Solving the Byzantine Postman Problem

Len Sassaman[1] and Bart Preneel[1]

Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{len.sassaman,bart.preneel}@esat.kuleuven.be

Abstract. Over the last several decades, there have been numerous systems proposed which aim to preserve the anonymity of the recipient of some data. Some have involved trusted third-parties or trusted hardware; others have been constructed on top of link-layer anonymity systems or mix networks.
In this paper, we examine the Pynchon Gate [34], a pseudonymous message system which takes an alternate approach to this problem by using Private Information Retrieval (PIR) as the basis for its pseudonymity properties. We restrict our examination to a flaw in the Pynchon Gate system first described in our technical report [35]; as it was originally presented, the Pynchon Gate detects the presence of (and protects against certain attacks by) Byzantine servers operating in the system, but it fails to identify *which* server or set of servers is Byzantine, thus opening the door for denial of service attacks as well as other potential anonymity compromises by Byzantine servers.
We show a trivial modification to the original PynGP which allows for detection and identification of Byzantine nodes, with no weakening of the security model necessary, at the relatively affordable cost of greater bandwidth requirements during certain communication operations. We demonstrate that this adequately solves the problems raised by [35], and argue that it is the most suitable method of addressing the attack in question yet proposed.
We then evaluate an alternate approach to solving to the problem described in [35], proposed by Goldberg in his recent paper [21]. We compare the security and performance trade-offs made in that proposal, and find it less secure against anonymity attacks as compared to the original (but flawed) Pynchon Gate Protocol (PynGP) [24] presented in the first Pynchon Gate paper. We show that this proposal is significantly weaker than the solution offered in this paper, which retains the security properties of the original Pynchon Gate Protocol.

1 Introduction

Several proposals have been made for the use of private information retrieval (PIR) [8] primitives to build secure, fault-tolerant pseudonymous mail retrieval systems [10, 3, 23, 34].

PIR-based pseudonym (or *nym*) servers have several significant advantages over nym servers based on other technologies. PIR protocols can be designed to

offer *information-theoretic security*, i.e., assuming that the system is correct, an attacker with unlimited computational power cannot defeat the system merely by virtue of being able to perform calculations which reveal the private information. Other PIR protocols merely offer computational security: in Computational PIR systems [7], the privacy of the PIR query is protected only against an adversary restricted to polynomial-time computational capability. CPIR-based solutions have the significant advantage that they can be performed using a single server, and do not require distribution of trust to ensure that the information retrieval requests remain private. However, such systems presently have prohibitive computational cost on commodity hardware.

1.1 Distribution of Trust in Public Anonymity Systems

In distributed-trust information-theoretic PIR systems, the privacy of the system is predicated upon no single entity being able to gain access to sensitive data, be it the private information itself, or second-order information which can be used to obtain information about the private information. Of particular concern are the possibilities that a single adversary may operate multiple nodes under different identities, effectively ensuring node collusion [19], or that significant amounts of the anonymity infrastructure may lack good *location independence* [20]. Thus, systems which encourage participation by many unaffiliated operators of diverse backgrounds across a wide range of network providers can provide stronger services than those in which infrastructure operation is more tightly restricted. Rather than attempting to ensure that the adversary is unable to gain control of any part of the infrastructure, this *laissez-faire* approach to anonymity service operation taken by some of the more successful anonymity services [17, 28] simply accepts that some nodes will be controlled by an adversary, and accounts for this fact in the design of these systems.

1.2 Background on Nym Servers

Pseudonymous messaging services allow users to send messages that originate at a pseudonymous address (or "nym") unlinked to the user, and to receive messages sent to that address, without allowing an attacker to deduce which users are associated with which pseudonyms. These systems can be used for parties to communicate without revealing their identities, or as a building-block for other systems that need a bi-directional anonymous communication channel, such as Free Haven [15].

1.3 The Pynchon Gate and The Byzantine Postman Problem

The most recent proposal for a nym server based on PIR with information-theoretic security, the Pynchon Gate [34], offers greater robustness, stronger anonymity assurances, and better traffic analysis resistance than previously proposed pseudonym systems. However, as we have previously noted [35], it contains a flaw in its protocol which can be used to launch a denial of service attack

against the system, rendering it unusable.[1] Furthermore, the attack is not merely limited to decreased utility of the system; due to the network-effects properties of anonymity systems, denying service to one set of users can effectively weaken the anonymity provided to a different set of users [1].

2 Background on The Pynchon Gate

To address the reliability problems of silent node failure, as well as the serious security problems of statistical disclosure attacks [33, 11, 13] and end-to-end traffic analysis [26], Sassaman, Cohen, and Mathewson propose a complete architectural design of a PIR-based pseudonym service offering information-theoretic protection, called the Pynchon Gate [34].

2.1 Architecture Overview

The architecture of the Pynchon Gate consists of an Internet-facing component referred to as the "nym server", which receives messages addressed to users of the system and acts as a gateway between the pseudonym service and other Internet services such as email. Behind the nym server is a component known as the "collator", which structures the incoming messages in the form of a three-level hash tree, which is then replicated to a series of mutually untrusted distribution databases referred to as "distributors".

Email addressed to a specific pseudonym is stored in a specific location in the database, such that the owner of the pseudonym knows what information to request to obtain his message. Using the PIR protocol described in Section 2.2, the user submits a PIR query to ℓ distributors, and his message is returned with none of the distributors able to deduce any information about the user's query unless all ℓ distributors collude. This form of PIR is referred to as an **information-theoretic $(\ell - 1)$-private ℓ-server PIR** protocol.

2.2 The Pynchon Gate PIR Protocol

The protocol runs as follows: after choosing distributors, the client establishes an encrypted connection to each (e.g., using TLS [14]). These connections must be unidirectionally authenticated to prevent man-in-the-middle attacks, and can be made sequentially or in parallel.

The client sends a different "random-looking" bit vector $\vec{\nu}_{s\beta}$ to each distributor s for each message block β to be retrieved. Each bit vector has a length

[1] If one or more of the servers in the system is Byzantine, the protocol will detect that the results of the PIR request are not correct, and will treat the results as poisoned, thus eliminating potential direct attacks by Byzantine servers against the *privacy properties* of the system. However, the specific PIR protocol used in the Pynchon Gate is designed in such a way that it is impossible to know which server was Byzantine, and therefore Byzantine servers can act with impunity, inevitably locking every user of the system into a state of constant denial of service.

equal to the number of message blocks in the database. Each distributor s then computes $R(\vec{\nu}_{s\beta})$ as the exclusive-OR of all message blocks whose positions are set to 1 in $\vec{\nu}_{s\beta}$. The resulting value is then returned to the client.

Thus, in order to retrieve the β'th message block, the client need only choose the values of $\vec{\nu}_{s\beta}$ such that when all $\vec{\nu}_{s\beta}$ are XORed together, all values are 0 at every position except β. (For security, $\ell - 1$ of the vectors should be generated randomly, and the bit vectors should be sent in a random order so that the ℓ'th, specially-crafted vector cannot be distinguished.) When the client receives the corresponding $R(\vec{\nu}_{s\beta})$ values, she can XOR them to compute the message block's contents.

2.3 Byzantine Server Protection

In a distributed-trust anonymity system such as the Pynchon Gate, there exists the possibility that some servers may be *Byzantine*, i.e., they may behave incorrectly, either due to intentional malice or simple error.[2] In the case of the Pynchon Gate, the Byzantine behavior we are concerned with is an incorrect response to a PIR query of a distributor's database.

All n distributors in the system have the exact same copy of the database, and the system is designed such that any attempt by a Byzantine server to modify its response to the PIR query will be detected by the user when he verifies the root of the hash tree. This is crucial to preserving the anonymity properties of the system, for if an attacker can alter a message or observe the cleartext of a message, he may potentially be able to later link an input message with a given output retrieved by the nym holder.

The Pynchon Gate's message and link encryption prevents an attacker from observing the cleartext of a message. Active attacks that are dependent upon the attacker's ability to alter some of the data being transmitted to the user such that the attacker may later link the user to his pseudonym based either on a variance in the user's response to altered versus unaltered data, or by simply recognizing the product of the altered data as it is processed by the system (collectively known as *tagging attacks* [18]) are ineffective, as TLS protects data integrity on the wire. Thus, any tagging attacks an attacker wished to attempt against a user would have to occur through the use of a corrupt distributor. To protect against the case where a distributor provides (intentionally or otherwise) an incorrect response to the PIR query, the client verifies that the hash of the message block it has received can be authenticated through the hash tree with the verified hash root.

3 A Remaining Byzantine Server Attack

We present the following attack not prevented by the hash tree verification system: a corrupt distributor can, through malice or error, create a denial of service

[2] This concern is present in many other anonymity systems, including Chaumian mixnets [6, 28, 12] and systems built on top of them [27, 25].

attack on the system by responding with incorrect data to a client's query. While the client will detect that the message block is invalid after performing the final step of the PIR protocol in Subsection 2.2, and thus can conclude that *some* server was Byzantine, the client cannot determine *which* server or servers returned the incorrect response. The client cannot safely pass the message block contents (assuming they consist of anything other than garbage) to the user, lest the user's anonymity be potentially compromised.

Furthermore, if attacks on portions of the pseudonymity infrastructure affect some users differently than others, an attacker may exploit such attacks on components of the system to facilitate an intersection attack against a user of the system as a whole [16]. In the Pynchon Gate, if a Byzantine distributor selectively performed denial of service attacks against certain users by returning garbage results to their queries, but correctly responded to other users' queries, the attacker would increase his chances of learning the identity of certain users, based on which users responded to messages that were successfully delivered.[3] In other cases, a passive adversary could observe the actions of Byzantine servers not under his control (and perhaps not even behaving maliciously, but simply incorrectly) to help facilitate intersection attacks [38]. Additionally, if a user cannot know with confidence which server is behaving in a Byzantine fashion, she is more likely to change the nodes she uses on a regular basis, both increasing her exposure to long-term intersection attacks and increasing the probability of selecting a server-set that consists of nodes operated entirely by a single adversary.

4 Byzantine Server Detection

Ideally, there would exist a way to identify an individual Byzantine server without modifying the existing threat model or positive security properties of the Pynchon Gate. This is a challenging problem to solve with the existing XOR-based PIR protocol, which makes verifying the results of a PIR query returned by a specific distributor impossible. (The client does not know what a "correct" response $R(\vec{\nu}_{s\beta})$ from any given distributor should look like; only that

$$R(\vec{\nu}_{s_1\beta}) \oplus R(\vec{\nu}_{s_2\beta}) \oplus \cdots \oplus R(\vec{\nu}_{s_\ell\beta}) = \beta' th \text{ message block}$$

and thus cannot identify which of the responses were invalid.)

5 A Novel Solution to the Byzantine Postman Problem

We propose a solution to the Byzantine Postman Problem which preserves the same threat model established for the original Pynchon Gate Protocol (hence-

[3] This type of attack is present (in a slightly different form) in non-PIR-based nym server systems as well. For instance, in a reply-block system, an attacker could disable certain mixes and observe which nyms ceased receiving traffic. If the nym holder has a fixed-route reply-block, this would enable the attacker to identify the mixes used in the nym holder's reply-block path, and increase his chances of successfully linking the nym with the nym holder's true name [36].

forth referred to as PynGP 1.0). Described below, our revised PynGP (PynGP 2.0) protocol relies only on additional sets of operations already performed by PynGP 1.0, yet this modified version of PynGP 1.0 allows for the detection and identification of Byzantine nodes with sufficient probability that our denial of service attack against the PynGP 1.0 is no longer feasible in practice. Furthermore, while it increases the amount of communication bandwidth needed to perform PynGP operations, the communcation overhead is still within the realm of affordability for the target user and operator demographic stated in [34].

5.1 PynGP 2.0

We have modified PynGP 1.0 only as much as necessary to address its known security flaws. The revised version of the protocol retains the same security properties set forth in the original design paper. We address the issue of Byzantine nodes by introducing a *cut-and-choose* methodology [32, 4]. To support this addition, we modify the query algorithm and add a response validation algorithm (to be run if the reconstruction algorithm fails) at the cost of trivial computation expense and a doubling in the bandwidth needed to perform queries of the database.[4] Finally, we introduce a new component in the Pynchon Gate architecture, known as the *validator*.

Thus, PynGP 2.0 is identical to PynGP 1.0 as described in Subsection 2.2, with the following modifications to the protocol:

Each time the protocol runs, the client prepares two sets of bit vectors to send to the chosen distributors. The first set, $\{\vec{\alpha}\}$, is used to obtain the private mailbox data via the PIR protocol; the second set, $\{\vec{\eta}\}$, is used to challenge the honesty of each distributor.

At the step in the PynGP 1.0 protocol where the client would transmit the "random-looking" bit vector to each distributor, the client submits two "random-looking" bit vectors instead, one from $\{\vec{\alpha}\}$ and one from $\{\vec{\eta}\}$, transmitted in a random order.

Upon receiving these bit vectors, the distributor performs the operations as described in Subsection 2.2, and then returns two responses, in the same order which the requests were received (or otherwise in such a manner that the responses are linkable to the requests which generated them). The client caches the response for the $\{\vec{\eta}\}$ request, then performs the PIR operation as previously described using the $\{\vec{\alpha}\}$ results from all distributors.

The Validator: We now introduce a new component in the Pynchon Gate architecture, known as the *validator*. This component is essentially a distributor,

[4] While this is indeed a significant increase in system overhead, it is still feasible, especially given the Pynchon Gate's resource tuning properties which permit a linear trade-off between bandwidth and storage, adjusted simply by changing the size of the message blocks stored in the database. Doubling the size of the message blocks halves the bandwidth necessary to perform a query.

except that it only exists to confirm that the other distributors are not Byzantine. This specialized distributor validates the "cut-and-chosen" responses as being correct, deterring the operation of Byzantine nodes and probabilistically uncovering them should they exist.

To ensure that additional trust is not required of this new component to the system, the validator **must** be operated by the same entity who operates the collator. The operator of the collator is already empowered to perform a denial of service attack by simply unplugging the power cord of the server running the collator. Ergo, the balance of power in this distributed trust system is maintained by placing the validator (whose operator could also force a denial of service attack, though not as easily) under the control of the same entity.[5] Note that communication with the validator occurs over an encrypted link, just as with normal distributors.

Auditing the Distributors: The requests comprising set $\{\vec{\eta}\}$ are crafted such that they return a specific *validation block* when the PIR algorithm is performed. Under normal circumstances, the contents of this block are of no interest to the user. The individual responses are cached by the client, along with the corresponding request that was sent to the distributor to generate them as well as an identifier for the distributor which returned the responses. To verify that a distributor is not attempting to behave in a Byzantine manner, the same bit vectors in $\{\vec{\eta}\}$ that had been submitted to each distributor can subsequently be submitted to the validator, and the validator should return a response identical to that which the original distributor returned for each request. (The entire $\{\vec{\eta}\}$ needs to be submitted, as there may be multiple Byzantine servers acting simultaneously.) Should the validator return a response that differs from the one received by the client from a given distributor, that distributor should be suspected of being Byzantine.[6]

Auditing the Validator: The addition of the validator component does raise the concern that a corrupt nym-server/collator/validator coalition may attempt to mount an attack on a user's anonymity by systematically framing honest distributors as Byzantine nodes so that the user selects only nodes operated by the coalition. As one of the main premises behind the security of the Pynchon Gate design is that the nym-server operator not be trusted to preserve the

[5] The validator is never provided the contents of $\{\vec{\alpha}\}$ queries, since the validator, collator, and nym server are *not* considered trusted with regard to a user's privacy, and knowledge of the contents of $\{\vec{\alpha}\}$ queries (or responses) could provide information about the user's identity.

[6] Strictly speaking, the $\{\vec{\eta}\}$ vectors should always be sent to the validator, regardless of the outcome of the PIR operation using the $\{\vec{\alpha}\}$ results, to guard against the scenario where an additional adversary not in collusion with the Byzantine server might learn additional information about the state of the network. However, this level of paranoia may not be affordable until bandwidth and computation become less expensive.

user's anonymity, a way to confirm that the validator is honest is needed. This confirmation procedure is simple:

If the $\{\vec{\eta}\}$ responses from the distributors differ from the $\{\vec{\eta}\}$ responses from the validator, the client should first attempt to verify the correctness of the $\{\vec{\eta}\}$ response by performing the PIR protocol and comparing the result to the known validation block. If the correct block is returned, there is nothing to be learned by querying the validator. If the responses to the $\{\vec{\eta}\}$ requests yield an incorrect validation block, the presence of at least one Byzantine distributor is verified. The client then proceeds as described above, submitting each $\{\vec{\eta}\}$ query to the distributor *one query at a time* and recording the results. When a differing result is received for a given query, it should be swapped in for the original result, and the PIR protocol performed. The substitution of the validator's response for the original response should yield the correct validation block if the validator is honest. In cases where the validator's responses differ for more than one query, this challenge should be performed for each differing response both individually, and as a whole.

Probability of Byzantine detection: As proposed above, this protocol gives a Byzantine server a 50% chance of being discovered each time it attempts to behave in a Byzantine manner. That threshold can be increased at the expense of greater bandwidth overhead; however, we feel that a 50% detection rate is sufficient to deter this sort of attack, due to the inherent reputation system involved with the distributor network.[7] This probability of detection is based on the assumption that the Byzantine server considers it acceptable that its Byzantine action may be *ineffective*. If the server wishes to *guarantee* a successful Byzantine operation, it can do so by providing Byzantine answers to all the client's requests, but the probability of detection in that case is 100%.

One Byzantine action by a distributor *verifiable as such* to the collator, validator, or nym server operator should be sufficient to blacklist a Byzantine distributor. Care must be taken to ensure that an attacker does not frame an honest server as Byzantine to have it blacklisted. Multiple reports identifying a given server as Byzantine might simply indicate a Sybil attack being performed to achieve the blacklisting of an innocent server.

6 Remarks on Performance

As previously stated, PynGP 2.0 has a higher performance cost than its predecessor. We have attempted to strike a balance between security and excessive resource consumption when the security issues are unlikely to be problematic, or the resource requirements too great to qualify the protocol as "deployable". Areas where security could be increased, if performance cost were no object, include

[7] A Byzantine server's chances of successfully providing a Byzantine response of unidentified origin decreases in an manner inversely proportional to its probabilty of detection.

the addition of more than a single challenge-tor set to decrease the probability of a Byzantine server successfully avoiding detection.[8] Also, as previously noted, it would be ideal, were it affordable, for the challenge-vectors to be validated every time a PynGP protocol run was performed. This is likely cost-ineffective, though, given that knowledge of the user's failure to validate the hash root is unlikely to give an adversary any significant advantage, let alone lead to a user-level privacy vulnerability. It is far more important that all clients in the system behave according to the same policy in this regard. Also, if the number of challenge-vector sets is increased, the cost of validation increases proportionally.

The resource requirements for the validator must be more fully investigated, and will not truly be known until the system is tested in a live environment with actual Byzantine nodes. It is conceivable that the validator might best be deployed as multiple load-balanced servers, should the level of resource consumption warrant it. It is also conceivable that there may be few to no challenge validations necessary under normal conditions. When challenge validations are required, however, the bandwidth cost per challenge is non-trivial. (The validator receives $(r \cdot \ell)$ bits from a given client, and returns $(n \cdot \ell)$ bytes, where r is the number of message blocks (and thus the length of any given bit vector) and n is the size of a message block.) Though a client must validate its challenge-set in its entirety, the client should avoid sending the entire set at once, however. It is advisable to submit each element of the challenge-set to the validator successively, so that the validator can impose rate-limiting on the incoming validation requests to cope with instances of sudden high load.

The ability to gracefully address the issue of Byzantine nodes is itself an anti-Byzantine measure; it has been suggested that when the Pynchon Gate is first deployed, it use PynGP 1.0 until evidence of the existence of Byzantine servers in the system is observed [9]. It is possible that merely having PynGP 2.0 implemented in the software and able to be enabled instantly could serve as a deterrent to a casual attacker whose goal is to deny service to the system, as the only effect such an attacker would have by deploying a Byzantine node would be to increase the network communication and verifier computation costs to a level that we have already deemed acceptable.

7 A Prior Attempt to Improve the Robustness of the Pynchon Gate

In his recent paper [21], Goldberg suggests that detection of Byzantine servers in the Pynchon Gate should be addressed using an **information-theoretic t-private v-Byzantine-robust k-out-of-ℓ PIR** protocol based on Shamir secret sharing [37], such as that proposed by Beimel and Stahl [2]. The paper then presents a performance improvement upon the results of [2], and introduces a two-stage Byzantine recovery procedure for its protocol.

[8] For n challenge-vector sets, the probability of avoiding detection is $\frac{1}{n+1}$, and consequently, the probability of failing to cause a Byzantine failure is $\frac{n}{n+1}$.

7.1 Usability Advantages in t-private k-out-of-ℓ PIR

One advantage to using a t-private k-out-of-ℓ PIR scheme over an $(\ell-1)$-private ℓ-server PIR protocol such as PynGP is that the user need not know which databases are online at the time she makes her request. Indeed, t-private k-out-of-ℓ PIR schemes handle the situation where some servers n (where $n < k$) crash while the request is being made, such that the user is still able to obtain the results to her query in an expedient fashion. From a usability standpoint, this is superior to the existing PynGP, where the user is not permitted to re-request his mail, and encouraged to be patient – undelivered mail will remain in the system for later retrieval.[9]

7.2 Security Degradation with t-private k-out-of-ℓ PIR

Unfortunately, t-private k-out-of-ℓ PIR protocols such as those based on Shamir secret sharing force a trade-off between the number of servers that may collude before the system is compromised, and the number of servers that may be Byzantine without affecting the user's ability to obtain the results of her query. (In a t-private k-out-of-ℓ PIR scheme, any $t+1$ may collude to break the security of the system, as long as those k servers all received a valid query from the user. Thus, the maximum collusion protection such a system can offer is where $t = (k-1)$.) An $(\ell-1)$-private ℓ-server PIR protocol is equivalent to a t-private k-out-of-ℓ PIR scheme, where $t = (\ell-1)$ and $k = \ell$. Since it is the case that when availability guarantees (inversely proportional to k) are increased, the threshold at which *any* query at *any* time may be compromised by colluding servers is diminished in response, the decision must be made as to how critical for a given use case it is that the system respond with a correct answer in the face of sudden node failure. This permits a safe trade-off between availability of service and the security of the system. In the case of a service offering persistent pseudonyms, we consider intermittent and infrequent service interruptions to be the lesser concern.

7.3 An Infeasible Attempt to Rectify this Security Degradation

Goldberg addresses this serious concern by introducing an extension to his t-private v-Byzantine-robust k-out-of-ℓ PIR protocol that provides *computational security* protection for the results of a query, even in the case that *all servers*

[9] The security implications of permitting users to recover from Byzantine failures (and obtain their data, even when some number of responding distributors are known to be Byzantine) are not well studied. It is our intuition that an attacker, particularly one with access to at least two colluding distributors, could manipulate his responses such that some users were able to pass the recovery step, and others unable to, and in this manner the attacker could use the user as an oracle and gain information based on his actions. Therefore, PIR schemes which provide Byzantine *recovery* properties (such as [2] and [21]) should be used with caution if intended to serve as a building-block for an anonymity system.)

collude, by proposing a hybrid privacy protection scheme which relies on the *Paillier additive homomorphic cryptosystem* [30]. This extension gives a PIR protocol with t-private v-Byzantine-robust k-out-of-ℓ information-theoretic protection and ℓ-computational protection. However, as the author states, adding this modification is quite expensive. While the rest of the protocol performs rather efficiently (and better than previous results, under certain configurations), the CPIR hybrid extension suffers the same problem as all other currently existing CPIR solutions. For each n-bit word in the database, the server must perform a modular exponentiation operation. Based on the performance calculations in [21], we do not believe this satisfies the goals of *deployability* and *usability* as set forth in Section 1.0.1 of the Pynchon Gate design paper [34]. The concept of a hybrid information-theoretic/computational PIR scheme is quite promising, should a CPIR scheme be proposed with affordable computation requirements. Until that happens, though, we are forced to discount the ℓ-computational protection extension to the protocol in [21], as it is presently impractical given real-world resource constraints.

These real-world limitations reduce Goldberg's protocol to a simple t-private v-Byzantine-robust k-out-of-ℓ PIR protocol such as that proposed in [2], which allows for the identification of Byzantine servers only at the cost of reduced overall security as compared to PynGP. PynGP 2.0 requires no such compromise in its security.

8 Conclusions

We have reviewed the attack as described in [35], and found that it has significant impact on the deployability and potential success of the Pynchon Gate, as well as other PIR-based nym server systems that do not account for Byzantine servers. A denial or degradation of service attack would be nearly impossible to thwart, and would likely happen soon after the system became popular among users. This vulnerability must not be present in the public system if it is to be expected to achieve and maintain any level of popularity or substantial user-base.

We have presented a simple modification to PynGP 1.0, relying on nothing more than an additional set of operations already performed by the original PynGP 1.0, to enable the detection and identification of Byzantine nodes with sufficient probability that the denial of service attack against the PynGP 1.0 is no longer feasible. This modified protocol, PynGP 2.0, requires no weakening of the original Pynchon Gate security model, and although it increases the bandwidth communication overhead, the bandwidth costs are still reasonable enough to fall within the engineering requirements of the original Pynchon Gate design goals: namely, that the system's bandwidth requirements be inexpensive enough to be reasonable for both users and system operators.

We have examined the prior solution proposed by Goldberg in [21] to address the Byzantine server vulnerability. We show that the trade-offs made in Goldberg's proposal do not satisfy the the security requirements set forth for the

Pynchon Gate in its original design paper, as Goldberg's core protocol weakens the security assumptions significantly compared to the original PynGP.

With the addition of PynGP 2.0, we consider the Pynchon Gate design to be superior to any other high-latency pseudonym service offering strong privacy properties currently proposed in the literature.

9 Future Work

The "cut-and-choose" Byzantine protection in PynGP 2.0 provided by the set of challenge bit-vectors $\{\vec{\eta}\}$ is possibly not as efficient as it could be. The security implications of a modified validator protocol should be explored, perhaps with inspiration from sister disciplines, such as digital cash authenticity verification [5], secure secrecy-preserving auctions [31], or randomized partial checking in mix networks for voting applications [22].

Using the PynGP 2.0 protocol, users can detect when a given server has behaved in a Byzantine fashion. However, they have no irrefutable way of proving this to a third party. We have considered a scenario where the distributors always return a signed response to a PIR query, but there is no reason why a Byzantine server would not simply refuse to sign (or more likely, sign incorrectly) data it wished to later repudiate. Until an adequate solution which enables the user to prove a given flawed response came from a particular server is available, distributor reputation is likely to be dictated by consensus.[10]

The prototype Pynchon Gate system needs to be deployed and performance tested, and user-studies must be performed to evaluate its success from a deployability and usability standpoint.

10 Acknowledgements

We would like to thank Brian Warner and Bryce Wilcox-O'Hearn for their helpful early discussions of the Byzantine Postman Problem; Ian Goldberg for suggesting the potential of polynomial interpolation-based PIR solutions; Orr Dunkelman, Meredith L. Patterson, and Julia Wolf for informative discussions of t-private k-out-of-ℓ PIR schemes; Elena Andreeva, Ian Goldberg and David Molnar for assistance in searching the literature for pertinent work; and David Chaum, Bram Cohen and Meredith L. Patterson for inspirational discussion of PynGP 2.0.

We would also like to especially thank Nikita Borisov, David Chaum, Shar van Egmond, Meredith L. Patterson, and Bram Cohen for their advice and support during the course of this research.

This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the IAP Programme

[10] However, it may be a positive property that signing is not used in this way, as this permits a wider degree of repudiation on the part of the user with regard to her received messages.

P6/26 BCRYPT of the Belgian State (Belgian Science Policy). Len Sassaman's work was supported in part by the EU within the PRIME Project under contract IST-2002-507591.

References

1. Alessandro Acquisti, Roger Dingledine, and Paul Syverson. On the Economics of Anonymity. In Rebecca N. Wright, editor, *Financial Cryptography*. Springer-Verlag, LNCS 2742, 2003.
2. A. Beimel and Y. Stahl. Robust information-theoretic private information retrieval. In S. Cimato C. Galdi G. Persiano, editor, *3rd Conf. on Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 326–341. Springer-Verlag, 2002.
3. Oliver Berthold, Sebastian Clauß, Stefan Köpsell, and Andreas Pfitzmann. Efficiency improvements of the private message service. In Ira S. Moskowitz, editor, *Proceedings of Information Hiding Workshop (IH 2001)*, pages 112–125. Springer-Verlag, LNCS 2137, April 2001.
4. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, 1988.
5. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In S. Goldwasser, editor, *CRYPTO88*, pages 319–327. Springer-Verlag, 1988. Lecture Notes in Computer Science No. 403.
6. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.
7. Benny Chor and Niv Gilboa. Computationally private information retrieval (extended abstract). In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of Computing*, pages 304–313, New York, NY, USA, 1997. ACM Press.
8. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995.
9. Bram Cohen. Personal communication, December 2006.
10. David A. Cooper and Kenneth P. Birman. Preserving privacy in a network of mobile computers. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, May 1995.
11. George Danezis. Statistical disclosure attacks: Traffic confirmation in open environments. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.
12. George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.
13. George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In *Proceedings of 6th Information Hiding Workshop (IH 2004)*, LNCS, Toronto, May 2004.
14. T. Dierks and C. Allen. The TLS Protocol. Request for Comments: 2246, January 1999.

15. Roger Dingledine, Michael J. Freedman, and David Molnar. The Free Haven Project: Distributed anonymous storage service. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*. Springer-Verlag, LNCS 2009, July 2000.
16. Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006)*, Cambridge, UK, June 2006.
17. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
18. Roger Dingledine and Len Sassaman. Attacks on Anonymity Systems: Theory and Practice. In *Black Hat USA 2003 Briefings*, Las Vegas, NV, USA, July 2003.
19. John Douceur. The Sybil Attack. In *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, March 2002.
20. Nick Feamster and Roger Dingledine. Location diversity in anonymity networks. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2004)*, Washington, DC, USA, October 2004.
21. Ian Goldberg. Improving the Robustness of Private Information Retrieval. In *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, May 2007.
22. Markus Jakobsson, Ari Juels, and Ronald L. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *Proceedings of the 11th USENIX Security Symposium*, August 2002.
23. L. Kissner, A. Oprea, M. Reiter, D. Song, and K. Yang. Private keywordbased push and pull with applications to anonymous communication. *Applied Cryptography and Network Security*, 2004.
24. Nick Mathewson. Pynchon Gate Protocol draft specification, September 2004. `http://www.abditum.com/pynchon/`.
25. Nick Mathewson. Underhill: A proposed type 3 nymserver protocol specification, August 2004. `http://www.mixminion.net/nym-spec.txt`.
26. Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, LNCS, May 2004.
27. David Mazières and M. Frans Kaashoek. The Design, Implementation and Operation of an Email Pseudonym Server. In *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS'98)*. ACM Press, November 1998.
28. Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2, July 2003. `http://www.abditum.com/mixmaster-spec.txt`.
29. Steven J. Murdoch. Hot or not: Revealing hidden services by their clock skew. In *Proceedings of CCS 2006*, October 2006.
30. Pascal Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In Jacques Stern, editor, *Advances in Cryptology: EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 1999.
31. David C. Parkes, Michael O. Rabin, Stuart M. Shieber, and C. A. Thorpe. Practical secrecy-preserving, verifiably correct and trustworthy auctions. In *ICEC '06: Proceedings of the 8th International Conference on Electronic Commerce*, pages 70–81, New York, NY, USA, 2006. ACM Press.
32. M. O. Rabin. Digitalized signatures. In R. Lipton and R. De Millo, editors, *Foundations of Secure Computation*, pages 155–166, New York, 1978. Academic Press.

33. Jean-François Raymond. Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 10–29. Springer-Verlag, LNCS 2009, July 2000.
34. Len Sassaman, Bram Cohen, and Nick Mathewson. The Pynchon Gate: A Secure Method of Pseudonymous Mail Retrieval. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2005)*, Arlington, VA, USA, November 2005.
35. Len Sassaman and Bart Preneel. The Byzantine Postman Problem: A Trivial Attack Against PIR-based Nym Servers. Technical Report ESAT-COSIC 2007-001, Katholieke Universiteit Leuven, February 2007.
36. Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In Fabien Petitcolas, editor, *Proceedings of Information Hiding Workshop (IH 2002)*. Springer-Verlag, LNCS 2578, October 2002.
37. A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.
38. Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. Defending anonymous communication against passive logging attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.