Pixy: A Privacy-Increasing Group Creation Scheme

David Mödinger, Nicola Fröhlich and Franz J. Hauck
Institute of Distributed Systems, Ulm University
Email: {david.moedinger, franz.hauck}@uni-ulm.de

***Abstract*—Modern peer-to-peer networks provide a lot of value. However, as the networks handle more and more sensitive data, e.g. in cryptocurrencyies, privacy becomes an issue. Several approaches to provide efficient privacy to network participants rely on group formation with little or no regard to the privacy impact of how groups are created. Group creation is often based on random selection, which can easily be highjacked by attackers. We propose Pixy, an extensible, component-based scheme to increase privacy during group formation stages beyond current approaches. Our scheme provides a two-stage setup for group formation. First, a selection based on personal and network-wide collaboration lists reduces the attack surface for group initiators. Second, a testing phase based on cryptographic puzzles and, for suitable contexts, CAPTCHAs sort out Sybil attackers. We show that this scheme improves the current state of privacy in group-creation processes.**

I. Introduction

Many cryptocurrency systems got popular in the last years. They apply peer-to-peer (P2P) networks for the underlying architecture. Such a P2P network is an easy way to realise a fully distributed, scalable system. Unfortunately, the combination of distribution and sensible financial information creates privacy problems.

One approach to solving privacy on the network level applies group-based privacy mechanisms [7], [17], [38]. The privacy is based on the indistinguishability of the originator of a message from all other network participants, i.e. the anonymity set. Although, if multiple participants of the network collude, they can reduce the effective privacy of this method.

In a group-based network protocol, the current state of the art Sybil detection techniques would be performed during the actual process of the group-based communication protocol. But during this process, privacy could have been violated already. To prevent this, testing needs to be done before any privacy requiring group communication takes places. This creates a novel combination of group-based communication techniques and Sybil detection mechanisms, as the current state of the art relies on problematic assumptions, e.g., on already established trust relationships. Those can be easily gained by the attacker through short preparation times before the attack.

Contribution

We propose Pixy, a novel group creation scheme focusing on privacy for the participants. Pixy increases privacy guarantees for group creation, reducing the required group size for a given desired privacy level, as they are used by von Ahn et al. [7] and others. The scheme provides a flexible component-based approach and well-suited default mechanisms.

Roadmap

The structure of this paper is as follows: Section II discusses existing group creation strategies and Section III will give an overview of the relevant background such as mechanisms we apply and the scenario we operate in. In Section IV we will describe Pixy, our group creation scheme. Lastly in Section V we will provide a first evaluation of Pixy.

II. Related Work

The literature does not have any ready-made solution for group formation under Sybil attacks or collaboration, as Sybil attack detection and prevention have to be customised to the underlying architecture to be effective.

But there are two noteworthy group creation schemes for privacy or security, which we will discuss here. The trustworthy group making algorithm [9] groups nodes based on a partially established trust graph between the nodes. The Secure group agreement protocol [19] deals with a group invitation message from which the group members can be determined and verified.

A. Trustworthy Group Making Algorithm

In 2011 Aikebaier et al. [9] published a novel approach which builds a group of peers based on transitive trust. They apply their previous trustworthiness concept [8]: A peer is considered more trustworthy, the more messages a peer successfully forwards.

Aikebaier et al. develop a broadcast algorithm which relies on the trustworthiness of the individual peers for a formed group. Group formation is carried out via a previously established trust graph and a group initiator peer which calculates the trustworthiness of the other peers from its point of view. The trustworthiness of a node is either direct trustworthiness computes as a ration of several successful interactions to all interactions with such a node, or as transitive trust based on direct trust multiplied by the trust of the intermediate path. For multiple paths, the maximum trust value is chosen. If there have been no previous interactions, direct trust is undefined.

To form a group, Aikebaier et al. [9] start with one source peer which invites the neighbour peers with trustworthiness over a specific threshold. These neighbour peers then introduce their neighbour peers with a trust value over this threshold to the source peer. If a group cannot be formed with the desired threshold, the source peer reduces the threshold and restarts the group formation.

B. Secure Group Agreement Protocol

The approach by Corman et al. [19] describes another group building mechanism in the context of peer-to-peer games over the Internet. In this context, a subset of peers in the network can form a verification group for occurred events of a game. Corman et al. base their approach on a distributed hash table (DHT) over a P2P network, which maps the peers and files into the key spaces using a hash function, as well as an established public key infrastructure.

A group initiator computes an invitation message for the group members and sends it to the peers of the group. Every invited peer answers with an acceptation message. If a response of a peer does not return, the group peers try to forward the invitation to the missing peer. If the answer of the absent peer is still missing, the group formation will be aborted.

An invitation message consists of a signature and hashable contents, including a group id, timestamp, purpose and an incrementing member number. Peers are chosen randomly based on the random oracle property of the hash value of the content, to prevent precomputation or groups and produce well-randomized group composition.

C. Applicability

The two group formation protocols serve different purposes in the field of peer-to-peer networks and are therefore not well applicable to the given problem of increasing privacy. The one by Corman et al. [19] represents a verified group invitation message, and the other by Aikebaier et al. [9] is based on already established trust relations between peers and contains an initiator peer which starts the group formation with the most trustworthy neighbour peers.

In our situation, the verified invitation message could be used to form a group, but would not help with our problem of Sybil attacks and collaboration. The group formation by Aikebaier would support the protection of the group forming against unfamiliar nodes, but if one Sybil peer is in the trust range, it can invite other Sybil nodes and the intrusion would never be detected. So these approaches are not suitable for our problem.

III. Background

In this section, we will discuss the relevant background information. First, we will elaborate on our assumptions and scenario, setting important boundary conditions for viable mechanisms. Second, we will discuss some possible mechanisms and classifications of mechanisms we will apply in the scheme.

A. Scenario and Assumptions

We attempt to create a group of nodes in a public P2P network, which can be joined by arbitrary participants. The P2P network is established over the Internet without a wireless connection. The connected devices are conventional computers without relying on a camera or microphone. For the purpose of this paper, we consider group sizes of at least 3 and at most 20 nodes.

We assume nodes will act rational and will not deny participation, as non-participation can be handled while handling network reliability concerns, which would needlessly complicate explanations. Communication within the network is secured and by usual means such as TLS. We further assume that attackers focus on utilizing the protocol and not on implementation errors or network properties, but no previously established trust network is required.

B. Sybil Prevention and Detection Mechanisms

When an attacker claims to have multiple identities to get an advantage, that's called a Sybil attack [23]. Sybil attacks typically occur in applications like online voting and reputation systems [28], [32]. P2P networks with no certificate authority are generally vulnerable to Sybil attacks [23].

Categories of Detection Mechanisms: We provide an overview over typical mechanisms to detect Sybils based on previous survey articles [11], [28], [32].[1]

- **Trusted Certification** makes a certification authority responsible for preventing Sybils [23].
- **Resource Testing** assumes limited hardware for participants and little variance in resources. If all participants need to prove the amount of resources they can expend, e.g., by a proof of work scheme, a Sybil attacker can only provide a fraction of the resources compared to a non-Sybil node [14], [33], [45].
- **Recurring Cost** reduces the complexity of the resource check but repeats it over time. An attacker, therefore, has to provision the required resources over longer amounts of time, renting additional resources in the beginning, is therefore uneconomical [14].
- **Incentive-based Detection** provides protocol level incentives, e.g., increased trust or virtual coins, to a node informing of collusion by cryptographic proofs of messages [36].
- **Social Graph** requires an already established social trust graph, which is evaluated for connections between the area of honest nodes and dishonest node [21], [53], [54].

Most schemes try to discourage the attacker by increasing the cost of an attack over the profit of the attack. While the Sybil attack cannot be prevented in full, the probability can be reduced. The effectiveness of Sybil detection or prevention technique depends on the use of a suitable technique for the current architecture [28], [32].

We exclude Privilege Attenuation, which is not suited for P2P networks, Location and Position Verification and Received Signal Strength Indication-based schemes, as they do not work for wired internet connections. Lastly, we also do not consider Random Key Pre-Distribution as they require a static, i.e., non-dynamic, setup phase. From the remaining techniques we will restrict ourselfes to resource testing, as

[1]Note that we could not determine clear first authorship between the very similar [32] and [11].

trusted certification either requires a trusted third party or another form of consensus. We do not consider recurring costs for now, as they could be implemented as a simple extension to our scheme. We also exclude social graph mechanisms as they require a pre-established trust graph, but they provide promising enhancements for pixy in long-running networks.

Resource Testing: Resource testing can be realised for Sybil detection and prevention by well known computational puzzles [22], [25], [37]. In the literature, there are different designations for such puzzles and these often confuse.

A puzzle is used between a prover and a verifier. In general, a puzzle consists of three phases: Generation, solving and verification of the puzzle. Puzzles can be interactive or non-interactive and have a symmetric or asymmetric workload. The puzzles can further be divided by their application, which resources are used, and implementation of the verification process [10].

The considered application types are pricing puzzles, delaying puzzles, timing puzzles and AI hard puzzles, sometimes called CAPTCHAs. Delaying puzzles require a predefined amount of time and are sometimes called timelock puzzles [42] while timing puzzles have no predetermined time at construction. Pricing puzzles have a known lower-bound on resources and are used to increase the cost of an otherwise cheap operation [25].

Puzzles can further be separated by resources, most often: CPU, memory, bandwidth, network latency or ordering and human attention. Applicability depends on the use-case and setting. Lastly, there are further properties of puzzles as, the most important are the uniqueness of the challenge, unforgeability of solutions and non-parallelisability of the computation.

For this work, we evaluated several puzzles based on these properties, shown in Table I.

C. CAPTCHAs

CAPTCHAs are mechanisms to tell humans and machines apart [48], [49]. As they prevent automation, they are well suited to prevent automation-based Sybil attacks [31], e.g., by botnets. It is important to note, that they can be only applied in contexts where automation is not an important goal. We argue that privacy is mostly relevant for humans, so having humans verify the initial step of a privacy-enhancing group formation is acceptable and automation is not desired.

CAPTCHAs are usually based on text, image, audio, video or games, e.g., jigsaws. The strength of each test depends on the development in research of AI and the realisation of the technique. The most commonly used CAPTCHAs are text and image-based variants. They are easy to implement and realise [30], [39], [44], [49].

Classical approaches have been targeted by machine learning research, which produces success rates of up to 70% [57] contributing to an arms race of CAPTCHAs and CAPTCHA solving. Research also produced new approaches to fight machine learning and pattern recognition using video-based gesture detection [47] and comparison-based identification [51]. These and mechanisms on adversarial examples [40], [43], [55] are promising but not developed enough to be used yet. Googles most recent version of reCAPTCHA seems to be considered secure for now.

IV. Pixy: Group Creation Scheme

This section presents Pixy, our privacy-increasing group creation scheme. First, we will give an overview of the scheme and then discuss the two phases of Pixy in detail. Lastly, we will show how decision making after the last phase works.

A. Overview

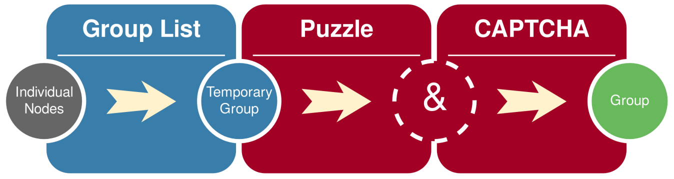

Fig. 1. Overview of the architecture of Pixy.

Pixy consists of two phases and is structured as a modular system. It is divided into two phases. The first phase is used to construct a temporary group based on group lists of untrusted groups, e.g., IP subnets. The second phase consists of aggregatable tests for participants to prove their independence. In figure 1 the process of the scheme is illustrated with its two phases and its three elements in total.

The elements are building blocks and can be exchanged easily to support fast assimilation to new developments or different network architectures. This is important to create a future proof scheme.

B. Phase 1: IP-based Gatekeeping

The result of the first phase is a temporary group with members that are not obviously collaborating. As we are dealing with internet messages, there is only little information we can use and will primarily use IP addresses similar to other systems such as Tarzan and others [26], [32]. We assume the protocol can determine which peer is in which subnet, which is warranted due to the use of general internet-wide IP addresses. While we can not assume a pre-established trust graph, some participants may build a personal, hidden trust graph over time, augmenting their personal decision making.

We use a group list for our scheme, similar to blacklists, we only allow one participant per group. There should be a global group list for a given network, containing well known collaborating groups, e.g., google. The collection of groups depends on the concrete threat model of the given network, e.g., a very paranoid network might restrict groups to participants from different continents by geo IP grouping. One risk factor for group lists are VPNs and Tor like services, so depending on the attacker model, a network should include these or risk users including them in their personal group lists anyways. To allow for evolving group lists, distributed

	Time-lock [12], [29], [34], [42]	Non-parallelisable [27], [46]	Memory-bound [6], [24]	PoSW [18], [35]	Delay [13], [41], [52]
Resource	CPU	CPU	Memory	CPU	CPU
Simultaneous	+	+	+	+	+
Asymmetric	+	+	+	+	+
Implicit	-	±	-	-	-
Unforgeable	+	+	+	+	+
Unique Puzzle	+	+	+	+	+
Unique Solution	+	+	+	-	+
Sequential	+	±	+	+	-

TABLE I

OVERVIEW OVER THE CHARACTERISTICS OF THE INTERACTIVE PUZZLES: TIME-LOCK PUZZLE, NON-PARALLELISABLE PUZZLE, MEMORY-BOUND FUNCTION, PROOF OF SEQUENTIAL WORK (POSW) AND DELAY FUNCTION. + SHOWS THAT A PUZZLE SUPPORTS THE PROPERTY, WHILE - SIGNALS NO SUPPORT. ± SHOWS THAT SUPPORT IS NON-HOMOGENOUS OVER EVALUATED VARIANTS.

append-only ledgers, e.g., blockchains, might provide suitable background stability while making the lists future proof.

The check of IP addresses takes place before the actual group formation. Peers might already cancel group formation of invalid groups according to their group list If no peer cancels at this stage, the participants create connections to all participants and continue with phase 2 as a temporary group.

C. Phase 2: Testing

Once the temporary group is formed, participants start the testing phase. For this phase, all participants apply known Sybil detection mechanisms to test if any node in the groups are Sybils.

The second phase is modular, so it can be constructed with different puzzles. For a sensible default application, we propose two independent core modules: A delay puzzle and a CAPTCHA.

Delaying Puzzle: The first presented module is a delaying puzzle. We recommend the delaying puzzle due to the properties we outlined in Table I. As for this step, a puzzle is needed that can test the hardware of all nodes simultaneously by one verifier node. For this, the workload on the verifier side has to be low, i.e. asymmetric workloads are needed, otherwise, it can lead to a denial of service (DoS) attack, and the puzzle set per node must be unique so that no solutions can be reused.

Delay puzzles fulfil all the required criteria. As different CPUs produce a wide spread in cost, we require a memory-bound puzzle to have as little variance in devices as possible [10]. The puzzle verification should be public to reduce the work verifier nodes which perform the current testing round. Although, in this testing phase every node in the group will be a verifier node and will set the puzzle once.

The verifier node creates a unique puzzle for every other node in the group to prevent the reuse of solutions and checks the returned responses of the nodes. The returning time of the response will be recorded as well. The puzzles are distributed simultaneously to all nodes in the group.

We applied the puzzle by Abadi et al. [6], but Dwork et al. [24] explored the same approach. They suggested a different random function, for simplicity, we restrict ourselves to the initial approach.

The concept of Abadi et al. for a memory-bound function is based on a tree with depth k. For now, k be an integer. The tree is constructed from a random leaf value x_0 to the root value x_k. This construction ensures a sufficient size of the tree, around the size of $(k+1)(k+2)/2$. The depth k of the tree has to be much smaller than 2^n, with n an integer. The authors suggest that k should fulfil $k < 2^{n-5}$. The tree is computed by the function $F : [0, 2^n - 1] \rightarrow [0, 2^n - 1]$. F is a random function with no permutation.

The equation

$$x_{i+1} = F(x_i) \oplus i$$

with the index $i \in (0, \ldots, k-1)$ calculates the nodes in the path from the leaf x_0 to the root value x_k. The calculation of the tree is a chain of repetitive applications of the function F.

The verifier has to choose a random leaf value x_0 and construct the tree path from x_0 to x_k. Over the path from x_0 to x_k, a checksum is built and sent to the prover with the root value x_k. The verifier adds the checksum of the path to x_0 and the prover has to compute x_0.

To find the desired element, the prover computes the inverse function F^{-1} of F. The function F is chose in such a way, that computing F^{-1} is less efficient than a memory access. So the most efficient solution is to construct a memory table of size 2^n for F and do reverse lookups. A depth-first search computation of the solution requires unpredictable random access to distributed locations of memory. From that, the prover constructs possible pre-image chains, which are compared to the given checksum by the verifier. If a solution is found, the verifier just needs to check if the solution matches x_0.

CPU-intensive algorithms for this puzzle solution exist, but for a good choice of the function F and the parameters n and k, these approaches do not bring any advantages. These solutions would need a longer time span to solve the puzzle than to use the memory-bound one.

Abadi et al. also give relative values for the choice of parameters. The researchers of [6] suppose that the work f for the function $F()$ should be $(r/8) \leqslant f \leqslant r$ and r is the work for a memory read to prevent a fast inversion and computation. The depth k has to be chosen carefully with the assumptions that $k < 2^{n-5}$ to prevent cache lines, $k >> 4 \cdot (f/r)$ to force a high work ratio on the prover, $k \geqslant (2^{(n/2)+1} \cdot \sqrt{f/r} \cdot \sqrt{c})$ with c as a cost factor for CPU intensive solutions to prevent a CPU-intensive search and

$k \geqslant (2^{(n/2)+1} \cdot \sqrt{1/p} \cdot \sqrt{(f+w)/r})$ so that the table can be built efficiently.

The integer p represents the number of such combined problems. A problem is considered as finding a x_0 from x_k. The costs of memory reading and writing are considered equal ($r = w$). The parameters c, r and w are integers. Abadi et al. [6] also provided an example for F and tested common cryptographic hash functions. They show that F can be constructed as a random function generated through a master function by $F(x) = MF(x) \oplus j = G(t, x)$. G consists of two predistributed random tables of sizie $2^{\frac{n}{2}}$ and discards the 16 most- and least significant bits of a multiplication of table entries. The resulting function is $F(x) = F(a_0|a_1)\,\text{middle-bits}(t_0[a_o] \cdot t_1[a_1])$.

The puzzle can be solved a little bit faster with the help of better hardware, but not that much faster than it would be with CPU-bound puzzles. For the uniqueness of the puzzle, the function $F(x) = MF(x) \oplus j$ and x_0 has to be varied for every node. [6]

A closer mathematical and practical examination of the concept of Abadi et al. [6] in the direction of a concrete realisation would exceed the scope of this paper.

CAPTCHA: In the second module of phase 2, the nodes are tested for human attention. In this puzzle, the number of verifier nodes can be reduced and thus the number of testing rounds. This step should, by design, not be automatable by the attacker or regular participants. This makes CAPTCHAs only acceptable in certain contexts, but we argue that privacy is a human-centric concept, which makes this a good compromise [31].

As discussed in Section III, thus far, we lack real-world implementations of newer and open source CAPTCHA approaches [30], [39], [44], [49]. When reimplementing and evaluating the approaches is not possible, we recommend the proprietary reCAPTCHA by Google [1], [2], [50] to provide at least a concrete realisation of a CAPTCHA.

Though, attacks on CAPTCHAs continue to improve over time and do not stop for reCAPTCHA [15], [57]. reCAPTCHA will block some bots, and it requires the effort to develop a machine learning algorithm of the attacker to overcome this puzzle challenge. The benefit of using a CAPTCHA for a scheme like Pixy prevent many kinds of attacks but require frequent updating and flexibility on the CAPTCHA part.

Depending on the kind of CAPTCHA applied, the result can either be a binary decision or a 'belief of humanness', that can be used with a custom minimum threshold to accept the participant.

D. Decision Making

The testing of nodes in phase two requires an aggregation of scores and decision making. We recommend that each node be verifier at least once to reduce the probability of manipulation. This results in up to N, the number of total participants, testing rounds which can be different for all modules. For reduced rounds, the group should use a secure random scheme, e.g., as used by dissent [20], to select nodes. For the discussion, we will assume that N testing rounds are performed.

The approach of evaluating the puzzle results is inspired by the approach of Cárdenas-Haro and Konjevod [16]. In a testing round of the puzzle, the current verifier node sets the puzzles for every node simultaneously and uniquely. Thus, every prover node starts the computation of the solution for the puzzle at the same time and should finish at the same time. After a delaying puzzle, the results of all provers should return at the same time.

To measure the time, the time is divided into time slots of the length t. The real length t of the time slots depends on the time p for the puzzle solution as well as on the Internet delay d. The length t of the time slot should be smaller than the needed time for the puzzle computation p ($t \leq p$). This requirement prevents that a prover receives and answers the puzzle in the same time slot, therefore it would not be sufficient to measure the elapsed time. Further, it should hold that $d < t < p$, i.e., the timeslot is longer than the expected network delay. The responses to the challenges should return to the verifier in the same time slot.

The verifier creates a matrix of all response timeslots. Two peers, which are not distinct, cannot respond at the same time as the solutions have to be calculated sequentially by a Sybil node. If two responses from two peers return in the same time slot, then these two peers have a high probability to be two distinct hardware devices. If the responses of two peers never return at the same time for all testing rounds, then it is very likely these two peers are Sybil nodes.

Due to network delay, a response might return a timeslot late. Therefore neighbouring timeslots to the expected slot have reduced penalty for detection. The matrix values can be aggregated over all tests. Low values correspond to a high probability of Sybils being present. Suitable thresholds need careful evaluation of acceptable risk for the network though. All results of all puzzles need to be above the desired thresholds for each puzzle for the group to be viable. If this condition is met, the group continues as a fully-formed group with the desired protocol.

V. Security Evaluation

The literature on Sybil attack prevention and CAPTCHAs did not provide evidence on the effectiveness of the algorithms. So a full evaluation of Pixy is yet to be done. Instead, we will focus on a theoretical exploration of the attack space. The two core goals to fulfil are:

- An attacker must not gain an advantage when Pixy is used.
- Multiple attacks must be impractical or noticeably harder by using Pixy.

Goal one is required to provide a strict improvement in privacy, while the results for goal two would indicate the effectiveness of Pixy. To reach a conclusion on goal one, we evaluated possibilities to subvert the phases of our scheme.

We consider various common attackers with varying capabilities. A **single attacker node** with moderate access to resources in money and IP addresses, i.e., they can access free VPN services. Their goal is to subvert the group with little resources and preferably no cost. A **botnet**, which has many fully automated nodes with IP addresses which are well spread throughout most subnets. The goal of a botnet is to subvert the group while being economical with little expense but large scale hardware resources. A **large scale** corporation or agency of a nation-state with quite a lot of resources. The restrictions of entities of this scale are mostly political, e.g., they do not want to be noticed.

A. Phase 1: IP Addresses

The first phase is primarily controlled by the node initiating the group creation, an attacker should therefore choose to initiate. As Pixy has no enforcement of random selection, an attacker might freely select cooperating nodes to participate, as long as they respect the global group list. This leads to the following attack vectors:

1) While **IP spoofing**, i.e., pretending to have a different IP address from the ones you actually control, allows to pass the first phase, usual applications of IP spoofing do not allow the attacker to control the address for subsequent phases [3]. Therefore the multi-phase setup prevents many forms of IP spoofing. IP spoofing by powerful local attackers, i.e., that can reroute traffic successfully, will still succeed.
2) Having **IP address distributed** over many unrelated subnets can be achieved with many different VPNs or comes naturally for botnets. As argued before, known VPNs should be represented in a sensible way in the group list to prevent cheap and inconspicuous attacks via VPNs.

Attack 2 is naturally applied for our botnet attacker, while the single attacker will have trouble depending on the setup of the group list, increasing probability of prevention. The large scale attacker can, depending on positioning, apply attack 1 without being detected, but on a large scale, the activities increase detection risk significantly.

B. Phase 2: Delay Puzzle

In phase 2 of the group creation scheme and the first step for the temporary group, the participating nodes in the group are tested on their available hardware. To overcome this step, additional hardware is required, which is naturally available to botnets and large scale attackers and only slightly increases the cost to them. Contrast to this single attackers will be unable to pass this phase with high probability. Possible circumventions like short term hardware rentals through cloud services introduce additional latency which increases the risk of detection significantly.

C. Phase 2: CAPTCHA

The last step in the scheme involves a test which requires human attention. A solution for this problem of the attacker are underground CAPTCHA solving services [39], [56]. These are services on the web which offer to solve the provided CAPTCHA by optical character recognition software or by human workers at around 1$ to 2$ per 1000 CAPTCHAs [4], [5].

Human attention for a single attacker will be natural to achieve on a small scale, but preventive in large scale systems. While it is available for large scale attackers, it is either a huge factor in cost, if done by themselves or increases the risk of detection significantly. For botnets, at last, the CAPTCHA solving services are the most viable route but increases the cost of operation.

D. Attack Conclusion

All presented steps are restrictive in nature, as they restrict previously available routes. This limits the possibility of gains for an attacker by using Pixy, compared to other current systems.

While we showed that all phases can be broken, all strategies either increase cost, prevent certain kinds of attacks or increase risk of detection for large scale entities, especially through the combination of the different phases. This layered security shows our original goal of making attacks impractical or noticeably harder.

VI. Conclusion

We developed Pixy, a flexible group creation scheme to prevent Sybil nodes in peer groups. Pixy provides increased protection from various common attacks by applying a two-phase scheme. Phase one creates a temporary group based on group lists and personal trust information to exclude Sybils and known collaborators. Phase two tests all participants of the temporary group using a memory timelock puzzle and a CAPTCHA. The results are aggregated per participant allowing every user to come to a satisfying conclusion on their own.

Our first evaluation shows that, while the system is still susceptible to powerful attackers, many common attack patterns are not viable when using Pixy. Therefore pixy provides a clear advantage over the current state of the art.

The design of Pixy is modular so additional puzzles can be included or existing ones can be removed, e.g., the CAPTCHA if automation is required. This allows for continuous further improvements based on recent research.

References

[1] Are you a robot? introducing “no captcha recaptcha”, https://security.googleblog.com/2014/12/are-you-robot-introducing-no-captcha.html

[2] Introducing recaptcha v3, https://security.googleblog.com/2018/10/introducing-recaptcha-v3-new-way-to.html

[3] Ip spoofing: An introduction, https://www.symantec.com/connect/articles/ip-spoofing-introduction

[4] Top 10 captcha solving services compared, https://www.prowebscraper.com/blog/top-10-captcha-solving-services-compared/

[5] Website 2captcha, https://2captcha.com/

[6] Abadi, M., Burrows, M., Manasse, M., Wobber, T.: Moderately hard, memory-bound functions. ACM Transactions on Internet Technology (TOIT) **5**(2), 299–327 (2005)

[7] von Ahn, L., Bortz, A., Hopper, N.J.: K-anonymous message transmission. In: Proc. of the 10th ACM Conf. on Comp. and Comm. Sec. (CCS). pp. 122–130. ACM, New York, NY, USA (2003)

[8] Aikebaier, A., Enokido, T., Takizawa, M.: Trustworthiness among peer processes in distributed agreement protocol. In: 2010 24th IEEE Int. Conf. on Advanced Information Networking and Applications. pp. 565–572. IEEE (2010)

[9] Aikebaier, A., Enokido, T., Takizawa, M.: Trustworthy group making algorithm in distributed systems. Human-centric Computing and Information Sciences **1**(1), 6 (11 2011)

[10] Ali, I.M., Caprolu, M., Di Pietro, R.: Foundations, properties, and security applications of puzzles: A survey. arXiv preprint arXiv:1904.10164 (2019)

[11] Balachandran, N., Sanyal, S.: A review of techniques to mitigate sybil attacks. arXiv preprint arXiv:1207.2617 (2012)

[12] Bitansky, N., Goldwasser, S., Jain, A., Paneth, O., Vaikuntanathan, V., Waters, B.: Time-lock puzzles from randomized encodings. Association for Computing Machinery (2016)

[13] Boneh, D., Bonneau, J., Bünz, B., Fisch, B.: Verifiable delay functions. In: Annual International Cryptology Conference. pp. 757–788. Springer (2018)

[14] Borisov, N.: Computational puzzles as sybil defenses. In: Sixth IEEE Int. Conf. on Peer-to-Peer Computing (P2P'06). pp. 171–176. IEEE (2006)

[15] Brown, S.S., DiBari, N., Bhatia, S.: I am 'totally' human: Bypassing the recaptcha. In: 2017 13th Int. Conf. on Signal-Image Technology & Internet-Based Systems (SITIS). pp. 9–12. IEEE (2017)

[16] Cárdenas-Haro, J.A., Konjevod, G.: Detecting sybil nodes in static and dynamic networks. In: OTM Confederated Int. Conf.s "On the Move to Meaningful Internet Systems". pp. 894–917. Springer (2010)

[17] Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of cryptology **1**(1), 65–75 (1988)

[18] Cohen, B., Pietrzak, K.: Simple proofs of sequential work. In: Annual Int. Conf. on the Theory and Applications of Cryptographic Techniques. pp. 451–467. Springer (2018)

[19] Corman, A.B., Schachte, P., Teague, V.: A secure group agreement (sga) protocol for peer-to-peer applications. In: 21st Int. Conf. on Advanced Information Networking and Applications Workshops (AINAW'07). vol. 1, pp. 24–29. IEEE (2007)

[20] Corrigan-Gibbs, H., Ford, B.: Dissent: Accountable anonymous group messaging. In: Proc of the 17th ACM Conf. on Comp. and Comm. Sec. (CCS). pp. 340–350. ACM, New York, NY, USA (2010)

[21] Danezis, G., Mittal, P.: Sybilinfer: Detecting sybil nodes using social networks. In: NDSS. pp. 1–15. San Diego, CA (2009)

[22] Dean, D., Stubblefield, A.: Using client puzzles to protect tls. In: Proceedings of the 10th conference on USENIX Security Symposium-Volume 10. p. 1. USENIX Association (2001)

[23] Douceur, J.R.: The sybil attack. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) International workshop on peer-to-peer systems. pp. 251–260. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)

[24] Dwork, C., Goldberg, A., Naor, M.: On memory-bound functions for fighting spam. In: Annual International Cryptology Conference. pp. 426–444. Springer (2003)

[25] Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Annual International Cryptology Conference. pp. 139–147. Springer (1992)

[26] Freedman, M.J., Morris, R.: Tarzan: A peer-to-peer anonymizing network layer. In: Proceedings of the 9th ACM conference on Computer and communications security. pp. 193–206. ACM (2002)

[27] Jerschow, Y.I., Mauve, M.: Non-parallelizable and non-interactive client puzzles from modular square roots. In: 2011 Sixth Int. Conf. on Availability, Reliability and Security. pp. 135–142. IEEE (2011)

[28] John, R., Cherian, J.P., Kizhakkethottam, J.J.: A survey of techniques to prevent sybil attacks. In: 2015 Int. Conf. on Soft-Computing and Networks Security (ICSNS). pp. 1–6. IEEE (2015)

[29] Karame, G.O., Čapkun, S.: Low-cost client puzzles based on modular exponentiation. In: European Symposium on Research in Computer Security. pp. 679–697. Springer (2010)

[30] Kaur, K., Behal, S.: Captcha and its techniques: a review. International Journal of Computer Science and Information Technologies **5**(5), 6341–6344 (2014)

[31] Kopp, H., Kargl, F., Bösch, C., Peter, A.: umine: A blockchain based on human miners. In: Information and Communications Security. pp. 20–38. Springer International Publishing, Cham (2018)

[32] Levine, B.N., Shields, C., Margolin, N.B.: A survey of solutions to the sybil attack. University of Massachusetts Amherst, Amherst, MA **7**, 224 (2006)

[33] Li, F., Mittal, P., Caesar, M., Borisov, N.: Sybilcontrol: Practical sybil defense with computational puzzles. In: Proceedings of the seventh ACM workshop on Scalable trusted computing. pp. 67–78. ACM (2012)

[34] Mahmoody, M., Moran, T., Vadhan, S.: Time-lock puzzles in the random oracle model. In: Annual Cryptology Conference. pp. 39–50. Springer (2011)

[35] Mahmoody, M., Moran, T., Vadhan, S.: Publicly verifiable proofs of sequential work. In: Proceedings of the 4th Conference on Innovations in Theoretical Computer Science. pp. 373–388. ITCS '13, ACM, New York, NY, USA (2013)

[36] Margolin, N.B., Levine, B.N.: Informant: Detecting sybils using incentives. In: Int. Conf. on Financial Cryptography and Data Security. pp. 192–207. Springer (2007)

[37] Merkle, R.C.: Secure communications over insecure channels. Communications of the ACM **21**(4), 294–299 (1978)

[38] Mödinger, D., Kopp, H., Kargl, F., Hauck, F.J.: Towards enhanced network privacy for blockchains. In: Proc. of the 38th IEEE Int. Conf. on Distributed Computing Systems (ICDCS) (2018)

[39] Moradi, M., Keyvanpour, M.: Captcha and its alternatives: A review. Security and Communication Networks **8**(12), 2135–2156 (2015)

[40] Osadchy, M., Hernandez-Castro, J., Gibson, S., Dunkelman, O., Pérez-Cabo, D.: No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation. IEEE Transactions on Information Forensics and Security **12**(11), 2640–2653 (2017)

[41] Pietrzak, K.Z.: Simple verifiable delay functions. In: 10th Innovations in Theoretical Computer Science Conference. vol. 124 (2019)

[42] Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Massachusetts Institute of Technology (1996)

[43] Shi, C., Xu, X., Ji, S., Bu, K., Chen, J., Beyah, R., Wang, T.: Adversarial CAPTCHAs. arXiv e-prints pp. 1 –16 (1 2019)

[44] Singh, V.P., Pal, P.: Survey of different types of captcha. International Journal of Computer Science and Information Technologies **5**(2), 2242–2245 (2014)

[45] Tegeler, F., Fu, X.: Sybilconf: computational puzzles for confining sybil attacks. In: 2010 INFOCOM IEEE Conference on Computer Communications Workshops. pp. 1–2. IEEE (2010)

[46] Tritilanunt, S., Boyd, C., Foo, E., Nieto, J.M.G.: Toward non-parallelizable client puzzles. In: Int. Conf. on Cryptology and Network Security. pp. 247–264. Springer (2007)

[47] Uzun, E., Chung, S.P.H., Essa, I., Lee, W.: rtcaptcha: A real-time captcha based liveness detection system. In: NDSS (2018)

[48] Von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: Using hard ai problems for security. In: Int. Conf. on the Theory and Applications of Cryptographic Techniques. pp. 294–311. Springer (2003)

[49] Von Ahn, L., Blum, M., Langford, J.: Telling humans and computers apart automatically. Communications of the ACM **47**(2), 56–60 (2004)

[50] Von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M.: recaptcha: Human-based character recognition via web security measures. Science **321**(5895), 1465–1468 (2008)

[51] Wang, H., Zheng, F., Chen, Z., Lu, Y., Gao, J., Wei, R.: A captcha design based on visual reasoning. In: 2018 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP). pp. 1967–1971. IEEE (2018)

[52] Wesolowski, B.: Efficient verifiable delay functions. In: Annual Int. Conf. on the Theory and Applications of Cryptographic Techniques. pp. 379–407. Springer (2019)

[53] Yu, H., Gibbons, P.B., Kaminsky, M., Xiao, F.: Sybillimit: A near-optimal social network defense against sybil attacks. In: 2008 IEEE Symposium on Security and Privacy (sp 2008). pp. 3–17. IEEE (2008)

[54] Yu, H., Kaminsky, M., Gibbons, P.B., Flaxman, A.D.: Sybilguard: Defending against sybil attacks via social networks. IEEE/ACM Transactions on networking **16**(3), 576–589 (2008)

[55] Zhang, Y., Gao, H., Pei, G., Kang, S., Zhou, X.: Effect of adversarial examples on the robustness of captcha. In: 2018 Int. Conf. on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). pp. 1–109. IEEE (2018)

[56] Zhao, B., Weng, H., Ji, S., Chen, J., Wang, T., He, Q., Beyah, R.: Towards evaluating the security of real-world deployed image captchas. In: Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security. pp. 85–96. ACM (2018)

[57] Zhou, Y., Yang, Z., Wang, C., Boutell, M.: Breaking google recaptcha v2. Journal of Computing Sciences in Colleges **34**(1), 126–136 (2018)