the two subproblems, the deanonymization problem will also be considered feasible, which can be expressed in the disjunctive form: P(d) = P(i)∨*P(s).*

Indistinguishability problem*P(i)*can be determined by two possible situations: 1) either by the method of finding the true ciphertext*ci=Ek(mi)*∈*Ck*; 2) or, on the contrary, by means of proving the absence of true ciphertexts*ci* ∉*Ck =Ø. The problem becomes trivial in the absence of ciphertexts in general, since C=Ø → Ck= Ø. If C ≠ Ø, then the problem is reduced to the problem of the ratio of ciphertextsci*∈*C*to their original subsets:*Cr*or Ck , which, as shown earlier, is a computationally difficult problem (Theorem 2).

Systematicity problem*P(s)*can be determined by finding additional connections in the ciphertext generation mechanism*C = {c1, c2, ..., cn}*. If we take a special case*C=Ø*, then the coherence of the ciphertexts will be a priori absent, which, in turn, shows the introductory difference between the systematicity problem and the indistinguishability problem. Further, let |C| = 1, i.e. C = {c}, then the systematicity problem will be reduced to the question: "as a result of what event x was the ciphertext c obtained?". If the event x has no connections with any plaintext m (received, being received, sent or being sent), i.e. the ciphertext c was not created as a result of the appearance of m as an event, then x should be considered as an independent event. If we assume that there is some algorithm A that generates ciphertexts ci by means of an independent event x, i.e. ci ← *A(x)*, where ci∈*C, and at the same time*C = Cr, then in the generation mechanism there will be a priori no additional connections except for the main connection represented by the event x, since ci ∉*Ck= Ø*. Now, if we assume the opposite:*C = Cr*⋃ *Ck, whereCk≠ Ø,* then the preservation of a single connection of generation, in the face of an independent event x, becomes possible if and only if the ciphertexts*ci*∈*Ckwill be created together and as a result of the same algorithm A as the ciphertexts of the subsetCr*, by fulfilling the condition:*(c= Ek(m)) if (m≠null) else(c= Er(v))* ← *A(x). As a result, the true relationship(m≠null)*is encapsulated in an independent connection x, and thus does not reveal its generating nature.

Consequence 3.1.A finite set of ciphertexts C can be viewed as the result of the step-by-step generation of subsets C1 = {c1}, C2 = {c1, c2}, ..., Cn = {c1, c2, ..., cn} and as the completion of network communication in general. Thus, if the conditions of indistinguishability and systematicity are satisfied by the set C = Cn, the subsets Ci⊆Cn continue to inherit their execution equally.

Corollary 3.2. Let t be the period of generation of the generated subsets Ci. In this case, t is also an input condition and an independent event for the ciphertext generation algorithm:*ci* ← *A(t)*. Since the period t is based on algorithm A and the generation of ciphertexts is not related to plaintexts, then its variable characteristic in the form of staticity (t = T) or dynamism ($t \in [l;k]$) is not able to influence the quality of anonymity.

Corollary 3.3. Theoretically, the provable anonymity of the QB problem does not depend on any one specific ciphertext generation algorithm A, and as a consequence, the set of system participants {1, 2, ..., n} is capable of using many different algorithms {A1, A2, ..., An}, due to the sufficiency of the condition in the form of independence of events in the generation of ciphertexts.

Example 3.1. In the QB problem, algorithm A is most often understood as the periodicity of generating ciphertexts based on queues with an input condition t (period). Although this is the most practical algorithm, it is still not the only one. For example, the generation algorithm A' can also be understood as the formation of a ciphertext*ci* ← *A'(n) based on the n-th number of ciphertexts received by the system from other participants. The absence of dependence on plaintexts naturally leads to similar theoretical provability and indicates the possibility of using several different algorithms by the system.*

Example 3.2. Sending all ciphertexts C = {c1, c2, ..., cn} to the network at once is also an algorithm*C* ← *A(C)*, although specific in nature, since it excludes any interactivity in the form of requests and responses. This example is also interesting in that with some probability all ciphertexts in the set C must necessarily remain false. Otherwise, two conditions will be violated at once: indistinguishability, where the inequality Ck will become known≠ *Ø*, and systematicity, where the generation algorithm will depend on open texts:*A(C)=A(m)*.

Example 3.3.A small difference in the algorithm A in generating true and false ciphertexts can disrupt the systematicity of the generation. Suppose that a static period t is specified as a condition of the algorithm, but the generation of ciphertexts ci takes a long time x, such that 0 < x < t. Further, if we assume that false ciphertexts are generated at the moment of sending*(ci= Er(vi))* ← *A(t)*, bypassing the queue and due to the absence of any messages in the queue, then the ciphertext*ci*will be sent after a time equal to x+t. In turn, if true messages are generated before the moment of sending, i.e. first placed in the queue*(ci= Ek(mi))* ← *A(tQ)*, then the time of their sending will be equal to t at the next stage of generation. Thus, the systematicity of the algorithm is violated by different input conditions (independent events) when generating true and false ciphertexts, respectively.

Example 3.4. Difference between algorithms*A(t)*And*A(tQ)*is determined only by the absence or existence of a message queue. Theoretically, both algorithms are based on independent events, but in practice they differ in that when*A(t)*the generation of true ciphertexts will be more difficult, since it leads to their manual creation at a specific time period t due to the lack of a mechanism for postponing messages until a specified condition. Algorithm with a queue*A(tQ)*, on the contrary, allows generating and storing messages at any time, regardless of the specified period t. This is why the QB task is called Queue, and not Time Based.

Further, in QB networks it is assumed that asymmetric cryptography is used by default, and as a consequence the keys $ki \neq ki-1$ are not directly related to each other. However, QB networks are quite capable of being guided exclusively by symmetric cryptography, in which $ki = ki-1$. In this case, the keys will not represent each individual participant of the system from n possible ones, but directly the connection between its participants from $n(n - 1)/2$ possible edges of the graph (system), which also leads to the appearance of common keys of the form $ki = kj$ for some i, j participants. If the system has a routing mechanism, then for decryption the recipient will have to use not one specific key from the sender, but all the keys known to him by means of their enumeration.

The use of symmetric cryptography can increase the cryptographic resistance of the QB-net system in preparation for post-quantum cryptography or as a result of its onset, since it is known that modern symmetric algorithms with a large key length (256 bits or more) are quantum-resistant [5, p. 131], while newly developed asymmetric algorithms, from a conservative point of view, may be too new to be considered secure, including even for classical computers.

In addition to all this, the use of symmetric algorithms can make the QB-net system not only quantum-resistant, but also absolutely cryptographically resistant, if the Vernam cipher is used as a symmetric algorithm [3, p.65]. In this case, it will be necessary to calculate the maximum possible amount of gamma G used for the required time interval P (the number of generation periods) with the available number of interlocutors N (including the false recipient) and the message length L.

$$G = P\times(1+N)\times L$$

Unfortunately, the use of exclusively symmetric cryptography also complicates practical use and brings with it an additional series of costs:

1. The general system of quantitative storage and distribution of keys becomes more complex: 2n in asymmetric cryptography, $n(n - 1)/2$ in symmetric cryptography [3, p.278],

2. Decrypting the data will involve a linear search of all known symmetric keys.*ki*, which in the presence of asymmetric keys required only one key k-1,

3. The threat model will be reduced from the ability of an active intermediary to replace public keys to the ability of a passive intermediary to view secret keys [3, p.80],

4. Imitation inserts, unlike digital signatures of the asymmetric section of cryptography, do not allow for unambiguous confirmation of the authorship of messages [6, p.46].

2.1. Disadvantages of QB networks

Unfortunately, QB networks are not ideal and also have problems and shortcomings inherent to their class, some of which lead to limitations in applied use, while others lead to problems with network availability:

1. Linear network load. In QB networks, everyone sends a message to everyone with the sole purpose of making it impossible to narrow the area of real communication between system participants. The routing algorithm becomes blind (flood) routing[7, p.398], as a result of which the increase in the number of nodes has a linear O(n) effect on the increase in the load of the entire system,

2. Binding to the order. Each node in the QB network is tied in one way or another to its own order of messages Q, where every period of time equal to t an encrypted message is sent to the network. This means that it is possible to increase the throughput of a node only in three scenarios, each of which will lead to an increase in the load on the entire network.:

1. Increase the size of transmitted open messages m1, m2, ..., mn,
2. Increase the number of ciphertexts sent at a time c1, c2, ..., cn ← t Q,
3. Reduce the message generation period t,

3. Connectivity of communication subscribers. QB networks do not assume anonymity between nodes directly participating in communication. This is primarily due to the fact that QB networks do not have such a concept as information polymorphism [2, p.62], that is, the state of information in the system in which its appearance constantly changes from node to node, both for internal and external observers. This property allows breaking the connection between the sender and the recipient by means of the transmitted object, i.e. the information itself.

Due to all the above mentioned disadvantages, the scope of application of QB networks becomes more limited:

1. Due to the linear network load and dependence on the queue, QB networks do not scale well and can only work in small groups of up to N participants. The limit of the number of participants is limited by the network bandwidth itself, as well as the power of the nodes constantly encrypting and decrypting outgoing / incoming traffic. Due to this drawback, the implementation of streaming services and video / audio calls becomes either a very difficult task or completely impossible,

2. Due to the connectivity of communication subscribers, a number of application solutions are limited in which the anonymity of nodes to each other is important. As a result, the most relevant composition of QB networks with F2F networks (friend-to-friend) appears, where the establishment of a communicating connection occurs by two subscribers of the system, and not by one of them. This does not solve the problem of the lack of anonymity between the connected nodes, but provides additional protection against uncoordinated automatic linking and a more explicit connection of trusted communications, assuming that neither subscriber will try to deanonymize the other.

Algorithm 2. Filtering messages in F2F networks in pseudocode

```
INPUT: set of friends F, sender s, handler function h, message m
OUTPUT: processed message h(m) OR completion of the algorithm
if ($s \notin F$) {
    return (* End of algorithm *)
}
return h(m) (* Process message *)
```

2.2. Active observations

From all of the above, it was previously proven that the QB problem is immune to any passive observations. This suggests that this problem belongs to the class of anonymization problems with theoretical provability. In turn, theoretical provability is not absolute, since it is reduced only to solving the problems of passive observations, ignoring and bypassing active ones. As will be shown below, the QB problem does not have absolute anonymity, and can be vulnerable to specific active observations.

There is no QB networkthe property of information polymorphism, as a result of which a number of active observations become possible when one of the subscribers to a communication wishes to hide the connection he has built from his interlocutor.POlimorphism in anonymous networks is most often achieved by multiple encryption, where during transmission from one node to another, the superimposed layers of encryption are gradually removed, such as in Tor, I2P, Mixminion [8][9][10]. This property allows for the separation of subscribers' communications with each other, thereby anonymizing them.

$Ek3(Ek2(Ek1(m))) \rightarrow Ek2(Ek1(m)) \rightarrow Ek1(m) \rightarrow m$

For example, if we assume that in the QB network there will be a role in the form of repeaters, hiding the network addresses of subscribers (IP addresses) from each other by redirecting traffic, and at the same time there will be cooperation of one of the subscribers with the global observer, then the task of linking IP ↔ ki will be trivial, since it will be enough for the subscriber to receive one true message m = Dk-1(c) from the interlocutor, and then, based on the received ciphertext c = Ek(m), the global observer will be able to determine its first appearance, thereby deanonymizing the sender or recipient.

The situation can be complicated for observers by adding channel encryption, as in Crowds [11]. In this case, a global observer will not be able to explicitly link the sent and received message, because it will constantly change its appearance as it passes from one node to another:

$Ek3(\mathrm{m}) \rightarrow \mathrm{Ek2(m)} \rightarrow \mathrm{Ek1(m)} \rightarrow \mathrm{m}$

However, this is not information polymorphism, since it does not perform the function of distinguishing nodes from each other to the routing information m. As a result, the task of the global observer will be to implant controlled nodes into the system next to each other node. In this scenario, it will also be easy to solve the IP ↔ ki problem.

Technically, multiple encryption can be implemented in the QB network to differentiate subscribers from each other, but in this case:

1. The speed of information transfer will decrease, since each routing node will have to save the message it received earlier in turn.,

2. The anonymization system as a whole will become more complicated, since instead of one anonymization task = QB, a composition of tasks = QB + Onion will be used,

3. The composition of QB + Onion tasks has a number of subtleties with more complex active observations [2, p.159], but still deanonymizes network subscribers.

Theorem 4. In the presence of an active internal observer fi in the role of interlocutor for node i, the task of anonymization will always be reduced to concealing the connection between these communication subscribers.

Proof 4. If we proceed from the opposite and assume that the anonymization task can be defined by the fact of hiding communication in the presence of an active internal observer fi in the role of the interlocutor for node i, then we will come to a contradiction, since the attacker, when sending or receiving messages from the interlocutor, will a priori know the information that in specific time intervals t1, t2, ..., tn, there was an exchange of true texts m1, m2, ..., mn, and therefore the very fact of hiding the communication of interlocutor i was violated. As a result, the deanonymization task begins to be reduced to finding a specific connection, and not to the presence of the fact of the existence of this connection.

Further, if we assume a scenario of an attack on a QB network in which in the circle of friends F = {f1, f2, ..., fn} of participant i there will be an attacker fj in the role of an active observer capable of sending requests and receiving responses from i, then the attack model will be reduced to an analysis of the state of the queue Qi. Let us further assume that participant i has set a static message generation period equal to ti. In this case, fj will be able to send requests to and receive responses from i at certain time intervals $\{t'_i, 2t'_i, ..., n'_i\}$, dependent on the time period ti⇒*(kt'$_i$=kti+x), where x* ∈ *[0;ti*), send Rkt request$_{'i}$participant i for the purpose of analyzing the response time. If the response received after the Rkt request$_{'i}$, will be generated in the range dti, where d > 1, then this will mean the fact of real communication of participant i with someone in the network in the set of periods D = {(1+k)ti, (2+k)ti, ..., (d-1+k)ti}, since more than one period was required for the response. If d = 1, then participant i did not cooperate with anyone in the period (1+k)ti.

Thus, the above-described attack reduces the quality of anonymity of QB networks from concealing the fact of activity to concealing the communication link between subscribers. In other words, with such active observation, it is now possible to determine the state of the subject in the person of sending or receiving true messages, but the following points are still questionable:

1. Which nodes did the listened participant communicate with iin a set of periods D?
2. Was the listened participant i the initiator of the requests whenset D?
3. Can participant i intentionally generate false messages as true ones?

Further, if we assume a situation in which each node i∈*{1, 2, ..., n}*, who is not an observer, will have at least one active observer fi among their friends∈*{f1, f2, ..., fn}*, then the first problem can be solved trivially,provided that the attackers are in cooperation, i.e. ∀ i, j,*fi*= *fj*, and participants will use the communication type"request-response". In this case, the queue is close to simultaneous loading*(d+x)ti, where d > 1, x* ∈*{0, 1, 2}*several network participants i will indicate a limitation of the initial set of observations due to the division of nodes i, j by filled Qi≠ *Ø*and empty Qj= *Ø*queues.

The solution to the second problem can be based on the conditions of the first, when there will be a set of active and cooperating observers fi∈*{f1, f2, ..., fn}, and the participants will use the request-response communication type*. In this case, to find an answer, observers will need to detect the fact that communication has begun for all nodes i∈*{1, 2, ..., n}*by transitioning the queue state from empty to full: Qi= *Ø* → *Qi*≠ *Ø*. The first node to make such a transition is more likely to become the request initiator.

The solution to the third problem is the most labor-intensive from the point of view of observers, since, firstly, it is associated with solving the problem of indistinguishability of ciphertexts P(i), which is a computationally difficult problem, and secondly, the absence of a solution to the third problem complicates the identification of patterns in the first two problems, by hiding the exact state of the queues of subscribers of the communication “$Qi = \varnothing$? ". Although such a task is related to the indistinguishability task, it is not identical to it, since in addition to it there is also the systematicity task P(s). As a result, in order to prove the security of the third task, it is necessary to reduce its complexity to the deanonymization task P(d). But this is impossible, since the response to the observer becomes a procedure for violating the independence of the event x in the systematicity task. Due to this, observers can be sure that at the moment of the response dti, participant i could not simultaneously respond to another subscriber. Thus, the third task is not able to provide guarantees of anonymity and, through longer observation, can still exhibit patterns characteristic of the solution of the first task.

The problem can be solved by increasing the number of queues $Q \rightarrow \{Q1, Q2, ..., Qn\}$on one node with their binding to the subscribers of communication $i \in \{1, 2, ..., m\}$. In this case, the number of queues must be a priori set to a static value n, so that it is impossible to identify the number of linked friends m, i.e., the number of friends m must always have a comparison: $m \leqslant n$. As a result, the expansion of queues will lead either toincreasing the number of ciphertexts sent at a time: c1, c2, ..., cn $\leftarrow$t Q1, Q2, ..., Qn, or increasing the ciphertext generation period: ci $\leftarrow$nt Qi. This, in turn, will be a guarantee / proof that the active observer will not be able to influence the message queue of other subscribers, since all his actions will be tied and limited to one non-intersecting queue.

2.3. Comparison with other tasks

In terms of its characteristics, the QB task is closest to the DC task due to the following features: theoretically provable anonymity, periodicity of message generation, belonging to the second vector of anonymous communications development [2, p.71], complexity of scaling. The differences between QB and DC networks, from a positive point of view, are the following: the periodicity of generation can have a dynamic value, anonymity does not depend on the established connections with other participants, simpler software implementation. The negative difference is determined by the absence of information polymorphism.A more detailed and general comparison of the QB problem with other anonymization problems, both theoretical and practical, is presented in Table 1.

	QB	**E.I.**	**DC**	**Onion**	**Proxy**
Theoretical provability	+	+	+	-	-
Cumulative effect of anonymity	-	+	-	-	-
Information polymorphism	-	+	+	+	-
Probabilistic Routing	-	+	-	+/-	+/-
Frequency of message generation	+/-	-	+	+/-	+/-
Independence of anonymity from connections	+	-	-	-	-
Easy to scale	-	-	-	+	+

Simplicity of software implementation[2]	+	-	-	+/-	+
Stage of anonymity	5^	6	1^	4 or 6	3
Network representative	Hidden Lake	-	Herbivore	Tor	Crowds

Table 1. Comparison of anonymization tasks

3. Encryption function

As shown earlier, QB networks depend on the quality of the function E and the encryption keys k,r. The quality of the encryption keys is determined primarily by the quality of the RNG (random number generator) and/or the CPRNG (cryptographically secure pseudorandom number generator). The analysis of such generators is complex due to the different environments in which they are executed and the means they use during their execution [6, p.190]. Therefore, based on the logic of abstraction, we will further assume that the keys are generated in a high-quality and secure manner, thereby focusing exclusively on the logic of the execution of the encryption function.

$$E(k,privA,pubB)(m) = E''k(E'(privA,pubB)(m))$$

The encryption function in the Hidden Lake network consists of two stages, each of which has a clearly defined role. The first stage$E'(privA,pubB)$comes down to direct and primary encryption of data in order to hide it from third parties, using a hybrid encryption scheme (asymmetric + symmetric cryptography) [2, p.125]. The second stageE''_kIt comes down to separating several networks by using different encryption keys (network keys).

3.1. First stage of encryption

$$E'(privA,pubB)(m) = (EpubB(k') \,||\, Ek'(H(pubA) \,||\, s \,||\, m' \,||\, h \,||\, SprivA(h))),$$

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

*Where*k'- session encryption key calculated for one message, s - cryptographic salt calculated for one message, m - open message, pubA, pubB - public keys of participants A, B respectively, privA - private key of participant A, h - hash result, S - signature function, f - function of message complement to constant value, H - hash function, HMAC - function of calculating imitative insertion based on hash function H. In this scheme it is assumed that A is the sender of information m, B is the recipient of this information. The security of this function depends directly on the public encryption key pubB, which is used to encrypt the subsequent session keyk', from the quality of the RNG / KSGPSN which was generatedk', and also from the security of the encryption functions themselves EpubB, $E_{k'}$.

[2] The complexity characteristic can be determined by the quantitative ratio of the sum of successful passive/active observations to the sum of the necessary procedures to prevent such observations. If the sum of procedures is determined by a small number, then the simplicity of the software implementation will be minimal, which, however, does not indicate the overall security of the network, since a small number of procedures may also indicate the absence of measures taken to eliminate successful observations. In this case, the simplicity of the software implementation may be not only a consequence of high security, but also a consequence of a reduced threat model.

This scheme is interesting because it hides all the information in an encrypted shell, which does not allow attacks on the identification of the sender or recipient. For example, if the hash value h and the signature*SprivA(h)*were not in the encrypted block*Yes'*, then it would be possible to attack the analysis of encrypted messages using the existing list of public keys {pub1, pub2, ..., pubn}, checking their authenticity*Vpubi(SprivA(h))= h*. Further, if the cryptographic salt s and the hash value h were known, then it would be possible to compile a table of the most frequently occurring messages {m1, m2, ..., mn} with different combinations of participants i, j from the set of all network nodes N using the equality HMAC(s)(pubi, pubj, f(ml)) = h.

In addition, this scheme is self-sufficient [2, p.121] at the network level of QB-networks in the context of flood routing, because it allows for identification of subjects only and only with the help of asymmetric cryptography. It becomes possible to determine the sender of a message by means of correct decryption, i.e., provided that the recipient of the encrypted message has the necessary private key.

The stage assumes that the message f(m) has a static value. In other words, each time the encryption function is called,*E'(privA,pubB),* for all mi, mj from {m1, m2, ..., mn} the message length L from the function l is observed, such that l⇒*l(f(mi)) = l(f(mj)) = L*. This is made possible by the preprocessing procedure f, which limits the length of the input message to L and supplements the length of the input message to L. The purpose of such a procedure is to protect against attacks on message size analysis, which can reveal the structure of the transmitted message. For example, requests are often smaller in size than responses, transmitted video or audio files are often larger in size than regular text messages, system/automatic requests are smaller in size than manually executed requests, etc. [12].

It is also assumed that the recipient of the ciphertext has a list of public keys and their hashes in dictionary format, i.e. H(pubi) ↔ *pubi*. If decryption is successful, the user receives the transmitted value H(pubA), after which he tries to match it with the existing public key in the dictionary. If the public key pubA is not found in the dictionary, then the message is ignored. Otherwise, the sender's public key pubA, his own public key pubB, the cryptographic salt s, the message m' are taken and the correctness of the received hash sum h = H is checked by them.$_{MAC(s)}$*(pubA || pubB || m')*. If the sum is incorrect, then the message is ignored. Otherwise, using the public key pubA and the received value SprivA(h), the digital signature is checked for correctness. If the signature is valid, then the message m' is decoded f-1(m') → *m*and is accepted. Otherwise, the message is ignored.

Along with the peculiarity of hiding information in an encrypted shell, there is also a problem of checking the correctness of the ciphertext content, since the hash value and signature are internally encapsulated. In order to successfully validate an encrypted message, it is necessary to perform two additional decryption operations - with a private and session key, which is a resource-intensive action, mainly due to the use of an asymmetric algorithm.

3.2. Second stage of encryption

$$E''k(m) = Ek(p(h) \mid\mid h \mid\mid m),$$

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

*Where*k is the network key, p is the proof-of-work function, h is the hash result, m is the plaintext message. The function follows the MtE (MAC then Encrypt) approach, where the MAC (Message Authentication Code) is first calculated, and then the message m, the resulting code h, and the proof p(h) are encrypted by the Ek function.

This encryption stage performs one task - to differentiate different networks by the network key k, so that they cannot be merged into one common system. This is achieved mainly due to the proof-of-work function p, since it makes it more expensive to re-encrypt all traffic directed from one network with the key k1 to another network with the key k2,

The proof-of-work function p is defined by the proof-of-work (PoW) algorithm [13], where for a specific hash value h, it is necessary to find a number i such that the result hi = H(h || i) is a bit vector with a certain n-th number of zeros as a prefix, for example 00000000(n)...11001010. The number n is called the complexity of the work.

It is also worth noting that the encryption key k can be an open parameter if there is no need to form a specific group of nodes with a common secret. In this case, the network key k simply becomes a well-known setting for distinguishing networks.

The Encrypt-then-MAC (EtM) approach is not used in the second-stage encryption scheme for two reasons:

1. One key k is used for encryption and authentication instead of two keys k1, k2 for these tasks. If we apply the EtM approach in this situation, then two attack vectors will be opened for the same key k, instead of one. This problem could be solved by using a KDF (key derivation function), which would allow several keys to be created from one key. However, this would complicate the overall encryption scheme, and also open an additional attack vector for the KDF itself,

2. The Horton principle is taken into account: "authenticate not what is said, but what is meant" [6, p.130]. In case of a successful attack on the MAC calculation function in the EtM approach, or with an incorrect distribution of keys k1, k2, a situation may arise when authentication will give a positive result for the encrypted message, but the decryption procedure itself will be incorrect. If the message is a chaotic set of bits, then we will never know its truth.

The MtE approach certainly has a drawback in that it requires decrypting the information before checking its integrity and authenticity. The EtM approach does not have this problem. As a result, Marlinspike's principle of cryptographic doom was also formed, which states: "if you are forced to perform any cryptographic operation before checking the ciphertext of the received message, then this will inevitably lead to a fatal end one way or another" [14, p. 93]. This principle is opposed to the Horton principle when it comes to choosing one of the two approaches: MtE or EtM. However, the choice of EtM is also due to the fact that the principle of cryptographic doom is also violated at the moment of the first, and much more expensive, encryption stage.

4. Network interaction

The presence of a QB task assumes that each message sent in the network will reach all its participants. At the same time, such a task does not say anything about how the message will reach nodes in the absence of an all-to-all connection, what data transfer protocol will be used, how applications should work in such a model, how a decentralized structure will bypass NAT on the Internet, etc. All these questions require additional clarification based on the specific implementation of network interaction.

4.1 Microservice architecture

Hidden Lake Anonymous Network as an Application[3], is a set of services, each of which performs its own specific task [15].The choice of microservice architecture, as opposed to monolithic, was made taking into account the following aspects:

1. Microservice architecture allows to simplify and decentralize the development of application services, which makes it possible to use various programming languages and technologies when implementing your own applications.,

2. Microservice architecture allows you to separate the responsibility of services for processed or stored information, thanks to which, when the system becomes more complex, the correlation of the complexity itself will be minimally distributed to services,

3. Microservice architecture allows you to add new features, edit them, or remove them completely without having to recompile and restart all services, which can have a positive effect on both testing and fault tolerance.

The core of the Hidden Lake network is the HLS service (service), which directly executes the QB task and all encryption/decryption functions, respectively. In addition to the HLS service, the Hidden Lake network also has a number of application services, such as HLM (messenger), HLF (filesharer), HLR (remoter), a number of assistant services, such as HLT (traffic), HLL (loader), HLE (encryptor), as well as a number of HLA adapters, currently tied to the common (test service) and chatingar services. As a result, the Hidden Lake network can be represented as a composition of several services.

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

$$Hidden\ Lake = \Sigma ni{=}1APPi \times HLS \times (HLT \times \Sigma mj{=}1HLAj)t$$

where is APP- a set of application services, HLA - a set of adapters to third-party services, D - a deliverer of open messages to a specific application service, t = 0 or 1 - a parameter that determines the absence / existence of message relaying. In the absence of application applications, the fact of relaying, and, as a consequence, adapters, the Hidden Lake network becomes equal to the HLS service: Hidden-Lake = HLS. This is the minimum characteristic at which HL still remains itself. When the HLS service is removed, the system ceases to be a Hidden Lake network, since it loses its core.

Algorithm 3. Handling requests in an HLS application in pseudocode

```
INPUT: request req, sender s, service mapping M, filter function H
OUTPUT: rsp response OR end of algorithm
host, method, path, head, body ← req (* Receiving a request *)
if (host∉M) {(* Service not found in display *)
    return (* End of algorithm *)
```

[3] Hidden Lake anonymous network repository:https://github.com/number571/hidden-lake.

```
}
intReq ← M(host), method, path, (head || s), body(*Query enrichment*)
intRsp ← do(intReq) (* Request to service *)
if (noResponse(intRsp)) {(* No response received OR No response required *)
    return(*Algorithm completion*)
}
status, head, body ← intRsp(* Receiving a response *)
rsp ← status, H(head), body (* Response filtering *)
return rsp (* Sending response *)
```

As a result of all the above, the implementation of the Hidden Lake network can represent different communication modes: classic, relay and adaptation.

1. The classic communication mode is understood as direct or indirect connections between HLS nodes without the use of third-party assistant services. In such a model, each participant is both a producer of ciphertexts and their relay,

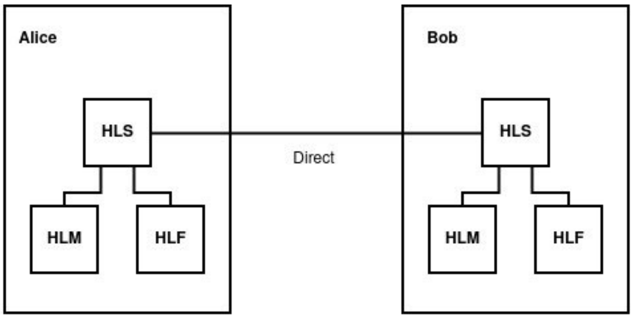

Figure 2. Classic mode. Hidden-Lake = (HLM+HLF) × HLS

2. The relay communication mode refers to connections between HLS nodes using HLT relay nodes. This communication mode allows NAT to be bypassed by using HLT services as dedicated TURN servers. The centralized nature of the servers will not affect the quality of anonymity due to the existence of theoretical provability in the form of the QB problem,

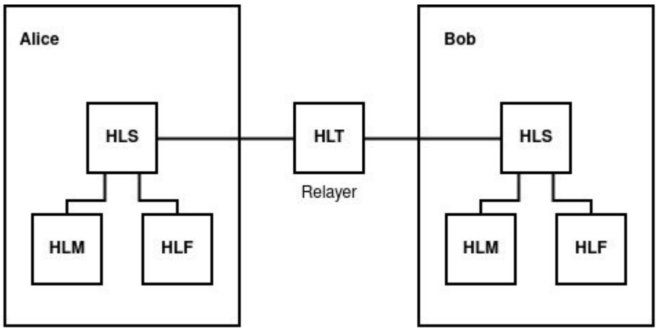

Figure 3. Relay mode. Hidden-Lake = (HLM+HLF) × HLS × HLT

3. The adaptive communication mode refers to connections between HLS nodes using HLA services. This mode allows for the injection of anonymized traffic into foreign/external systems that

are not connected in any way to the Hidden Lake network. This mode also uses the HLT assistant service, which allows for the translation of TCP traffic received at the output of HLS into HTTP traffic sent to the input of HLA.

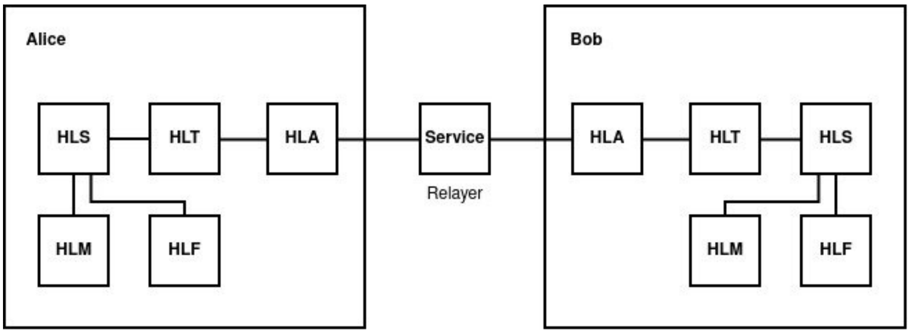

Figure 4. Adaptation mode. Hidden-Lake = (HLM+HLF) ✕ HLS ✕ HLT ✕ HLA=Service

4.2. GP/12 Protocol Stack

The abstract nature of the Hidden Lake anonymous network is formed by the GP/12 protocol stack (short for go-peer[4]and 1,2 – encryption stages), similar in essence to the TCP/IP protocol stack. It also has four levels: channel (CL), network (NL), transport (TL) and application (AL), but at the same time there are a number of the following differences:

1. GP/12identifies nodes by public keys rather than IP addresses,
2. GP/12can use the problem of theoretically provable anonymity,
3. GP/12does not depend on network protocols and communication systems,
4. GP/12 uses an end-to-end encryption scheme.

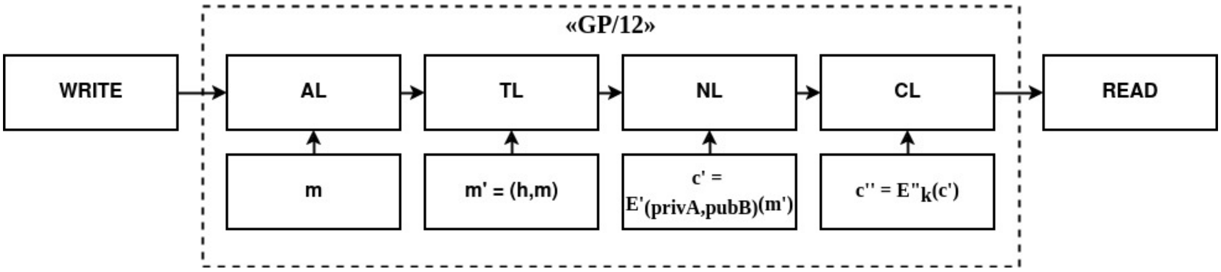

Figure 5.GP/12 protocol stack

The data link layer in the GP/12 network model is the distribution of messages using the flood routing method. This layer is characterized by the use of the second stage of encryption, when the very fact of successful transmission of the message to all its connections is important. The network layer is the distribution of messages to specific network nodes, using public keys as identifiers for this. This layer is characterized by the use of the first stage of encryption, when the confidentiality

[4] go-peer project repository:https://github.com/number571/go-peer. go-peer contains all the main modules: encryption functions, work with the message queue, network interaction. Hidden Lake, in turn, actively uses these components to implement services such as HLS, HLT, HLE. For the Hidden Lake network, the go-peer project was previously a framework due to their common affiliation with the code base and release versions. Now these projects are separated.

and authenticity of messages are important. The transport layer is the routing of open messages (successfully decrypted) to specific application services. The application layer is the acceptance of the open message with its subsequent processing. Thus, the GP/12 protocol stack can be depicted as a composition of four levels from the position of message acceptance: CL×NL×T.L.×AL.

As a result of the above, the execution of the full GP/12 protocol stack from the channel to the application layers is possible only with the use of at least two services in the Hidden Lake network, since HLS, being the core of the network, covers only the first three layers: CL, NL, TL, while the last layer AL can be covered only by using application applications such as: HLM, HLF, HLR.

Also, the GP/12 network model does not necessarily have to have the property of anonymity, which is why the HLS service, with a remote QB task in order to maintain GP/12 compatibility, can be replaced by a composition of three services: HLT× HLE× D, providing the first, second and third levels respectively. This is how the secpy-chat application was created[5].

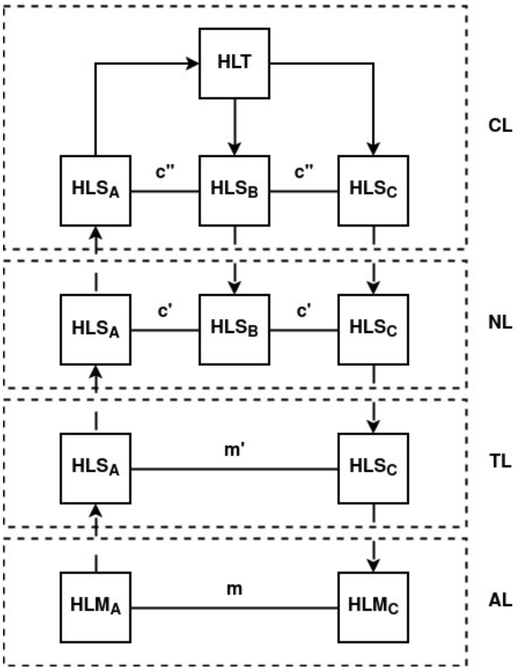

Figure 6.GP/12 model using Hidden Lake network in three-node relay mode as an example, where A is the sender, C is the recipient

Traffic anonymization can be formed at the second level of the GP/12 stack, when message generation is tied to some task. This can be either QB or DC, EI tasks. The main limitation in such a choice is the need for theoretical provability, without which it will be impossible to further form an abstract system with the property of anonymity. If Proxy or Onion tasks are used, then in this case the network based on the GP/12 network model will not become more anonymous, because these tasks, for their correct execution, require a non-monolithic system, which contradicts the definition of abstract anonymous networks, which do not care about system centralization.

For its implementation, the GP/12 protocol stack only requires the existence of WRITE and READ network interfaces, as a result of which it can be further adapted to a specific communication environment. For example, the GP/12 model in the Hidden Lake network is based on the TCP transport layer protocol by default, and with the help of HLA services (adapters) it can be based

[5]secpy-chat application repository: https://github.com/number571/secpy-chat.

both on top of application layer protocols: HTTP, SSH, FTP, etc. of the TCP/IP model [16], and even without TCP/IP at all, for example, by using radio broadcasting or a visible light system.

5. Software implementation

Mathematical models allow us to determine the correctness of the general logic of operation, but do not allow us to determine the safety of a specific implementation. For example, we can only assume that a function or the value it accepts will be safe. However, all this does not say anything about the specific implementation, the selected parameters, the threat model, and the design approaches. Thus, it is necessary to pay attention not only to the general description of the Hidden Lake network, but also to the detailed presentation of its structural and configuration parameters.

5.1 Structural parameters

1. The symmetric encryption function Ek is defined by the AES block algorithm with a key length of 256 bits in the CFB encryption mode (ciphertext feedback mode), where c0 = IV, ci = Ek(ci-1) xor mi.

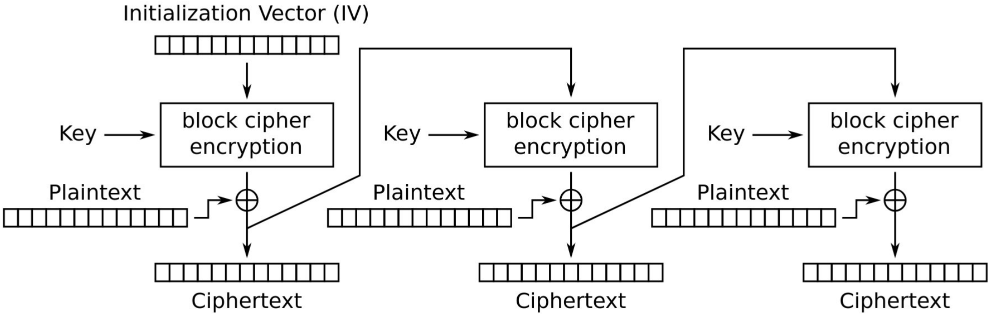

Figure 7.CFB encryption mode

This encryption mode was chosen taking into account the following points:

1. The CFB encryption mode does not require padding, as is required, for example, by the CBC encryption mode (ciphertext block chaining mode). As a result, it is easier to set a static encrypted message size, and also eliminates possible oracle attacks.[17],

2. The CFB encryption mode is not a stream encryption mode, such as OFB (output feedback mode) or CTR (counter mode). As a result, CFB does not have the problem of gamma repeatability. If the IV (initialization vector) is repeated, this will lead to much less security problems than with OFB, CTR modes [6, p.93],

3. CFB encryption mode does not require encryption algorithms to adhere to exact parameters, such as 128-bit blocks in GCM mode.(counter mode with Galois authentication)[18, p.190], and also does not require a decryption function from the algorithms, which allows not only to simplify the replaceability of insecure ciphers, but also to use one-way functions as an encryption procedure [19, p.282], *4.* The GCM encryption mode was also not used due to the unnecessary authentication and token storage operations. The first and second stages of encryption use different methods of message authentication. As a result, in terms of flexibility of use, the CFB mode becomes more preferable,

2. The asymmetric encryption function Epub is defined by the ML-KEM-768 algorithm (before standardization – Kyber-768) [20]. The asymmetric signing function Spriv is defined by the ML-DSA-65 algorithm (before standardization – Dilithium-M3) [21]. The choice of such algorithms was made taking into account the preparation of the Hidden Lake network for post-quantum cryptography. Establishing an F2F connection to a communication subscriber is actually determined by two public keys – ML-KEM-768 for encrypting Epub messages (recipient key) and ML-DSA-65 for verifying the signature of Vpub messages (sender key),

3. The network key in the second stage of encryption is assumed to change rarely, have high entropy, and is not a password. But since the network key does not have a fixed size, it is passed through the PBKDF2 key generation function to fix the size to 32 bytes, which is required by the AES-256 encryption algorithm. The cryptographic salt and the number of iterations in PBKDF2 are not specified, i.e. they are equal to the empty string and the number 0 by default. The hash function is SHA-512,

4. HMAC-SHA-384 is used as the impersonation algorithm (MAC). The security of HMAC depends on the hash function it uses [18, p.168]. The security of SHA-384 can be determined by its resistance of 384 bits for the problem of finding the first and second preimages, and by 192 bits for finding collisions, based on the birthday paradox attacks [6, p.52],

5. Quantitative and unchangeable parameters of the algorithms AES-256, SHA-384, ML-KEM-768, ML-DSA-64were chosen from a conservative point of view to maintain a sufficient level of security in the realities of post-quantum cryptography [5, p.131]. Grover's algorithm theoretically allows to reduce the security of symmetric ciphers to brute-force attacks and hash functions to collision searches, reducing the required number of calculations to 2n/2 and 2m/3, where n is the length of the symmetric cipher key, m is the size of the resulting hash function block. Thus, when using AES-256 and SHA-384, the minimum security will be determined by 128 bits, which is a stable value to brute-force attacks. In the absence of quantum computers, the minimum security will be determined by the ML-KEM-768 and ML-DSA-64 algorithms, the security of which is comparable to a 192-bit value,

6. Neither the first nor the second stage of encryption protects against a message replay attack [6, p.279], where after a certain period of time tthe attacker can retransmit an encrypted message that was previously saved by him. Such a message will be completely correct, since it was generated by one of the network participants. To protect against this type of attack, the Hidden Lake network uses the simplest and most radical method, saving message hashes in its local database. This approach allows to completely eliminate the possibility of successful acceptance of duplicates, including during a node reboot, and also to simplify the deduplication procedure in general by eliminating additional message checks within time windows.

Unfortunately, protecting a node from receiving and subsequently processing duplicate messages does not protect against an attacker reproducing messages that have not yet been received by the node directly owning the database. Such an event becomes possible if the node was absent during the periods of ciphertext generation by the remaining system participants. As a result, an attacker can load a newly arrived node with old and long-forgotten network messages. It is possible to protect against this attack by connecting to a long-running anonymizing node or relay that has saved the history of received messages. In this case, it will begin to act as a firewall, distinguishing between old and new ciphertexts. Another possible solution may be to update the network key k if a lot of traffic has been generated in the system, and long-running nodes are unknown or unavailable.

Adding a timestamp t to each message of the second encryption stage, on the contrary, can violate the anonymity of the QB problem, since if implemented incorrectly, it can create an additional connection in the generation of ciphertexts and make the problem of systematicity P(s) easily solvable. For example, when creating a true message, the timestamp will be fixed not at the moment of sending, but at the moment of generation. Due to the fact that messages from the queue are sent sequentially to the network, the pre-generated false ciphertext will have a timestamp close to the timestamp of the open message. In other words, the connection between the generation of true messages will be traced through the existence of two timestamps, the difference between which will be less than one period. The situation can be changed by replacing the false ciphertext with the true one, and not simply by moving the false ciphertext in the queue. But, in addition to this problem, the mechanism for generating ciphertexts must also take into account the moment of specifying the timestamp t depending on the generation period T. In other words, for the n-th message in the queue, the timestamp must be as follows: t = now + nT, where now is the current time. This solution can also have a negative effect if the speed of generating ciphertexts, for example, when the node is heavily loaded, does not keep up with the timestamp. In this case, desynchronization of the queues by the timestamp between true and false messages may occur. The situation can be changed by constantly removing old generated ciphertexts from the queue, but in this case there is a risk of losing true messages that were not even sent to the network,

7. Unlike the classical description of QB networks, in the implementation of the go-peer project the structure of the queues Q is represented by two queues: a queue of true Qk and a queue of false Qr messages, which are then merged into one. The need for two queues is due to the need for background generation of false messages, so that when the period t is reached, the queue Qr is predominantly non-empty. In turn, this need is directly related to the proof-of-work algorithm, which significantly slows down the encryption of messages, as a result of which the set generation period t can be extended. Although the situation in this case is similar to Example 3.3, nevertheless, the systematicity of the algorithm is not violated here, since the generation of true and false messages occurs not at the moment of their sending, but at the moment of placing them in the queue,

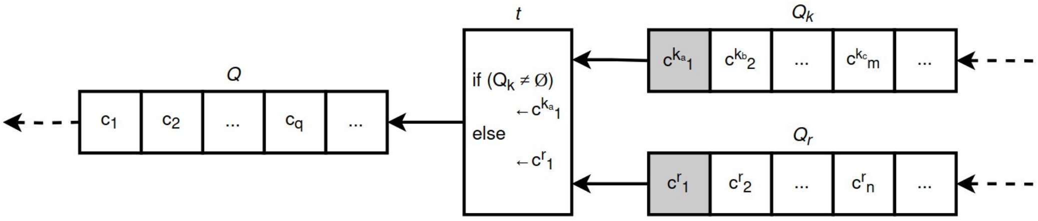

Figure 8. Double message queue diagram in go-peer project

8. The Hidden Lake anonymous network may be vulnerable to active cooperating observers if they are present in the friends list of several subscribers of the system. In this case, in addition to the deterioration of the anonymity quality: from the task of hiding the fact of communication to hiding the connection between subscribers of communication, there will also be a risk of solving / deanonymizing the QB problem itself. The problem could be solved by creating several queues

Qk1, Qk2, ..., Qkn for each individual subscriber, but in this case the total response time will increase n times. Another way to solve this problem may be to change the threat model, in which a friend will be a priori equal to a trusted node, and therefore will not be interested in and involved in deanonymization procedures. The Hidden Lake network is based on such a design of F2F systems,

9. MThe threat model of the Hidden Lake anonymous network is limited exclusively to protecting network communications between its participants, which, in turn, implies the absence of any additional measures, actions and conditions aimed at protecting private keys, databases, configuration files or interactions of services with each other in the conditions of the local execution environment. As a result, the measures taken to ensure the security of the local environment, in the conditions of the existence of sensitive information, should be assigned to lower levels of interaction, such as: setting access rights to files and processes, isolating programs through the use of a virtual environment, software and hardware full-disk encryption, restricting physical access to hardware for third parties, etc.

5.2 Configuration parameters

1. The approach of storing hashes of all previously received messages also has a disadvantage in the face of the tendency of the database volume to constantly increase. As soon as the hash database appears, it should never be deleted or cleared (without changing the network key), otherwise all previously received messages can be repeated again if the attacker uses a replay attack of repeated messages. In such a paradigm, the question begins to be based on the volume and frequency of constantly generated information. The message hash in the Hidden Lake network is the hash value h = HMAC(k) obtained from the information in the second stage of encryptionE''_k. The hash size is determined by the cryptographic hash function SHA-384, i.e. 48 bytes. Thus, one received or sent unique message will increase the DB by 48 bytes. The period of generation of one message by one node is 5 seconds. If we assume that there are 10 nodes in the network, each of which generates one encrypted message in a period of 5 seconds, then 480 bytes will be generated and saved in 5 seconds, or 96 bytes per second. Further, if we extend the obtained result to a year, the DB will be increased by ~2.6 Gb[6]information, which is a completely acceptable value,

2. With each hash stored in the database, the total area of valid ciphertexts decreases due to the accompanying collisions inherent in all hash functions. As a result, the more ciphertexts are accepted, the fewer of them can be accepted in the future. The probability of finding a collision in more than 50% of cases begins after overcoming the threshold of 2192 accepted ciphertexts for the SHA-384 hash function using classical computers. After this, the network will begin to work unstable, discarding half of all messages received by the system, and the only way to "restart" the network will be to delete the database and then change the network key. But such an event is unrealistic, since it requires saving249230249209671726169463823802564608 YiB[7]information for 2192 SHA-384 hashes. If there are 8000 nodes in the network with a ciphertext generation pattern of 1.6KiB/s, i.e. when the communication channel limit of 100 Mbit/s is reached, ~2076 GiB will be stored in the local database per year. At this rate, it will take more than 1047 years to overcome the threshold of finding collisions of >50% of cases.[8].

[6] ((48 [bytes]×10 [knots] / 5 [seconds])×60 [seconds]×60 [minutes]×24 [hours]×7 [days]×4 [weeks] + 2 [days])×12 [months] / 230 [GiB] = 2.595520041882992≈2.6GiB.

[7](2192[hashes]×48[bytes] /280[YiB]) = 249230249209671726169463823802564608 YiB.

[8](2192[hashes]×48[bytes]) / ((48 [bytes]×8000 [nodes] / 5 [seconds])×60 [seconds]×60 [minutes]×24

3. Messages at the first stage of encryption have a static size of 8192 bytes, of which 4569 bytes are allocated to header data: initialization vector (16B), data hash (48B), ML-DSA-65 signature (3309B), sender's public key hash (48B), salt (32B), ML-KEM-768 encapsulated encryption key (1088B), as well as the data sizes in bytes: hash (4B), signature (4B), public key (4B), salt (4B), encrypted data block (4B), payload (4B), padding bytes (4B). The second stage of encryption adds a minimum of 84 bytes (if there are no empty bytes), of which: initialization vector (16B), proof of work (8B), hash (48B), network mask (4B), number of empty bytes (4B), message size (4B). The message size after full encryption is 8276 bytes, of which 3623 bytes are payload. Thus, if the message generation period is 5 seconds, then the node can transmit ~724.6 significant bytes per second. If the application assumes request-response communication, then as a result of the request made, the response waiting stage will inevitably begin, which will lead to a twofold decrease in throughput to ~362.3 significant bytes per second due to the increased waiting interval of ~10 seconds. In total, if there is a task to transfer a 1 MiB file over the network, then the transportation will take approximately ~24.1 to ~48.2 minutes (~1447 and ~2894 seconds, respectively),

4. The complexity of the Hidden Lake network is determined by the work_size_bits parameter, which is 22 bits in the global configuration. This parameter limits the throughput of each network node, thereby counteracting DoS and DDoS attacks, as well as attacks aimed at duplicating messages across multiple subnets. This parameter is able to determine the approximate quantitative limit of network participants. Let's assume that the throughput of each node in the network is 100 Mbps, the size of the generated message is 8 KiB, and the generation period is 5 seconds. In this case, network overload will be possible if there are more than 8000 nodes[9]. This, in turn, is an optimistic estimate, because it does not take into account malicious actions from the nodes themselves, capable of generating traffic of more than 8 KiB/5s = 1.6 KiB/s, and also does not take into account the ability of nodes to decrypt received traffic in a timely manner.

Without taking into account the work_size_bits parameter (the second stage of encryption), the natural limitation for attackers is the time it takes to encrypt messages (the first stage of encryption). On an "Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz" processor, encrypting one 8KiB message takes approximately 0.254ms, which requires an average of 0.4 seconds.[10]to reach the maximum value of 12.5 MiB, with the further possibility of overflowing the network channel of 252 Mbit/s = 12.5 / 0.4. As a result, 8000 honestly working nodes become equal to less than one malicious one≈0.4. To prevent such a difference, a parameter is introduced that determines the complexity of the work. With work_size_bits=22, encryption of one 8KiB message takes approximately 1.98s on the same processor without taking into account the time of the first encryption stage, which on average requires 52.8 minutes.[11](3168 seconds) to reach the limit of 12.5 MiB. Thus, to reach the limit, an attacker would need ~3168 parallel calculations of the same processor power to be able to overcome the channel bandwidth of 100 Mbit/s.

[hours]×7 [days]×4 [weeks] + 2 [days])×12 [months] = 135140700251269152632028 727792496229215884627189 > 1047 years.

[9] (100[Mbps] / 8[bits] = 12.5[MiB/s]) / ((8[KiB]/5[s] = 1.6[KiB/s]) / 1024[bytes] = 0.0015625[MiB/s])≈8000 nodes, each generating 1.6 KiB/s.

[10] (8[KiB] / 0.254[ms]) → (1.6[KiB] / 0.0508[ms]), 0.0508[ms] × 8000[packets of 1.6 KiB] = 406.4 ms ≈ 0.4 s.

[11] (8[KiB] / 1.98[s]) → (1.6[KiB] / 0.396[s]), 0.396[s] × 8000[packets of 1.6 KiB] = 3168 s = 52.8 m.

The generation time of 1.98 seconds is close to half of 5 seconds, which is the most relevant value for successful accumulation of true and false ciphertexts. With work_size_bits=23, the generation time will be ~3.96 seconds, which, with the probability of additional loads, will lead to an implicit increase in the message generation period > 5 seconds, facilitating further deanonymization of subjects by identifying periods of the most intense activity.

6. Conclusion

In the course of the work, the functioning of the Hidden Lake anonymous network was analyzed based on its mathematical models, with further indication of the advantages and limitations resulting from the QB task. The main network services were given, as well as their network interactions with each other in the microservice architecture paradigm. The GP/12 protocol stack was described, allowing networks with theoretically provable anonymity to become abstract. The criteria for choosing encryption, signing, hashing algorithms, their parameters and settings at the time of network design and implementation were presented.

References

1. Popescu,B., Crispo, B., Tanenbaum, A. Safe and Private Data Sharing with Turtle: Friends Team-Up and Beat the System[Electronic resource]. — Access mode:https://web.archive.org/web/20070316085325/http://www.turtle4privacy.org/documents/sec_prot04.pdf (date accessed: 04.07.2024).
2. Kovalenko, G. General Theory of Anonymous Communications. Second edition / G. Kovalenko.— [b. m.]: Publishing Solutions, 2023. - 208 p..
3. Schneier, B. Applied cryptography. Protocols, algorithms and source codes in C / B. Schneier. - St. Petersburg: OOO Alfa-kniga, 2018. - 1040 p..
4. Alferov, A., Zubov, A., Kuzmin, A., Cheremushkin, A. Fundamentals of cryptography: Textbook / A. Alferov, A. Zubov, A. Kuzmin, A. Cheremushkin. - M.: Helios APB, 2002. - 480 p.
5. Grimes, R. The Apocalypse of Cryptography / R. Grimes. - M.: DMK Press, 2020. - 290 p.
6. Schneier, B., Ferguson, N. Practical Cryptography / B. Schneier, N. Ferguson. - M.: Publishing House "Williams, 2005. - 420 p.
7. Tanenbaum, E., Weatherall, D. Computer networks / E. Tanenbaum, D. Weatherall. - St. Petersburg: Piter, 2017. - 960 p.
8. Perry, M. Securing the Tor Network [Electronic resource]. — Access mode:https://www.blackhat.com/presentations/bh-usa-07/Perry/Whitepaper/bh-usa-07-perry-WP.pdf(date accessed:04.07.2024).
9. Astolfi, F., Kroese, J., Oorschot, J. I2P - Invisible Internet Project [Electronic resource]. — Access mode:https://staas.home.xs4all.nl/t/swtr/documents/wt2015_i2p.pdf(date accessed:04.07.2024).
10. Danezis, G., Dingledine, R., Mathewson, N. Mixminion: Design of a Type III Anonymous Remailer Protocol [Electronic resource]. — Access mode:https://web.archive.org/web/20170312061708/https://gnunet.org/sites/default/files/minion-design.pdf(date accessed:04.07.2024).
11. Reiter, M., Rubin, A. Crowds: Anonymity for Web Transactions[Electronic resource]. — Access mode:https://www.cs.utexas.edu/~shmat/courses/cs395t_fall04/crowds.pdf(date accessed:04.07.2024).

12. Ishkuvatov, S. Method and algorithm for determining the type of traffic in an encrypted communication channel [Electronic resource]. — Access mode:https://cyberleninka.ru/article/n/sposob-i-algoritm-opredeleniya-tipa-trafika-v-shifrovannom-kanale-svyazi(date accessed:04.07.2024).
13. Nakamoto, S. Bitcoin: A Digital Peer-to-Peer Cash System [Electronic resource]. — Access mode:https://bitcoin.org/files/bitcoin-paper/bitcoin_ru.pdf(date accessed:04.07.2024).
14. Khlebnikov, A. OpenSSL 3: the key to the secrets of cryptography / A. Khlebnikov. - M .: DMK Press, 2023. - 300 p..
15. Lewis, J., Fowler, M. Microservices [Electronic resource]. — Access mode:https://martinfowler.com/articles/microservices.html(date accessed:07.09.2024).
16. Kovalenko, G.Anonymous P2P network inside a centralized HTTPS server: we sew parasitic traffic by hook or by crook [Electronic resource]. — Access mode:https://habr.com/ru/articles/800965/(date accessed:07.09.2024).
17. Heaton, R. The Padding Oracle Attack[Electronic resource]. — Access mode:https://robertheaton.com/2013/07/29/padding-oracle-attack/(date accessed:04.07.2024).
18. Aumasson, J. Taking Cryptography Seriously. A Practical Introduction to Modern Encryption / J. Aumasson.— M.: DMK Press, 2021. - 328 p..
19. Mao, V. Modern cryptography: theory and practice / V. Mao. - M.: Publishing house "Williams, 2005. - 768 p.
20. Module-Lattice-Based Key-Encapsulation Mechanism Standard [Electronic resource]. — Mode:https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf (date accessed:10/19/2024).
21. Module-Lattice-Based Digital Signature Standard [Electronic resource]. — Mode:https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf (date accessed:10/19/2024).