P3CA: Private Anomaly Detection Across ISP Networks

Shishir Nagaraja[1], Virajith Jalaparti[2], Matthew Caesar[2], and Nikita Borisov[2]

[1] IIIT Delhi {nagaraja@iiitd.ac.in}
[2] University of Illinois at Urbana-Champaign
{jalapar1,caesar,nikita}@illinois.edu

Abstract. Detection of malicious traffic in the Internet would be much easier if ISP networks shared their traffic traces. Unfortunately, state-of-the-art anomaly detection algorithms require detailed traffic information which is considered extremely private by operators. To address this, we propose an algorithm that allows ISPs to cooperatively detect anomalies without requiring them to reveal private traffic information. We leverage secure multiparty computation to design a privacy-preserving variant of *principal component analysis* (PCA) that limits information propagation across domains. PCA is a well-proven technique for isolating anomalies on network traffic and we target a design that retains its scalability and accuracy. To validate our approach, we evaluate an implementation of our design against traces from the Abilene Internet2 IP backbone network as well as synthetic traces, show that it performs *efficiently* to support an online anomaly detection system and and conclude that privacy-preserving anomaly detection shows promise as a key element of a wider network anomaly detection framework. In the presence of increasingly serious threats from modern networked malware, our work provides a first step towards enabling larger-scale cooperation across ISPs in the presence of privacy concerns.

1 Introduction

A serious threat to Internet users is the increasingly advanced set of attacks employed by malware to remotely compromise their resources. Compromised machines are used to propagate spam, worms and viruses and participate in DoS attacks. The tremendous scale and complexity of network traffic makes detection of malicious behavior in network traffic an extremely hard problem.

The seriousness of this problem has led network operators to pursue *in-network* solutions to detect and localize malicious traffic. Modern ISPs run monitoring systems to localize malicious traffic, and some offer "scrubbing" services to customers to remove malicious traffic before delivery to the customer [31]. While

This research was supported in part by the National Science Foundation grants: CNS 08–31488 and CNS 10–53781 and the IBM X10 innovation award.

several approaches have been proposed for detecting malicious traffic, the use of *principal component analysis* (PCA) stands out. An array of techniques based on PCA have been recently proposed [17, 25, 27] to detect statistical anomalies in volume or other characteristics of traffic flowing across networks. These schemes rely on monitoring protocols employed at routers to sample traffic (e.g., NetFlow [8] and SNMP counters), aggregate these observations across routers and perform anomaly detection on the aggregate. PCA enables scaling to large datasets by reducing the dimensionality of the traffic and has been shown in the literature to perform well on a variety of workloads and topologies to detect malicious traffic, performance problems, and other forms of outages and hard-to-detect failures.

Performance of these techniques improves with increasing number of vantage (monitoring) points. In addition to providing visibility into a larger number of inter-host paths, additional vantage points increase the likelihood that a given malicious traffic flow is "exposed" due to different statistical mixes of traffic appearing on each observed link. For example, it has been shown that if neighboring ISPs were to cooperate by sharing traffic measurements, anomaly detection could be done with much higher accuracy and anomalies that cannot be detected by each of the ISPs individually could be detected [30]. However, anomaly detection today is unfortunately constrained to operate within a single ISP network as ISPs are highly reluctant to reveal the topology and traffic information necessary for these algorithms to run since they are extremely confidential business information.

Contributions: In our work, we leverage work in secure multiparty computation (SMC) and propose *Privacy-Preserving PCA* (P3CA), a mechanism which supports cooperation among ISPs, allowing them to perform anomaly detection jointly with other ISPs without requiring them to reveal their private information. P3CA retains the desirable properties of PCA, including its accuracy and robustness. One challenge with using SMC-based approaches is scalability, as we target designing a system that can handle collaborations across the large workloads of several core ISP networks (millions of flows, hundreds of routers, tens of collaborating ISPs). To address this, we develop efficient algorithms that scale *linearly* with the number of observations per ISP. We also support incremental anomaly detection to speed up processing by updating the previously computed principal components when new data is obtained. Unlike previous work on preventing information leakage in data mining algorithms [13, 23], we target algorithms in the context of anomaly detection in large-scale networks. In addition, unlike some schemes [13], P3CA *does not publish the dominant principal components* (i.e., their plain text values) allowing privacy of the network traffic and topology to be retained. At the end of P3CA, each of the participating ISP finds the anomalies in its own network and these are not revealed to the others. Our evaluation results show that P3CA (and its incremental version) perform quite efficiently at the scale of large networks. We note that P3CA extends PCA for multiple ISPs and thus, like PCA, can be used only to find the anomalies in a network and not the end hosts responsible for these. Further mechanisms (e.g., [28]) are required to perform root cause analysis and are out of the scope of this paper.

2 System Overview

The Internet is made up of a set of *Internet Service Providers* (ISPs) connected by peering links. Each ISP is a network owned and operated by a single administrative entity (e.g., a campus/enterprise network). To discover routes across ISPs, the Border Gateway Protocol (BGP) [3] is run across peering links. BGP routing advertisements carry information such as the reachable destination prefix and do not reveal the details of the ISP's internals such as topology, traffic (for e.g., set of communicating IP addresses, the load on the ISP's links) as they are considered highly private. Revealing such information can make an ISP subject to directed attacks along with revealing confidential information. ISPs also have economic reasons to hide this information as it may reveal shortcomings of their own deployments to competitors. For example, in one recently publicized case, a tier-1 ISP published information about their internal failure patterns in a technical paper and a second ISP re-published that information in their marketing literature to convince customers to use the second ISP's own services [4]. Therefore, to enable PCA-based anomaly detection across ISPs, we must ensure the privacy of data regarding internal traffic information.

To address this problem, we design for the target architecture shown in Figure 1. Each ISP runs a Secure Exchange Point (SEP) that collects information about its traffic and coordinates with SEPs located in other collaborating ISPs to diagnose anomalies together. To simplify deployment, the SEP runs on a separate server infrastructure and communicates with routers using existing protocols (SNMP and NetFlow to learn traffic information and the IGP to learn topology information). SEPs may be configured into arbitrary topologies following the trust relationships between ISPs (the inter-SEP connections may traverse multiple intermediate ISPs, if the two collaborators are not directly adjacent). ISPs often already run dedicated infrastructures to detect anomalies and our design can be incorporated into such existing deployments to reduce need for new infrastructure.

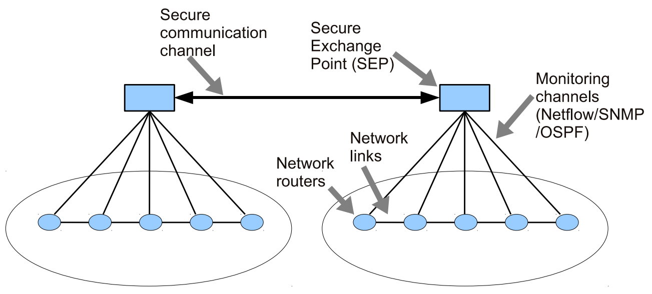
Secure communication channel
Secure Exchange Point (SEP)
Monitoring channels (Netflow/SNMP /OSPF)
Network links
Network routers

Fig. 1: P3CA system architecture.

2.1 Threat model

Adversarial model: In our work, we assume that adversaries are *semi-honest* (also referred to as *honest but curious*), as defined in [14]. In this model, all participants correctly follow the protocol but may observe and record arbitrary information from protocol message exchanges. We believe this model is appropriate for inter-ISP collaborations. ISPs enter into contracts that require them to follow the protocol as well as perform periodic audits to verify its correct operation. However, any private data that is revealed directly to another ISP (for e.g., through accidental misconfiguration) is difficult to contain. Thus, our goal is to limit the amount of private data that can be obtained by any ISP following the protocol.

Privacy goals: Our aim is to develop an privacy preserving scheme where several ISPs can jointly perform PCA on their private traffic observation datasets and detect anomalies in their networks while revealing no further information. In particular, there are three main sources of information about ISPs and we aim to reduce the amount of information revealed about each of them:

1. *Topology:* Each ISP consists of a set of routers and links organized into a graph. We would like to avoid revealing any information about this graph, including its size and topology.
2. *Workload:* Each ISP forwards data traffic between its routers. We would like to avoid revealing information about the set of inter-communicating hosts, packet headers and the volume of traffic.
3. *Monitoring infrastructure:* Each ISP runs a monitoring infrastructure to monitor and collect information about traffic for anomaly detection. We would like to avoid revealing information about the structure, size and visibility of this infrastructure, including the number of vantage points and their placement within the network.

2.2 Protocol overview

We build upon several key features of previous work [20, 21] and provide P3CA, a protocol the allows ISPs to detect anomalies in their networks by computing the principal components of their aggregated data in a privacy preserving manner. In our model, p ISPs collaborate to jointly perform PCA over a distributed traffic matrix Y. Y is a $t \times l$ non-symmetric dense matrix with $l = m \cdot p$, the total number of vantage points at p ISPs, each running (up to) m vantage points internally. Y can be represented as $Y = \left[Y_1|Y_2|\ldots|Y_p\right]$ where each Y_i $(i = 1 \ldots p)$ is an $t \times m$ matrix supplied (i.e., *owned*) by ISP i and '|' indicates column-wise concatenation. Each column of matrix Y_i corresponds to the traffic values collected by ISP i from a specific vantage point and each row corresponds to the traffic values collected over one time interval from all vantage points. The specific mechanism by which Y is measured may be selected independently by the participants.

Given a $t \times l$ matrix Y distributed across p ISPs as described above, we wish to privately compute the principal components of Y. The SEPs of the collaborating ISPs execute the following protocol to find the principal components of the combined traffic matrix Y:

1. All ISPs jointly select two special parties, the *Coordinator* and the *Keyholder*. The Coordinator collects encrypted data from all ISPs and all data is en-

crypted with the Keyholder's public key using the Paillier homomorphic encryption scheme [24] (described in Appendix A.1). The Coordinator uses the Keyholder to perform computations on the encrypted data after blinding it. In this way, neither the Coordinator nor the Keyholder are trusted with the plaintext content of actual traffic data. We note that the Coordinator and the Keyholder are unique for the entire protocol and are elected before each execution of the protocol. To minimize the threat of a compromised Coordinator, the computation may be repeated by multiple Coordinators, with simple voting to resolve conflicts. Likewise, the Keyholder key may be replaced by one generated in a distributed fashion by all the ISPs; queries to the Keyholder could then be replaced by distributed encryption [12].

2. All ISPs execute the semi-centralized procedure P3CA (Algorithm 1) to obtain the encrypted set of n principal components $\{Enc(\boldsymbol{x}_i)\}_{i=1}^{n}$ of the matrix Y. n is chosen apriori and is typically between 5-8 [20, 21].
3. Each ISP i now uses these encrypted principal components to verify if its traffic matrix Y_i has any anomalies. This is done by calculating the residual traffic matrix $Enc(R_i) = (I - Enc(P)Enc(P)^T)Y_i$ where $Enc(P) = [Enc(\boldsymbol{x}_1)|Enc(\boldsymbol{x}_2)|\ldots|Enc(\boldsymbol{x}_n)]$. $Enc(R_i)$ is blinded by multiplying with a random rotation matrix R (an orthogonal matrix) and a random number r similar to the procedure described in Section 3.3, decrypted with the help of the Keyholder and unblinded by multiplying it with R^T (the inverse of R) and dividing by r. ISP i then uses statistical methods like Q-statistic [18] over R_i to detect the anomalies in its own network.

3 The P3CA Algorithm

In this section we present the Privacy Preserving PCA (P3CA) algorithm, which enables multiple cooperating ISPs to calculate the principal components of their combined traffic matrix without revealing their private values to others. We start by describing the core P3CA algorithm (Section 3.1). We then describe three subroutines used in P3CA (Sections 3.2–3.4). We further give extensions to P3CA to support *incremental* computation by leveraging previously-computed results to speed up processing of incoming updates to the traffic matrix (Section 3.5).

3.1 P3CA Overview

P3CA computes the principal components of the traffic matrix Y distributed across p parties (ISPs), with each party holding one or more columns of data. This translates to the computation of the top n eigenvectors of the corresponding covariance matrix YY^T such that none of the participants learn any information about the principal components of the matrix. However, calculating the entire covariance matrix is fairly expensive, requiring $O(l^2t)$ operations. To reduce computation overhead, P3CA uses a modified version of the *power method* (original method described in Appendix A.2), reducing computational costs to $O(lt)$, and reducing communication costs from $O(t^2)$ to $O(t)$. Algorithm 3 (pseudo-code given in Appendix) provides a centralized version of our scheme for plaintext input.

Algorithm 1 presents P3CA, a *semi-centralized privacy preserving* version of Algorithm 3 in which a set of p collaborating ISPs privately compute the top n principal components of the $t \times l$ traffic matrix Y. For this, we introduce secure linear algebra primitives later in this section which are used as building blocks. These include efficient privacy preserving matrix-vector multiplication: *MULR* and *MULC* (Section 3.2), privacy preserving vector normalization: *VECNORM* (Section 3.3) and privacy preserving number comparison: *INTCMP* (Section 3.4).

```
Input: $P$ is the set of ISPs contributing data including the Coordinator but
        excluding the Keyholder and $|P| = p$. ISP $i$ has its traffic matrix $Y_i$
Output: Top $n$-eigenpairs of $YY^T$ namely encrypted principal component
        matrix $P = (Enc(\boldsymbol{x}_1), Enc(\boldsymbol{x}_2), \dots, Enc(\boldsymbol{x}_n))$ and corresponding
        eigenvalues $\lambda_1, \dots, \lambda_n$ are known to all ISPs
Notation:
1. $A \Rightarrow B : M$ denotes a network communication of M from party A to party B;
2. $A \Longleftrightarrow B : r = f(\cdot)$ indicates that parties A and B compute $f(\cdot)$ in a multi-step
protocol ending with party A holding the result $r$.
3. In $F()$ [[text]], text is the corresponding plain text equivalent of $F()$.
4. $\oplus$ and $\otimes$ denotes addition and multiplication on ciphertext as illustrated in
Appendix A.1
foreach Eigenpair $(\lambda_q, \boldsymbol{x}_q)$ for $q = 1, \dots, n$ to be calculated do
    Coordinator: $\boldsymbol{v} = random\_vector()$ ; $S \leftarrow t \times t$ zeros matrix; $\delta \leftarrow Enc(1)$,
    $\lambda_q \leftarrow 0$ ;
    while $INTCMP(Enc(\delta), Enc(\tau|\lambda_q|))$ is TRUE [[While $\delta \geq \tau|\lambda_q|$]] do
        Coordinator $\Longleftrightarrow$ Keyholder: $Enc(\hat{\boldsymbol{v}}) = VECNORM(\boldsymbol{v})$ [[ $\hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{||\boldsymbol{v}||}$ ]];
        $\forall i \in P$ Party $i$: $Enc(\boldsymbol{v}'_i) = MULR(Y_{i*}^T, Enc(\hat{\boldsymbol{v}}_i))$ [[ $\boldsymbol{v}' = Y^T\hat{\boldsymbol{v}}$]];
        $\forall i \in P$ Party $i \Longleftrightarrow$ Coordinator:
        $\boldsymbol{w} = MULC(Enc(Y), [Enc(\boldsymbol{v}')_1|Enc(\boldsymbol{v}')_2|\dots|Enc(\boldsymbol{v}')_p])$ [[$\boldsymbol{w} = Y\boldsymbol{v}'$]];
        Coordinator: $Enc(\boldsymbol{v}) = \boldsymbol{w} - S \otimes Enc(\hat{\boldsymbol{v}})$ [[$\boldsymbol{v} = \boldsymbol{w} - S\hat{\boldsymbol{v}}$]];
        Coordinator $\Longleftrightarrow$ Keyholder: $Enc(\lambda_q) = MULC(Enc(\hat{\boldsymbol{v}}^T), Enc(\boldsymbol{v}))$
        [[$\lambda_q = \hat{\boldsymbol{v}}^Y \boldsymbol{v}$]];
        Coordinator: $Enc(\delta) = \boldsymbol{v} - Enc(\hat{\boldsymbol{v}}\lambda_q)$ [[$\delta = \boldsymbol{v} - \hat{\boldsymbol{v}}\lambda_q$]];
    end
    Coordinator: generates a random value $a$;
    Coordinator $\Rightarrow$ Keyholder: $Enc(a') = a \otimes Enc(\lambda)$ [[$a' = a + \lambda$]];
    Keyholder $\Rightarrow$ Coordinator: $a'$ (decryption);
    Coordinator: $\lambda_q = a' - a$, $Enc(\boldsymbol{x}_q)$=$Enc(\boldsymbol{v})$ [[$\boldsymbol{x}_q = \boldsymbol{v}$]] ;
    $S = S + \lambda_q * (\hat{\boldsymbol{v}}^T\hat{\boldsymbol{v}})$ [[$S = S + \lambda_q(\hat{\boldsymbol{v}}^T\hat{\boldsymbol{v}})$]] ;
end
```

Algorithm 1: P3CA: Privacy Preserving PCA

Handling fixed point computation and negative numbers: The inputs in our traffic matrix Y are floating point numbers. To perform real arithmetic over a finite field, we represent floating point numbers as fixed point numbers by multiplying them by a fixed base. By ensuring that the modulus N is large enough, we can obtain the necessary precision our application requires. Further, N of the field used for encryption is chosen to prevent overflow: for a core router with 10Gbps bandwidth (2^{35} values) and 5 minute time bins (2^7 values), a 42-bit key is sufficient. We use 1024-bit keys in our implementation because this is necessary to achieve an acceptable security level in Paillier encryption. In our experiments (see Section 4), we use a base of 10^6. Negative numbers are represented by subtracting them from the modulus; so $-x$ is represented as $N - x$.

Security: The security of the P3CA algorithm results from the security of the individual steps, as described in the following sections. Further, we use existing methods to ensure that malicious inputs to P3CA do not affect the accuracy of our method. The details of this are discussed in Appendix B.

3.2 Private matrix-vector multiplication

The P3CA algorithm performs computations, iteratively, on the combined traffic matrix Y containing different columns belonging to different ISPs. All ISPs are collectively required to first compute the product $YY^T Enc(\boldsymbol{v})$ where $\boldsymbol{v}$ holds the current estimate of a principal component of Y. Since the matrix is distributed across multiple ISPs , we require a scheme to securely multiply a matrix with a vector, without leaking information about the contents of either the matrix or the vector. We implement this step in a distributed fashion, with all the ISPs participating. Private matrix-vector computation algorithms proposed in the past [19] are designed for a two-party model and would be computationally inefficient due to the fact that they require the entire matrix Y to be encrypted and assembled at the Coordinator.

In P3CA, the matrix Y is column-wise distributed among the p ISPs . Correspondingly, Y^T is distributed row-wise. We, thus, perform the computation $YY^T Enc(\boldsymbol{v})$ in two steps; first we compute $\boldsymbol{v'} = Y^T \boldsymbol{v}$ and then $\boldsymbol{v} = Y\boldsymbol{v'}$ These steps are computed using the *MULR* and *MULC* primitives (described below) which satisfy the following privacy goals:

Privacy goals: Given the vector $Enc(\boldsymbol{v})$ and matrix Y, the *MULR* and *MULC* protocols should ensure the privacy of the length ($||\boldsymbol{v}||$) and direction of the vector $\boldsymbol{v}$ and the length and direction of the columns of Y.

Matrix-vector multiplication with row ownership (*MULR*): Given the $l \times t$ matrix Y^T, we would like to calculate $\boldsymbol{v'} = Y^T \boldsymbol{v}$. Each party owns one or more rows Y^T_{i*} of Y^T and has access to $Enc(\boldsymbol{v})$. This means that each party can locally compute a part of $Enc(\boldsymbol{v'})$ without having to exchange the values of $Y^T_{i,*}$ with others. To compute $Y^T \boldsymbol{v}$, the ISP which owns the row Y^T_{i*} of the matrix Y^T (corresponds to owning column i on matrix Y) computes encrypted value of the i^{th} element of $\boldsymbol{v'}$ i.e. $\boldsymbol{v}'_i$, using $Enc(\boldsymbol{v}'_i) = ((Enc(\boldsymbol{v}_1) \otimes Y^T_{i1}) \oplus \ldots \oplus (Enc(\boldsymbol{v}_t) \otimes Y^T_{it}))$ (Notation $\oplus$ and $\otimes$ given in Appendix A.1).

Matrix-vector multiplication with column ownership (*MULC*): Now the second step of computing $\boldsymbol{v} = Y\boldsymbol{v'}$ uses the result of *MULR*. The party which owns the i^{th} column of matrix Y i.e. Y_{*i} also has access to $Enc(\boldsymbol{v}'_i)$, the ciphertext of the i^{th} element of $\boldsymbol{v'}$ from MULR. The party owning column i in Y

computes $Enc(Y'_{ji}) = Enc(\boldsymbol{v}_i) \otimes Y_{ji}$ for $j = 1 \ldots t$, and forwards $Enc(Y'_{ji})$ to the Coordinator. This step requires $O(t)$ exponentiations and $O(t)$ communication. The Coordinator then computes the encrypted j^{th} element (for $j = 1 \ldots t$) of the new estimate of $\boldsymbol{v}$, i.e. $Enc(\boldsymbol{v}_j)$ using $Enc(\boldsymbol{v}_j) = Enc(Y'_{j1}) \oplus \ldots \oplus Enc(Y'_{jl})$ which finally gives the ciphertext of the result of $YY^T\boldsymbol{v}$. This step requires $O(l)$ multiplications and $O(t)$ communication.

Security: Note that each party transmits values encrypted under the Keyholder's public key, and no party involved in this protocol has a copy of the secret key. The Paillier encryption scheme is known to be CPA-secure [24]; thus these protocols reveal no information about the vector or matrix in the honest-but-curious setting.

3.3 Private vector normalization

In P3CA computation, we have a vector $\boldsymbol{v}$ which is an estimate of the principal component of the traffic matrix Y. Part of the P3CA computation involves normalizing $\boldsymbol{v}$, which is done to speed up the convergence of the power method in practice. Since different parts of $\boldsymbol{v}$ contain information from different ISPs, we need some way to normalize the vector in a privacy-preserving way. We now describe a technique to perform this efficiently.

Given an input vector $\boldsymbol{v}$, the normalization of $\boldsymbol{v}$ is simply another vector $\hat{\boldsymbol{v}}$ in the same direction as $\boldsymbol{v}$ but of unit length (or norm). The Coordinator holds the encryption of vector $\boldsymbol{v}$, i.e. $Enc(\boldsymbol{v})$ (as a result of *MULC* described above) to be normalized while the Keyholder holds the corresponding decryption key. At the end of the private vector normalization protocol *VECNORM*, the Coordinator holds the encryption of the normalized vector $Enc(\hat{\boldsymbol{v}})$ whilst neither of them gains any information about the input vector.

Privacy goals: Given input vector $\boldsymbol{v}$, the *VECNORM* protocol ensures that both length $||\boldsymbol{v}||$ of the vector and its direction are not revealed.

Vector normalization (*VECNORM*): In order to secure vector normalization, the Coordinator rotates the encrypted input vector $Enc(\boldsymbol{v})$ to hide its direction and multiplies the result with an random integer to hide the length of the vector. It then sends this modified vector to the Keyholder who decrypts it, normalizes it and returns the result. The Coordinator then derives the normalization of the input vector, $Enc(\hat{\boldsymbol{v}})$, using the Keyholder's result. The protocol can be summarized as follows:

1. *Blinding direction:* In this step, the Coordinator generates a $t \times t$ transformation matrix R that maps a given vector $\boldsymbol{v}$ to a randomly chosen vector $\boldsymbol{w}$ on a c-dimensional sphere, S^c, of radius $||\boldsymbol{v}||$. Here $c + 1 \leq t$ is a security parameter chosen such that the security of the scheme is the same as that of the finite field used by the cryptosystem; i.e., $|S^c| \geq N$. For $i = 1 \ldots (c + 1)$, the Coordinator generates an orthogonal rotation matrix R_i using 3 parameters: (θ_i, p_i, q_i), where (p_i, q_i) are selected from $1, \ldots, t(t-1)/2$ uniformly at random without replacement and θ_i is chosen uniformly at random from $-\pi$ to π. R_i can be represented as:

$$R_i = \left[\begin{array}{lccccccc}
 & & & \text{col. } p_i & & \text{col. } q_i & & \\
 & & & \downarrow & & \downarrow & & \\
 & 1 & \dots & 0 & \dots & 0 & \dots & 0 \\
 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\
\text{row } p_i \rightarrow & 0 & \dots & \cos(\theta_i) & \dots & -\sin(\theta_i) & \dots & 0 \\
 & \vdots & & & \ddots & & & \vdots \\
\text{row } q_i \rightarrow & & \dots & \sin(\theta_i) & \dots & \cos(\theta_i) & \dots & 0 \\
 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\
 & 0 & \dots & 0 & \dots & 0 & \dots & 1
\end{array}\right]$$

The Coordinator then multiplies all the R_i's to obtain a single transformation matrix R using $R = R_{c+1} * R_c * \ldots * R_1$. and applies the rotation transformation R on $Enc(\boldsymbol{v})$ to get $Enc(\boldsymbol{v}_{rot}) = Enc(\boldsymbol{v}) \otimes R$.

2. *Blinding length:* the Coordinator generates a random blinding factor r and computes $Enc(\boldsymbol{v}'_{rot}) = Enc(\boldsymbol{v}_{rot}) \otimes r$ to blind the length $||\boldsymbol{v}||$ of the vector $\boldsymbol{v}$. Note that both blinding and rotation are required so that no information about $\boldsymbol{v}$ is leaked.
3. The Coordinator sends $Enc(\boldsymbol{v}'_{rot})$ to the Keyholder. The Keyholder then decrypts $Enc(\boldsymbol{v}'_{rot})$ to obtain $\boldsymbol{v}'_{rot}$, computes $\hat{\boldsymbol{v}}'_{rot} = \frac{\boldsymbol{v}'_{rot}}{||\boldsymbol{v}'_{rot}||}$ and sends $Enc(\hat{\boldsymbol{v}}'_{rot})$ to the Coordinator.
4. The Coordinator obtains the normalization of $\boldsymbol{v}$ by applying the inverse of the earlier transformation: $Enc(\hat{\boldsymbol{v}}) = Enc(\hat{\boldsymbol{v}}'_{rot}) \otimes R^T$ to obtain the normalization of $\boldsymbol{v}$. R^T is the inverse of R as it is an orthogonal matrix.

Security: The two blinding steps serve to hide the value of the original vector. Due to the fixed-point representation of the vector values, this hiding is imperfect, as we select a random discretized rotation matrix, rather than a random rotation matrix in general. We have tried to estimate how much uncertainty the Keyholder has about the original vector, given the blinded one. In other words, we computed the conditional entropy $H(O|B)$, where O and B are random variables that represent the original and blinded vector, respectively. To do this, we performed an exhaustive search of the state space, enumerating all possibilities for random choices of rotation matrices and the blinding factor. Of course, we were only able to do this for very small parameter sizes: 2-dimensional vectors and 3- and 4-bit fixed-point representation. Even in this situation, we found that there was between 4 and 5.5 bits of uncertainty, depending on the fixed-point size (out of a total 6–8 bits). Extrapolating from this (albeit limited) data set, we expect that for larger fixed-point sizes, it is possible to introduce 10–15 bits of uncertainty per vector.

This estimate is an information-theoretic upper bound on the success of a possible attack; a computationally bounded adversary would not be able to perform such a brute-force state exploration. A successful attack would have to exploit some algebraic structure of the integers used to represent the fixed-point numbers; we leave the exploration of such attacks to future work and recommend that conservative parameters (i.e., large fixed-point bases with randomized lower bits) be used in practice.

3.4 Private number comparison

P3CA uses an iterative process (the *while* loop in Algorithm 1) to determine each eigenvector of YY^T. In particular, an initial estimate $\boldsymbol{v}_i$ of the eigenvector $\boldsymbol{x}_i$ is chosen, it is checked to see if $\boldsymbol{\delta}_i = Y\boldsymbol{v}_i - \lambda_i \boldsymbol{v}_i$ is within a threshold, τ times the correct eigenvalue λ_i and if not the loop is repeated. Since the contents of Y and hence $\boldsymbol{\delta}_i$ are private, we need some way to securely compare $||\boldsymbol{\delta}_i||$ and $\tau\lambda_i$ without revealing their contents. Since the L^2 norm is difficult to compute for encrypted vectors, we approximate it by using the L^∞ norm. To see if $||\boldsymbol{\delta}_i||_\infty$ is less than $\tau\lambda_i$, the Coordinator executes the *INTCMP* protocol to see if $|(\boldsymbol{\delta}_i)_j| > \tau\lambda_i$ for any j; if so, the power method is continued for another round.

Given encrypted real numbers $Enc(a)$ and $Enc(b)$, *INTCMP* allows the holder of these encryptions to learn if $a > b$ using the Keyholder. The Coordinator learns if $a > b$ and nothing more while the Keyholder learns nothing. The protocol can be summarized as follows:

1. The Coordinator knows $Enc(a)$ and $Enc(b)$. It picks a random r_1 while ensuring that $ar_1, br_1 < 2^{1024}$.
2. The Coordinator sends $Enc((a - b) * r_1)$ to the Keyholder. The Keyholder returns "$>$" if the decrypted value is > 0, "$\leq$" otherwise.

Note that to compute whether $|a| > b$, we must first run *INTCMP* to see if $a < 0$ and then compute $a > b$ or $(-a) > b$ depending on the answer. To ensure privacy from the Keyholder, the Coordinator should randomly swap a and b during the *INTCMP* protocol.

Security: As in the previous protocol, blinding does not provide perfect hiding; in particular, if the Keyholder can estimate the maximum values for $a - b$, and r_1, he can learn some information when the values are in fact close to the limit. However, we can pick r_1 from a very large range (e.g., $[1, 2^{500}]$), thus reducing the chance that we will pick values close to the maximum. Note that the semi-honest model is essential for security in this step, as otherwise the Coordinator could decrypt an arbitrary number by performing a binary search with *INTCMP* [26].

3.5 Incremental Private PCA Computation

So far, the P3CA algorithm we discussed requires the entire set of inputs from all the parties to be available before finding the principal components of the input dataset (Y). However in the context of ISPs jointly computing PCA, an anomaly detection system needs to function as new traffic data arrives. To further reduce computation overhead, we describe how to modify our approach to enable the result to be incrementally updated rather than performing the entire computation from scratch when new information arrives. This allows principal components to be incrementally derived for a long stream of incoming network traffic, thereby speeding up their computation.

Several techniques have been proposed in the image processing literature to address this requirement. Our scheme for incremental private PCA computation is based on the popular and rigorously analyzed CCIPCA [32] (Candid Covariance-free Incremental PCA) algorithm, which provides an iterative method for updating the principal components when new data arrives. In Algorithm 2, we extend CCIPCA to privately compute the principal components of the traffic matrix of p cooperating ISPs. Consider the traffic matrix Y for time intervals $1, \ldots, t-1$ containing rows $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-1}$ (recall each $\boldsymbol{u}_i$ is distributed across ISPs). Now suppose a new traffic vector $\boldsymbol{u}_t$ is generated for time interval t. The new eigenpair for this modified traffic matrix (with u_t included) is estimated as: $\boldsymbol{x}_t = \frac{1}{t} \sum_{i=1}^{t-1} \boldsymbol{u}_i \boldsymbol{u}_i^T \boldsymbol{x}_i$. The idea is to update Y with a covariance matrix estimate using the new distributed vector $\boldsymbol{u}_t$ and $\boldsymbol{x}_{i-1}$ is set as $\boldsymbol{x}_i$ which is the eigenvector estimate obtained using the P3CA method. Note that the entire covariance matrix is not recalculated from scratch.

```
Input: Eigenpairs $(\lambda_1, \boldsymbol{x}_1) \ldots (\lambda_n, \boldsymbol{x}_n))$ over traffic vectors $\boldsymbol{u}_1 \ldots \boldsymbol{u}_{t-1}$, and $\boldsymbol{u}_{i,t}$;
       $P$ be the set of collaborating ISPs
Output: Top $n$-eigenpairs of traffic vectors $\boldsymbol{u}_1 \ldots \boldsymbol{u}_t$ namely
        $(\lambda_1, \boldsymbol{x}_1), \ldots, (\lambda_k, \boldsymbol{x}_n)$ are known to all ISPs
foreach eigenvector required $i = 1, 2, \ldots, n$ do
    foreach $p \in P$ do
        $\boldsymbol{u}^i_{p,t} = \boldsymbol{u}_{p,t}$;
        Party $p \Rightarrow$ party $q \in P \setminus p$: $Enc(\boldsymbol{u}^i_{p,t})$;
        $Enc(\boldsymbol{a}_p) = (Enc(\boldsymbol{u}^i_{p,t}) \otimes Enc(\boldsymbol{u}^i_{1,t}), \ldots, Enc(\boldsymbol{u}^i_{p,t}) \otimes Enc(\boldsymbol{u}^i_{|P|,t}))$;
        $Enc(\boldsymbol{b}_p) = \frac{(t-1-l)}{t} * (\lambda_i \boldsymbol{x}_i) + Enc(\boldsymbol{a}_p) \otimes \frac{1+l}{n} \frac{\lambda_i \boldsymbol{x}_i}{||\lambda_i \boldsymbol{x}_i||}$;
        // except $\boldsymbol{a_p}$ all other inputs are plaintext and known to all ISPs
    end
    Party $p \Rightarrow$ party $q \in P \setminus p$: $Enc(\boldsymbol{b}_p)$;
    $\boldsymbol{u}_{p,t} = \boldsymbol{u}_{p,t-1} - \boldsymbol{u}_{p,t-1} \otimes VECNORM(\boldsymbol{b})_p$ ;
    //Remove any component in the direction of the new eigenvector to ensure
    orthogonality of eigenvectors
end
```

Algorithm 2: Incremental P3CA extensions

4 Evaluation

Our design enables ISPs to collaborate to detect anomalies. However, our design also comes at some cost. First, it incurs additional computation overhead since it requires encryption and multi-round exchanges between nodes. Second, the use of fixed point operations in our design can lead to a loss of precision resulting in the calculation of principal components that are potentially different compared to those calculated by PCA. To quantify the affect of these, we evaluate the computational overhead of our design (Section 4.1) and compare the detection probability using PCA with that using our algorithm (Section 4.2).

To measure these costs, we constructed an implementation of our design. Our design is implemented in roughly 1000 lines of C++. We use the GMP library for large numbers (bignums), OpenMP library for parallelization and the libpaillier library for the Paillier cryptosystem [1]. All encryptions use a 1024-bit key. To evaluate performance over realistic workloads, we leveraged traces from the Abilene Internet2 IP backbone [2]. In particular, we used NetFlow traces to determine traffic volumes between source/destination IP addresses, and used OSPF and BGP traces to map the flows onto the underlying physical topology.

4.1 Scalability

To evaluate the computational overhead of P3CA, we measured the amount of time it takes for our implementation to finish running on its input data set. To characterize performance, we would ideally like to run our design on varying network topology sizes. However, due to the privacy of traffic information, acquiring traces from a number of different-sized ISPs represents a substantial challenge. Hence, to study scalability for different network sizes, we extrapolated a traffic model from the Abilene dataset and used that to construct synthetic traces for larger networks. Extrapolation of traffic traces is itself a challenging research problem and beyond the scope of this paper. We use a fairly simple extrapolation procedure: we generate a random network topology, randomly select flows contained in the Abilene data set, and map their ingress/egress to random pairs of routers contained in the generated topology. We then take this graph and divide it up into a set of ten constituent ISPs. We do this by picking ten random points in the graph, and performing breadth-first search to select nearby regions to form the ISP (repeating this process until ten connected ISPs are created).

Table 1a shows run time of a single run of our algorithm as a function of topology size, where we vary (i) the total number of monitored links in all ISPs (l), and (ii) the total number of time bins (each lasting 10 minutes) used for traffic history when computing principal components (t). These experiments were performed on an 3.07GHz Intel Core i7 processor with 6GB memory. As a comparison, we note that large networks only monitor a subset of links (e.g.,the tier-1 ISP network used for PCA in [21] had $l = 41$, as monitoring a subset of core links is sufficient to observe most traffic in the network) and t is often set to small values to speed reaction and adapt to recent changes. Table 1b shows the run time for incrementally refreshing current PCA results with new traffic observations using Algorithm 2.

We find that P3CA requires only a few minutes to run, even for relatively large numbers of monitored links. P3CA requires only around 10 minutes to process data for 320 minutes, making it an *efficient online algorithm* for anomaly detection. Further, the advantages of using incremental P3CA are clearly evident (as seen in Table 1b): incremental updates are processed within 6 seconds even for large networks with a total of 320 links for a data spanning 320 minutes. This makes incremental P3CA as efficient as PCA on raw data. To investigate the source of computational bottlenecks, we instrumented our code with timing functions to collect the microbenchmarks shown in Table 2a. Our design can be trivially parallelized across a cluster of machines or CPU cores to further reduce overhead, as shown in Table 2b. While the Coordinator requires additional computation, this additional work may be distributed across several machines to accelerate computation and improve resilience.

l	t	Party i (secs)	Coordinator (secs)
20	2	0.525	2.183
40	4	1.376	6.472
80	8	3.529	14.134
160	16	52.999	194.175
320	32	194.126	637.649

(a) P3CA

l	t	time (secs)
20	2	3.56
40	4	3.72
80	8	4.03
160	16	4.66
320	32	5.91

(b) Incremental P3CA

Table 1: Performance and scalability of original and incremental P3CA with increasing # of links l and # of bins t for 5 eigenpairs on 4 processors;

Percent of time	Operation
39.6%	multiplying cipher and plain texts
36.6%	adding cipher texts
18.2%	decryption
16.5%	private vector normalization
82.5%	private matrix-vector multiplication and subtractions

(a) Microbenchmarks. The first three and the last two rows represent two different breakdown of operations.

l	t	r	Party i (secs)	Coordinator (secs)	
320	32	8	194.1	637.6	(10m 37s)
640	64	8	6649.6	20823.4	(5h 47m)
320	32	32	48.5	151.4	(2m 31s)
640	64	32	1662.4	5205.8	(1h 27m)
320	32	64	24.2	75.7	
640	64	64	881.2	2602.9	(43m 22s)

(b) Performance of P3CA with increasing numbers of processors r for l links, t bins

Table 2: Microbenchmarks and evaluation of parallelization of P3CA for 5 eigenvectors

4.2 Precision

The performance of PCA algorithms in general has been widely evaluated in previous work. Our approach performs essentially the same computation as PCA, but might potentially lose some precision due to the use of fixed point numbers. We note that some implementations of PCA intentionally round to filter minor traffic fluctuations. To evaluate how the precision of our algorithm affects the results, we must use a more realistic topology and traffic information. We use the real Abilene dataset and topology here (but do not investigate sensitivity to network size), and run PCA and P3CA to detect traffic volume anomalies. To investigate sensitivity to anomaly type, we also inject synthetic anomalies with different characteristics. To do this, we randomly choose time bins, and insert a constant amount of extra traffic on a randomly chosen subset of 1 to 5 links.

We ran the experiment 100 times with different random seeds, for two different kinds of injected anomalies: *small* corresponds to the case in which the volume of the injected anomalies is twice the volume of the background traffic on the link, and *large* corresponds to the case in which the anomalies have a volume which is ten times larger than the background traffic in the link. We use the Q-statistic test [18] for detecting abnormal variations in the traffic at a 99.9% confidence level. Figure 2 plots the CDF of the anomaly detection percentage of PCA and

P3CA. The cumulative fraction is over the multiple runs we performed. We find that in every run, P3CA and PCA computed nearly the same result (detected the same set of anomalies).

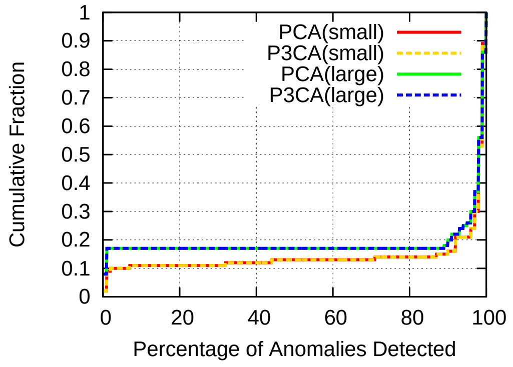

Fig. 2: Comparison in precision between PCA and P3CA

5 Related Work

To our knowledge, our work is the first attempt to perform scalable privacy-preserving traffic anomaly detection. Our work builds upon two key areas of previous results:

Anomaly detection in ISP networks: Given the increasing severity of DoS, scanning, and other malicious traffic, traffic anomaly detection in large networks is gaining increased attention. Lakhina et. al. [20, 21] showed that PCA has much potential in uncovering anomalies by leveraging traffic matrices constructed using summarizations of traffic feature distributions. While there are alternatives to PCA (for e.g., [35]), PCA-based approaches remain a state-of-the-art technique due to its robustness to noise and high efficiency on limited data. Extensions to PCA make it robust to attacks such as variance-injection [33], and enable PCA to be used for other goals, such as diagnosing network disruptions [17].

Further, accuracy of anomaly detection is improved with more visibility of traffic. If ISPs cooperated to share data, accuracy of anomaly detection could be substantially improved. Soule et. al. [30] show that by jointly analyzing the data of peering ISPs more anomalies were detected, especially those anomalies that transited the two ISPs they studied. However, traditional anomaly detection requires sharing of detailed traffic traces, which are considered highly private by network providers. Our work extends PCA to multiple parties, preserving the privacy of participants' data. By extending PCA, our approach computes the same result as this well-proven technique, retains the desirable properties mentioned above and enables more widespread cooperation of ISPs to counter the increasing threat of malicious traffic.

Secure multiparty computation: Secure multiparty computation (SMC) techniques allow a collection of participants to compute a public function F without revealing their private inputs to F. Generic SMC techniques date back to Yao [34] and Goldreich et. al. [15] and have been well studied in cryptography literature [6, 16]. Recent years have seen some improvements in efficiency [11, 7]. However from the viewpoint of the systems designer, the generic schemes are only of theoretical interest. For the scale of computation required for mining anomalies in Internet traffic, privacy and security must be added with manageable costs. Developing a practical, scalable way of computing PCA in a privacy-preserving way is the main focus of our work.

P4P [13] presents a generic scheme for private computation of statistical functions that can be represented as an iterated matrix-vector product. When used to compute PCA, the privacy goal of P4P and several other schemes is to reveal no more information apart from the principal components themselves. However, given the eigenvectors and eigenvalues of a matrix it is possible to reconstruct the matrix itself. When used by ISPs, this scheme would reveal the eigenvectors of Y^TY but not YY^T for traffic matrix Y. In the context of the concrete problem of anomaly detection, this does not constitute privacy preservation at all since Y^TY can be reconstituted to a close approximation from its eigenvectors and eigenvalues (end result of the P4P scheme section 6 of [13]). Y^TY can then be used to infer Y (the input distributed matrix) which is supposed to be private. This can be done as follows: suppose Y_{ij} (traffic volume) is a real number between 1 and N. Y_{ij} is the dot product of columns Y_{*i} and Y_{*j}. When the elements of column Y_{*i} and Y_{*j} are close to N then Y^TY_{ij} will be close to maximal. Similarly, close to minimal values in Y_{*i} and Y_{*j} leads to a close to minimal Y^TY_{ij}. Therefore Y^TY_{ij} can be used to construct Y to a close approximation. In contrast with P4P, our scheme presents an advancement in that we only reveal the variance of a projected traffic data point Y_{*j}, namely $||PR_i||^2$ where P the eigenvector matrix of YY^T itself is never revealed.

6 Conclusions

The increasingly distributed nature of malicious attacks renders their identification and localization a difficult task. The ability to identify traffic anomalies across multiple ISPs could be a significant step forward towards this goal. P3CA represents an important step, by allowing a set of ISPs to collectively identify anomalous traffic while limiting information propagation across them. P3CA scales to large and high-bandwidth networks addressing the need for refreshing current results with fresh traffic observations, yet retains the accuracy and precision of PCA-based approaches. We envision our work as an important step towards enabling larger-scale cooperation across ISPs to counter the increasingly serious threats posed by modern networked malware.

References

1. `http://acsc.cs.utexas.edu/libpaillier/`.

2. http://www.internet2.edu/network/.
3. A Border Gateway Protocol 4 (BGP-4). RFC 4271.
4. Private communication, employee of tier-1 ISP. 2006.
5. G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the kth-ranked element. In *Eurocyrpt*, 2004.
6. D. Beaver and S. Goldwasser. Multiparty computation with faulty majority. In *CRYPTO*, 1989.
7. Hao Chen and Ronald Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *CRYPTO*, 2006.
8. B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954, October 2004.
9. C. Croux, P. Filzmoser, and M. Oliveira. Algorithms for projection-pursuit robust principal component analysis. In *Chemometrics and Intelligent Laboratory Systems*, 2007.
10. C. Croux and G. Haesbroeck. Principal component analysis based on robust estimators of the covariance or correlation matrix: Influence functions and efficiencies. In *BIOMETRIKA*, 2000.
11. I. Damgard, Y. Ishai, M. Kroigaard, J. Nielsen, and A. Smith. Scalable multiparty computation with nearly optimal work and resilience. In *CRYPTO*, 2008.
12. I. Damgard and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography*. Springer, 2001.
13. Y. Duan, N. Youdao, J. Canny, and J. Zhan. P4P: Practical large-scale privacy-preserving distributed computation robust against malicious users.
14. O. Goldreich. Secure multi-party computation. *Theory of Cryptography Library*, 1999. http://philby.ucsb.edu/cryptolib/BOOKS.
15. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *ACM Symposium on Theory of Computing*, 1987.
16. S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *CRYPTO*, 1991.
17. Y. Huang, N. Feamster, A. Lakhina, and J. Xu. Diagnosing network disruptions with network-wide analysis. In *SIGMETRICS*, 2007.
18. J. Edward Jackson and Govind S. Mudholkar. Control procedures for residuals associated with principal component analysis. *Technometrics*, 21:341–349, August 1979.
19. E. Kiltz, P. Mohassel, E. Weinreb, and M. Franklin. Secure linear algebra using linearly recurrent sequences. In *TCC*, 2007.
20. A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *ACM SIGCOMM*, pages 219–230, 2004.
21. A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *ACM SIGCOMM*, pages 217–228, 2005.
22. R. B. Lehoucq and D. C. Sorensen. Deflation techniques for an implicitly restarted arnoldi iteration. In *SIAM J. Matrix Anal. Appl.*, 1996.
23. Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. 2008. http://eprint.iacr.org/.
24. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.
25. H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of pca for traffic anomaly detection. In *SIGMETRICS*, June 2007.
26. R.L. Rivest, L. Adleman, and M.L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, 1978.

27. B. Rubenstein, B. Nelson, L. Huang, A. Joseph, S. Lau, S. Rao, N. Taft, and D. Tygar. Antidote: Understanding and defending against poisoning of anomaly detectors. In *IMC*, November 2009.
28. Fernando Silveira and Christophe Diot. Urca: pulling out anomalies by their root causes. In *INFOCOM*, March 2010.
29. Gerard L. G. Sleijpen and Henk A. Van der Vorst. A jacobi–davidson iteration method for linear eigenvalue problems. In *SIAM Rev.*, 2000.
30. A. Soule, H. Ringberg, F. Silveira, J. Rexford, and C. Diot. Detectability of traffic anomalies in two adjacent networks. 2007.
31. R. Vasudevan, Z. Mao, O. Spatscheck, and J. Van der Merwe. Reval: A tool for real-time evaluation of DDoS mitigation strategies. In *USENIX ATC*, 2006.
32. J. Weng, Y. Zhang, and W. Hwang. Candid covariance-free incremental principal component analysis. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2003.
33. W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan. Detecting large-scale system problems by mining console logs. In *SOSP*, 2009.
34. A. Yao. Protocols for secure computations (extended abstract). In *FOCS*, 1982.
35. Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network animography. In *IMC*, 2005.

A Appendix: Background

In this section, we describe two existing techniques we build upon in our design: *homomorphic encryption* and the *power method*.

A.1 Homomorphic Encryption

A homomorphic cryptosystem allows operations to be performed on plaintext by performing a corresponding operation on its ciphertext. In our scheme participants only have access to the encrypted data of others. They can perform computations over it without knowing its unencrypted value hence protecting the privacy of the party supplying the data. To protect privacy, we use the Paillier encryption [24] to perform computation on encrypted values. We now briefly describe the operation of the Paillier cryptosystem. The original cryptosystem is defined over scalars, but we present its natural extension to encrypted vectors. (They can be readily extended to handle matrices).

Given a public key (N, g) produced by a key-generation algorithm, a random number $r \in \mathbb{Z}_N$ and a k-dimensional vector $\boldsymbol{u} = (u_1, \ldots, u_k) \in \mathbb{Z}_N^k$, its encryption, $Enc(\boldsymbol{u})$, is given by: $Enc(\boldsymbol{u}) = (g^{u_1} r^N \bmod N^2, \ldots, g^{u_k} r^N \bmod N^2)$

Suppose we are given two vectors $\boldsymbol{u} = (u_1, \ldots, u_k)$ and $\boldsymbol{v} = (v_1, \ldots, v_k)$ in $\mathbb{Z}_N^k$. The Paillier encryption scheme provides us the two following properties which we use to perform various arithmetic operations on ciphertexts in P3CA:

1. We can compute the encrypted value of the sum of $\boldsymbol{u}$ and $\boldsymbol{v}$ by multiplying their corresponding ciphertexts: $Enc(\boldsymbol{u} + \boldsymbol{v}) = Enc(\boldsymbol{u}) \oplus Enc(\boldsymbol{v}) = (Enc(u_1) * Enc(v_1) \bmod N^2, \ldots, Enc(u_k) * Enc(v_k) \bmod N^2)$

2. We can compute of the encrypted value of the product $\boldsymbol{u}$ and $\boldsymbol{v}$ by multiply the ciphertext of $\boldsymbol{u}$ i.e. $Enc(\boldsymbol{u})$ and the plain text value of $\boldsymbol{v}$: $Enc(\boldsymbol{u} * \boldsymbol{v}) = Enc(\boldsymbol{u}) \otimes \boldsymbol{v} = (Enc(u_1)^{v_1} \bmod N^2, \ldots, Enc(u_k)^{v_k} \bmod N^2)$

A.2 The Power Method

To use principal component analysis, we are required to find the n largest principal components of the traffic matrix Y. This translates to finding the n largest eigenpairs of its covariance matrix $COV = YY^T$. The *power method* [29] is one of the appropriate candidate techniques when n is much smaller than the rank (sum total of traffic observations) of the covariance matrix. Indeed, previous studies [20, 21] indicate that five to eight principal components capture most of the variance within ISP traffic. Based on this, we believe that the power-method is the most appropriate technique for PCA, which we briefly describe below. To calculate the principal components of Y, we replace Y by COV in the following.

The power method first computes the dominant eigenvector, x_1, of a matrix Y by simply choosing a random vector $v_0^1 \in \mathbb{R}^l$ and iteratively multiplying v_0^1 by powers of Y until the resulting vector converges to x_1. This is ensured as long as the starting vector v_0 has a non-zero component along x_1. A single iteration is given by:

$$v_j^1 = \frac{Yv_{j-1}^1}{||Yv_{j-1}^1||}; j = j + 1 \tag{1}$$

This process is repeated until $j = s$, the smallest value for which $||Yv_j^1 - v_j^1\lambda_1|| \leq \tau|\lambda_1|$. The corresponding eigenvalue is computed using $\lambda_1 = \frac{{v_s^1}^T Y v_s^1}{{v_s^1}^T v_s^1}$.

To obtain the next largest eigenvector the power method uses a well known deflation technique [22]. Once the i^{th} eigenpair (x_i, λ_i) (x_i is the i^{th} eigenvector and λ_i is the i^{th} eigenvalue) is computed, a transformation is applied on the matrix Y to move λ_i to the center of the eigenspectrum. To compute the $i + 1^{th}$ largest eigenvector the power method applies the following transformation on Y_i, where Y_i is the matrix used for computing the i^{th} dominant eigenvector of Y: $Y_{i+1} = Y_i - \lambda_i \frac{x_i x_i^T}{x_i^T x_i}$ with $Y_1 = Y$.

This process is repeated until n eigenpairs have been found. The parameter n need not be decided beforehand. Instead, n is simply specified in terms of the smallest eigenvalue of the eigenpair we are interested in. Upon uncovering eigenpair (λ_i, x_i), if $\lambda_i \leq \epsilon$, the eigenpair is discarded and the algorithm terminates. ϵ can be interpreted as the accuracy required for representing column vectors of Y as a linear combination of its eigenvectors.

B Handling malicious inputs

PCA is an excellent example of how machine learning techniques can assist in anomaly detection. However in its basic form it is fairly vulnerable in that a small fraction of false inputs can significantly change the final result. To address this problem, Croux published a series of papers [10, 9] showing that PCA could be made more robust by centering input data over the median instead of the mean. In ANTIDOTE [27], Rubenstein et al. study the malice resistance modifications proposed by Croux in the context of anomaly detection in AS networks. They show that the modifications are of significant help in defending against malicious

```
Input: $t \times l$ matrix $Y$; $\tau$, the convergence parameter
Output: Top $n$-eigenpairs of $YY^T$ namely $(\lambda_1, \boldsymbol{x_1}), \dots, (\lambda_n, \boldsymbol{x_n})$
foreach Eigenpair $(\lambda_{q \le n}, \boldsymbol{x_{q \le n}})$ to be calculated do
    $\delta \leftarrow 1$; $\boldsymbol{v} \leftarrow random\_vector()$; $S \leftarrow t \times t$ zeros; $\lambda_q \leftarrow 0$;
    while $\delta \ge \tau|\lambda_q|$ do
        $\hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{||\boldsymbol{v}||}$;
        $\boldsymbol{v}' = Y^T\hat{\boldsymbol{v}}$;
        $\boldsymbol{w} = Y\boldsymbol{v}'$ ;
        $\boldsymbol{v} = \boldsymbol{w} - S\hat{\boldsymbol{v}}$;
        $\lambda_q = \hat{\boldsymbol{v}}^T v$;
        $\delta = \boldsymbol{v} - \hat{\boldsymbol{v}}\lambda_q$;
    end
    $\boldsymbol{x_q} = \boldsymbol{v}$;
    $S = S + \lambda_q(\hat{\boldsymbol{v}}^T\hat{\boldsymbol{v}})$;
end
```

Algorithm 3: Algorithm for computing principal components of a matrix

inputs that could, for instance, enable a participant to hide the presence of a DDoS attack.

P3CA readily supports the modifications proposed by Croux and verified by Rubenstein et al. In particular, along with the primitives we discussed above, we need a privacy-preserving method of computing the median over the entire dataset and centering the data over the median. This requirement is met by the scheme of Aggarwal et al. [5] which computes the median of a distributed dataset among N parties in the honest-but-curious threat model. Its complexity is $O(N(\log M)^2)$ where $\log M$ is the number of bits needed to describe each (unencrypted) scalar input.