“Discriminative Sensing Based on Signal Processing”

By

Thiago Pereira de Brito Vieira

M.Sc. Dissertation

Universidade de Brasília
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

BRASÍLIA, June/2018

Universidade de Brasília
Departamento de Engenharia Elétrica - ENE/FT
Progama de Pós-Graduação em Engenharia Elétrica - PPGEE

Thiago Pereira de Brito Vieira

"Discriminative Sensing Based on Signal Processing"

A M.Sc. Dissertation presented to the Departamento de Engenharia Elétrica - ENE/FT of Universidade de Brasília in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering.

Advisor: *João Paulo Carvalho Lustosa da Costa*
Co-Advisor: *?*

BRASÍLIA, June/2018

Ficha Catalográfica

Signatures

Dedicatory.

Agradecimentos

First and foremost, I would like to thank God for giving me the life, strength, knowledge and opportunity to undertake this research study and to ..

(...)

Thank You!!

Wherever you go, go with all your heart.

—CONFUCIUS

Resumo

Último passo, por ser um resumo da introduçao, resultados e conclusão.

Palavras-chave: Discriminative Sensing, Signal Processing, Eigen Similarity Analysis, Tensor Based Dictionary Learning, Critical Factor Analysis, Network Attack Detection, Fraud Detection

Abstract

Last step, after introduction, results and conclusion.

Keywords: Discriminative Sensing, Signal Processing, Eigen Similarity Analysis, Tensor Based Dictionary Learning, Critical Factor Analysis, Network Attack Detection, Fraud Detection

Contents

List of Figures xiii

List of Tables xiv

List of Acronyms xv

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Problem Statement . . . 1
- 1.3 Contributions . . . 1
- 1.4 Thesis Organization . . . 1

2 PCA and SVM for Critical Factors Analysis 2

3 Model Order Selection and Eigen Similarity based Framework for Detection and Identification of Network Attacks 11
- 3.1 Related Works . . . 12
- 3.2 Data Model . . . 15
  - 3.2.1 Analyzed Scenario and Data Collection . . . 15
  - 3.2.2 Modeling Data . . . 16
  - 3.2.3 Synflood, Fraggle and Port scan . . . 18
  - 3.2.4 The DARPA Dataset . . . 20
- 3.3 Proposed Framework for Detection and Identification of Network Attacks 21
  - 3.3.1 Largest Eigenvalue by Time Frames . . . 22
  - 3.3.2 MOS Schemes . . . 23
  - 3.3.3 Eigenvalue Analysis . . . 24
  - 3.3.4 Eigen Similarity Analysis . . . 25
    - Time Similarity Analysis . . . 25
    - Port Similarity Analysis . . . 28
- 3.4 Experiments and Results . . . 29
  - 3.4.1 Experimental Scenario . . . 29
  - 3.4.2 Largest Eigenvalues Analysis . . . 29
  - 3.4.3 MOS Schemes Evaluation . . . 32
  - 3.4.4 Eigenvalue Analysis . . . 33
  - 3.4.5 Eigen Similarity Analysis . . . 33

Time Analysis . . . 34
Port Analysis . . . 37
3.4.6 DARPA Scenario . . . 40
3.5 Performance Evaluation . . . 42
3.5.1 Complexity Analysis . . . 42
3.5.2 Processing Time Analysis . . . 43
3.6 Conclusion and Future Works . . . 45

4 Offline Mode for Corporate Mobile Client Security Architecture 47
4.1 Introduction . . . 47
4.2 Related works . . . 49
4.3 The mobile security architecture . . . 51
4.4 The proposed solution for offline mobile security . . . 53
4.4.1 Offline mode workflow . . . 55
4.4.2 Offline Behavioral Analysis . . . 56
4.5 The algorithms, key usage and data protection methods . . . 58
4.5.1 AES file encryption . . . 59
4.5.2 ABE encryption to protect the FILE_KEY . . . 60
4.5.3 Secret sharing scheme to protect the key storage . . . 61
4.5.4 MOS for Threat Intelligence . . . 62
4.6 Results and analysis . . . 65
4.6.1 Security analysis . . . 65
Adversary model . . . 65
Security analysis . . . 65
Common threat scenarios . . . 67
Data Modeling for Behavioral Analysis . . . 70
4.6.2 Complexity analysis . . . 72
4.7 Conclusion and future work . . . 76

5 Tensor-Based Discriminative Dictionary Learning 78

6 Conclusion and Future Work 79
6.1 Conclusion . . . 80
6.2 Contributions . . . 81
6.2.1 Lessons Learned . . . 82
6.3 Future Work . . . 83

Bibliography **84**

List of Figures

3.1 Scenario to reproduce legitimate traffic, noise, flood and port scan. . . . 15
3.2 Traffic from user's operations, that can be characterized by web access, traffic of well-known applications or network protocols. 17
3.3 Network traffic of user independent operations for network management. 17
3.4 A large quantity of SYN requests to a target, in order to cause a DoS. . . 18
3.5 Large amount of "UDP echo" requests and replies, causing packet flooding. 19
3.6 Connection attempts in order to identify active ports. 20
3.7 Overview of The Framework for Detection and Identification of Network Attacks. 21
3.8 Traffic selection for incremental approach. 26
3.9 Traffic selection for individual approach. 27
3.10 Traffic selection for incremental individualized approach. 27
3.11 Eigenvalues of the sample covariance matrix (synflood). 30
3.12 Eigenvalues of the sample covariance matrix (fraggle). 31
3.13 Eigenvalues of the covariance matrix of zero mean and unitary standard deviation (port scan). 31

4.1 The core set of functions and protocols of the mobile cloud security infrastructure 51
4.2 The Mobile Client Architecture 52
4.3 Proposed Architecture for Offline Mobile Security 54
4.4 Offline Mode Operations 56
4.5 Offline mode authorization workflow 56
4.6 The Threat Intelligence Manager Workflow 57
4.7 Encryption workflow 58
4.8 Encryption key hierarchy 59
4.9 Server-side encryption 59

List of Tables

3.1 Largest Eigenvalue related to attacks detection 31
3.2 MOS schemes applied to port scan and flood detection 32
3.3 Eigen Similarity Analysis for Port Scan Detection 34
3.4 Eigen Similarity Analysis for Synflood Detection 36
3.5 Eigen Similarity Analysis for Fraggle Detection 36
3.6 Eigen Similarity Analysis for Detection of Ports Under Port Scan Attack (q=3 and n=15) 38
3.7 Eigen Similarity Analysis for Detection of Ports Under Synflood Attack (q=4 and n=11) 39
3.8 Eigen Similarity Analysis for Detection of Ports Under Fraggle Attack (q=5 and t=11) 39
3.9 Results of the attack detection evaluation 41
3.10 Processing time of the main steps for anomaly detection 44

4.1 Speed of decryption on a client device 72
4.2 Speed of receiving the list of files 72
4.3 Speed of generating the user keys 73
4.4 Data Modeling and Eigenvalue Decomposition Time 73
4.5 EDC MOS scheme processing time for anomaly detection 77

List of Acronyms

CSF Critical Success Factor

CFA Critical Factors Analysis

PCA Principal Component Analysis

DL Dictionary Learning

1 Introduction

Though nobody can go back and make a new beginning, anyone can start over and make a new ending.

—CHICO XAVIER

1.1 Motivation

...

1.2 Problem Statement

...

1.3 Contributions

...

1.4 Thesis Organization

...

2

PCA and SVM for Critical Factors Analysis

No one knows it all. No one is ignorant of everything. We all know something. We are all ignorant of something.

—PAULO FREIRE

There is a significant continuous public spending on information technology (IT) by the Brazilian Federal Public Administration. In order to monitor and to diagnose the concerned public institutions IT governance (ITG), the Federal Court of Accounts, in Portuguese Tribunal de Contas da União (TCU), surveys data from these institutions practices as well as best practices for IT in government and business. These surveys started in 2007 with 39 questions and nowadays they contain more than 100 questions impacting on the ITG of Brazilian public institutions. Moreover, these questions encourage public institutions to adopt best practices not only in IT but also in all other management areas. In this paper, we propose the identification and the verification of the critical success factors (CSFs) of the TCU survey variables by means of statistical analysis. The CSFs are defined as the factors that most contributed for the high performance of the institutions and they are measured by using the ITG index created by TCU. Therefore, in order to validate our results, the statistically identified CSFs are compared with those mentioned by the public administration IT executives in interviews. Besides the statistical analysis, we successfully apply the Support Vector Classification (SVC) algorithm to classify public institutions in terms of their ITG index. By comparing the SVC based classification with the CSFs obtained from the interviews, we show that there is a very high level of similarity. Hence, the CSFs identified with SCV provide a very high impact, regarding

the public administration ITG.

2.3 Análise de Componentes Principais

Segundo Sabin, Ferrão e Furtado (2004) e Silva et al. (2012), a Análise de Componentes Principais (ACP), do inglês Principal Component Analysis – PCA, é utilizada para compressão de dados para identificação das relações entre características dos dados. Tal compressão é obtida por meio da substituição das variáveis originais por um novo conjunto de variáveis, conhecidas como Componentes Principais (CPs). As CPs são obtidas pela combinação linear das variáveis que apresentam a maior variabilidade na matriz de covariância. Assim, objetivo principal da ACP é identificar as CPs responsáveis pelas maiores variações entre os resultados. Em termos práticos eliminam-se algumas variáveis com pouca informação e determinam-se as variáveis de maior influência na formação de cada CP (Vicini, 2005). Segundo Vicini (2005), o conjunto ortogonal de eixos não correlacionados representam as variáveis que mantém ao máximo a variabilidade do conjunto, em outras palavras, a menor perda de informação. Conforme Vicini (2005:29), para o processo de determinação das CPs "é necessário calcular matriz de variância-covariância ($\sum$), ou na matriz de correlação (), encontrar os autovalores e autovetores e, por fim, escrever as combinações lineares, que serão as novas variáveis, denominadas de componentes principais, sendo que cada componente principal é a combinação linear de todas as variáveis originais [...] em ordem de estimação e em termos da variância total, contida nos dados iniciais". Ressalta-se que não são todos as CPs a serem analisadas, mas apenas as com maior variância, alguns autores (Vicini, 2005) determinam os CPs por meio da porcentagem que representam, geralmente, mais de 70% da informação. Conforme Silva et al. (2012) para a geração dos CPs, considera-se uma matriz X = (X1, X2, ..., Xp) de dimensões nxp (a Seção 4 mostra as dimensões utilizadas nesta pesquisa) originada a partir de um vetor aleatório. A matriz de variância-covariância das amostras da matriz X é calculada por meio da seguinte expressão , onde é o operador de transposição de uma matriz. Nota-se que a matriz de variância-covariância é uma matriz quadrada e não negativa. Portanto, pode-se aplica a decomposição em autovalores, do Eigenvalue Decomposition (EVD), da matriz resultando em um produto de três matrizes. , vi = A i-nésima coluna da matriz é o autovetor normalizado correspondente ao autovalor . Desta forma, define-se i-ésima componente principal (Yi) como sendo Salienta-se que a ACP é baseada na EVD já que parte dos autovalores e dos autovetores representam as componentes principais (CP). A Figura 1 mostra geometricamente a ACP de duas variáveis X1 e X2. A nuvem de pontos representa o diagrama de espalhamento destas duas variáveis. Conforme explicado, o primeiro passo é o cálculo da matriz de

variância-covariância e em seguida se aplica a EVD. Verifica-se que a CP1, denominado CP principal (com a maior variância), é ortogonal à segunda CP (CP2 com a menor variância). Note que as duas variáveis são perpendiculares porque as variáveis obtidas pela ACP são independentes. Logo, após a aplicação da ACP, ao invés de representar os dados por meio de duas variáveis correlacionadas X1 e X2, eles podem ser representados por duas variáveis independentes Y1 e Y2. Dada a elevada correlação entre X1 e X2, verifica-se que as duas variáveis X1 e X2 podem ser aproximadas por uma única variável Y1 na Figura 1.

Figura 1 – Representação e Rotação do Componente Principal Fonte: Desenvolvido pelos autores

A Figura 1 reforça a ideia da ACP cujo objetivo inicial é a identificação de planos e linhas que representem um conjunto de pontos em um espaço com um número menor de variáveis tendo em vista as possíveis correlações que podem existir entre as variáveis originais. Assim, para o método da estatística multivariada ACP utilizar todas variáveis, separa-se a informação útil da informação redundante (Finkler, 2003). Ressalta-se que a ACP é um dos primeiros passos para outras análises multivariadas, sendo um dos métodos mais comuns empregados na análise de informações (Sabin, Ferrão e Furtado, 2004; Silva et al.,2012).

2.4 Máquinas de Vetores de Suporte (MVS)

Máquina de Vetores de Suporte (MVS), do inglês Support Vector Machine (SVM), é um método supervisionado de aprendizagem de máquina, não probabilístico, baseado na teoria de aprendizagem estatística, usado para classificação, regressão e detecção de padrões. MVS pode ser aplicado por meio de dois passos: o primeiro passo é o treinamento de um modelo a partir de um determinado conjunto de dados; o segundo consiste em estimar a classificação de dados a partir da aplicação do modelo treinado. Quando aplicado para classificação, MVS busca a identificação de um hiperplano que separa os dados em classes distintas, por meio de uma margem máxima entre duas classes de dados. Desta forma, um conjunto de dados é linearmente separável se for possível dividir seus dados em duas classes, por meio um hiperplano, conforme na figura a seguir.

Figura 2 – Separação Linear por Margem Máxima

Dado um conjunto de dados previamente classificados, (xi, yi), xi $\in$ Rn, yi $\in$ 1, -1, i = 1, . . . , l, um classificador linear pode ser definido por: wT $\cdot$ x + b = 0, desta forma o hiperplano ótimo é definido pelos valores ótimos do vetor de pesos wi e do bias bi, de forma que wT $\cdot$ x + b = 1 e wT $\cdot$ x + b = -1 representam respectivamente os vetores de suporte positivos e negativos, que são os pontos próximos à margem do hiperplano ideal.

A margem máxima, entre os vetores de suporte, é definida por:

As funções de kernel têm o objetivo de projetar os vetores de variáveis de um conjunto de dados em um espaço de maior dimensão, para a classificação de classes originalmente apresentadas em espaços não separáveis linearmente. Com o aumento da dimensão, aumenta a probabilidade desses dados poderem ser linearmente separáveis. Foi adotada a estratégia "um contra um" (Knerr, Personnaz e Dreyfus, 1990), para a utilização de MVS para a classificação de muitas classes. Esta estratégia consiste em construir uma máquina vetor de suporte para cada par de classes. Para um problema com c classes, c(c-1)/2 MVSs são treinados para classificar as classes entre as c classes possíveis.

2 Eliminação Recursiva de Variáveis (ERV)

Dado um algoritmo de classificação, que possa estimar pesos para as variáveis de um conjunto de dados, o objetivo da Eliminação Recursiva de Variáveis (ERV), do inglês Recursive Feature Elimination (RFE) (Guyon, 2002), é selecionar variáveis por meio da redução recursiva da quantidade de variáveis, eliminando recursivamente as variáveis de menor peso para classificação por meio do algoritmo adotado. Primeiramente, um modelo é treinado, utilizando o conjunto de dados inicial e o algoritmo selecionado, durante o treinamento são atribuídos pesos para cada variável, representando a importância de cada variável para a classificação. Em seguida, as variáveis com menor peso são eliminadas do conjunto de dados. Este processo, de treinamento, ordenamento de variáveis e eliminação de variáveis menos importantes, é repetido recursivamente até a obtenção do número desejado de variáveis, ou até a satisfação de alguma condição, como um limiar de taxa de erro de um algoritmo de classificação. As variáveis com maiores pesos apresentam maior influência na classificação (Guyon, 2002). Desta forma, se um algoritmo de classificação apresenta boa acurácia, as variáveis com maiores pesos representam as variáveis que apresentam maior influência para a classificação. MVS-ERV (Guyon, 2002) é uma aplicação da ERV utilizando os pesos obtidos por meio do treinamento utilizando MVS como algoritmo de classificação, para identificar as variáveis mais importantes para predições de classificação e eliminar recursivamente as variáveis que menos influenciam na classificação.

4.3 Análise das componentes principais

Para a análise eficiente da matriz original é relevante separar as informações úteis das redundantes. Segundo Johnson e Wichern (1992), há vários instrumentos para essa finalidade, mas a ACP é a que melhor desempenha este papel. Uma vez calculados os autovalores da matriz original, temos um resultado que mostra que aproximadamente 70% da variabilidade dos dados é explicado por 51 componentes principais. O critério

para a escolha desses fatores foi o de identificar os autovalores que possuem variância acumulada em torno de 70% (Mardia, 1979). Tal valor também será utilizado nesta pesquisa (Quadro 7).

Quadro 7 – Autovalores Ordem dos autovalores Autovalores da variância explicada Autovalores acumulados da variância explicada acumulada Fonte: Dados da Pesquisa

Depois da extração dos autovalores e percentual da variância explicada, é decidida a quantidade de fatores a serem retirados para análise. Para isso, o Gráfico 1 em que o total de autovalores está no eixo das ordenadas e os autovalores no eixo das abscissas, auxilia na identificação. Esse gráfico consiste no ranking dos autovalores (eixo x), relacionado com o valor de cada autovalor (eixo y). Verifica-se que uma queda menos acentuada ocorreu entre o quarto e o quinto autovalor e analisando-se os autovalores superiores a 2, observa-se que pode-se considerar até o vigésimo valor já que a partir daí os valores dos autovalores sucessivos são praticamente constantes.

Gráfico 1 – Ranking dos autovalores Fonte: Dados da pesquisa

Visando encontrar os planos fatoriais realizou-se uma rotação dos eixos, onde as cargas fatoriais mais elevadas são as responsáveis pelas denominações das componentes e são estatisticamente significativas. As rotações de eixos melhor expressam a dispersão de dados. No modelo fatorial final, as variáveis das medidas estão maximizadas e as relações entre dimensões suavizadas (Vicini, 2005). Para esta análise buscam-se valores que possuem significância maior que 0,7, mostrando que a correlação entre as variáveis está de moderada a forte (Vicini, 2005). Essa identificação não seria possível sem a rotação dos eixos, possibilitando assim a melhor visualização das variáveis mais significativas em cada componente. Tal rotação mantem os eixos perpendiculares entre si, ou seja, ortogonais e a variabilidade do sistema não é alterada, apenas as coordenadas dos eixos são rotacionadas e a inércia do sistema fica inalterada. A partir dos valores obtidos pela rotação das componentes principais, podem-se obter valores com significância maior que 0,7. Logo, foi possível a identificação das variáveis significantes de cada componente principal. Para visualização desses fatores foi utilizado o gráfico de dispersão (Gráfico 2). O Gráfico 2 mostra a caixa de seleção de variáveis e comandos para ACP em que se utilizam os fatores 1 e 2, eixo x e eixo y respectivamente. O objetivo deste gráfico é fazer os planos principais com a nuvem de pontos dos indivíduos, no caso as 349 instituições, destacando o posicionamento das respostas das instituições classificadas pelo iGovTI. Tal gráfico se baseia na rotação dos componentes principais. Para a elaboração do Gráfico 2 apenas os e foram utilizados.

Gráfico 2 – Relação entre os e Fonte: Dados da pesquisa

No Gráfico 2, para analisar apenas a , projetam-se os pontos sobre o eixo da e se tem três grupos (V, II e I). O mesmo processo se aplica a analise da . Assim, observam-se três grupos (V, IV e III). Para cada grupo se seleciona apenas uma variável representativa, o restante é desconsiderado, uma vez que cada grupo significa correlação. Para o grupo I, tomam-se as variáveis Q16d, Q16e e Q16f que são descorrelacionadas, já no grupo II a questão Q16g é questão selecionada. Ambos os grupos não apresentam FCS. Essas questões denotadas Q16, referem-se a Subdimensão 1.6 (A Alta Administração Utilizou Informações Fornecidas pela Auditoria Interna). Tal subdimensão verifica a participação da auditoria interna das instituições para o preenchimento do questionário. Depois, buscou-se por meio da análise das , e a identificação de FCS (Gráfico 3). O Gráfico 3 mostra a análise dessas componentes principais que revela a existência de três grupos.

Gráfico 3 – Relação entre os , e Fonte: Dados da pesquisa

A análise do Gráfico 3 mostra revela a existência de três grupos (VI, VII e VIII). O grupo I contém a variável Q51l que corresponde a FCS e o grupo II também mostra FCS, a variável Q23b. Na identificação das variáveis significativas das primeiras 20 componentes principais, com os autovalores superiores a 1, as questões que mais contribuem são Q16b (para responder às questões do grupo 2. Estratégias e planos), Q16c (para responder às questões do grupo 3. Informação e conhecimento), Q16d (para responder às questões do grupo 4. Pessoas), Q16e (para responder às questões do grupo 5. Processos) e Q16f (para responder às questões do grupo 6. Resultados da gestão), todas da dimensão "Governança corporativa e de TI, a alta administração utilizou informações fornecidas pela auditoria interna (ou instância equivalente)". Nota-se que as variáveis Q16, citadas anteriormente, fazem parte do grupo I do Gráfico 2. Ao total foram identificadas 29 variáveis dos 20 primeiras componentes principais (autovalores maiores que 1) das variáveis com significância maior que 0,7. O Quadro 8 mostra quais são essas variáveis significativas.

Quadro 8 – Variáveis com alta significância CP1 a CP20 Fonte: Dados da pesquisa

No Quadro 8, verificou-se as variáveis que são FCS, de acordo com resultado da pesquisa apresentada na Subseção 4.1. Das 30 questões apenas 5 são FCS, de acordo com classificação obtida por meio de pesquisa qualitativa, quais sejam: Q12c (designou representantes de todas as áreas relevantes para o negócio institucional para compor o Comitê de TI); Q23b (a instituição aprovou e publicou PDTI interna e externamente); Q51l (gestão de configuração de ativos); Q53e (formalizou a política corporativa de segurança da informação); Q57h (os pagamentos são feitos em função da mensuração objetiva dos resultados entregues e aceitos. Ainda no Quadro 8, chama-se atenção as , e. Nelas constam variáveis com valor positivo ou negativo. Tal fato significa que quanto

maior for a resposta de uma questão menor será a da outra. Frisa-se que isto só vale para questões que estejam em uma mesma variável ACP. Caso as questões tenham sinais diferentes (i.e. os elementos dos autovetores), mas em ACP diferentes o sinal positivo e negativo não importa. A análise das categorias (Quadro 5) relacionadas às variáveis revela: Q53e referente à Categoria Processos de Gestão de Serviços de TI em Desenho do Serviço que mostrou porcentual igual a 53,85%; Q57h se liga à Categoria Gestão de Contratos que teve porcentual de 38,46%; Q23b se vincula à Categoria PDTI que teve 38,46%. A variável Q51l se relaciona à Categoria Processos de Gestão de Serviços de TI em Transição de Serviços com 19,23%. Devido ao baixo percentual de 16,66% de identificação de FCS do Quadro 8, realizou-se a análise das variáveis que completam as 51 componentes principais, referentes aos 51 autovalores do Quadro 7. Nesta identificação, foram identificadas 33 variáveis, sendo que 8 são FCS, 24,24% (Quadro 9). Somando os totais obtidos pelo Quadro 8 e pelo se tem, 20,63% de variáveis que são FCS.

Quadro 9 – Variáveis com alta significância CP 21 a CP 51 Fonte: Dados da pesquisa

Já o Gráfico 4 mostra a caixa de seleção de variáveis e comandos para ACP em que se utilizam os fatores 1 e 2, eixo x e eixo y respectivamente. O objetivo deste gráfico é fazer os planos principais com a nuvem de pontos dos indivíduos, no caso as 349 instituições.

Gráfico 4 – Dados brutos da pesquisa Fonte: Dados da pesquisa

O Gráfico 4 representa a relação entre os 2 principais componentes, que representam as questões ou variáveis com maiores autovalores, assim cada ponto representa a relação entre uma instituição e seus 2 principais componentes obtidos por meio de ACP. Cada ponto deste gráfico representa a classificação da instituição em relação a sua classificação no iGovTI de 2012. A partir do resultado apresentado no gráfico 4 é possível identificar um padrão formado pela localização dos componentes principais das organizações com maior índice, onde as organizações com maior iGovTI tiveram valores relativamente semelhantes para seus 2 componentes principais, entretanto ainda não é possível obter uma separação clara entre os resultados. Fazendo-se um corte no eixo das abcissas, a partir do ponto (-8,8, 0), destacam-se 23 instituições à esquerda desse ponto (Quadro 5).

Gráfico 5 – Destaque de corte nas instituições Fonte: Dados da pesquisa

Dessas instituições, 22, são consideradas pelo TCU como aprimoradas, ou seja, 95,65%. Apenas a instituição 346 é avaliada como intermediária. Após a análise ACP, verifica-se que a mesma não é útil para a análise da Hipótese 2. Para responder a essa hipótese, é necessária a execução dos algoritmos de classificação da Subseção 4.4 e da Subseção 4.5.

4.4 Máquinas de Vetores de Suporte

Para prever a classificação de uma instituição de acordo com suas respostas para as questões do questionário iGovTI, foram avaliados algoritmos de classificação que pudessem apresentar acurácia para a classificação de organizações de acordo com o iGovTI. Uma vez encontrado um algoritmo capaz desta classificação, é possível utilizar o algoritmo selecionado em conjunto com técnicas de feature selection para identificar as questões mais relevantes para a classificação da instituição de acordo com o iGovTI. Assim, para a definição do algoritmo de classificação foram realizados testes em 21 algoritmos, a seguir é apresentada uma listagem dos algoritmos avaliados e a taxa de acerto de classificação obtida: 1. KNN: 0.714286 2. ElasticNet: 0.155336 3. ElasticNetCV: 0.831531 4. LassoCV: 0.827440 5. LassoLarsIC: 0.713763 6. LinearRegression: 0.360878 7. LogisticRegression: 0.771429 8. OrthogonalMatchingPursuit: 0.769630 9. PassiveAggressiveClassifier: 0.800000 10. PassiveAggressiveRegressor: 0.852184 11. Perceptron: 0.800000 12. Ridge: 0.529922 13. RidgeClassifier: 0.657143 14. RidgeClassifierCV: 0.742857 15. RidgeCV: 0.750149 16. SGDClassifier: 0.828571 17. MultinomialNB: 0.742857 18. lda.LDA: 0.628571 19. SVM.SVR: 0.826885 20. SVM.SVC: 0.914286 21. SVM.LinearSVC: 0.714286 22. O algoritmo que obteve maior sucesso para classificação foi o SVC, que é uma implementação de Máquina de Vetores de Suporte aplicado para a classificação. SVC apresentou uma taxa de acerto de 91,4%. Para a avaliação dos algoritmos, utilizamos uma metodologia que divide os dados dos questionários pelo iGovTI entre questões que serão utilizadas para aprendizado do algoritmo e questões que serão utilizadas para comparativo de predições. Para treinar o algoritmo utilizou-se 90% dos dados e os 10% restantes foram utilizados para avaliar a eficiência de classificação dos algoritmos, comparando a taxa de acerto entre as predições feitas e os valores reais de classificações de organizações pelo iGovTI. 23. Uma vez identificado um algoritmo capaz de efetuar a classificação desejada, o próximo passo é identificar as variáveis mais relevantes para esta classificação. Para isso, será utilizado o algoritmo ERV. 24. 4.5 Eliminação Recursiva de Variáveis 25. 26. A ERV pode usar vários algoritmos de classificação como critério de seleção das variáveis mais importantes, escolhemos o algoritmo SVC por ele ter apresentado maior acurácia entre os algoritmos de classificação avaliados. Ainda é necessário definir o quantitativo de variáveis mais importantes a serem selecionadas. A partir da pesquisa de identificação por meio de entrevistas na APF, identificou-se 54 variáveis consideradas FCS, este critério foi utilizado para determinar o quantitativo de variáveis mais importantes para a classificação. 27. Assim, aplicaram-se o algoritmo ERV utilizando o SVC como critério para a seleção das variáveis mais importantes para a classificação, selecionando as 54 variáveis que

correspondem aos FCS levantados nas entrevistas com os executivos de TI. Os resultados mostraram que 69,9% das variáveis foram classificadas da mesma forma que os FCS identificados anteriormente (Quadro 10) por meio de pesquisa qualitativa.

3

Model Order Selection and Eigen Similarity based Framework for Detection and Identification of Network Attacks

All difficult things have their origin in that which is easy, and great things in that which is small.

—LAO TZU

Traditionally, cyber defense methods can be effective against ordinary and conventional types of attacks, yet may fail against innovative malicious techniques [14]. In order to be able to detect and avoid novel attacks and their variations, it is necessary to develop or improve techniques to achieve efficiency on resource consumption, processing capacity and response time. Moreover, it is crucial to obtain high detection accuracy and capacity to detect variations of malicious patterns. Recently, signal processing schemes have being applied to the detection of malicious traffic in computer networks [16, 11, 24, 7, 6, 21], showing advances in network traffic analysis.

Information security may consist of both technical and procedural aspects. The former includes equipment and security systems, while the latter corresponds to security rules and recommendations. Intrusion detection and intrusion prevention systems are security systems used, respectively, to detect (passively) and prevent (proactively) threats to computer systems and computer networks. Such systems can work in the following fashions: signature-based, anomaly-based or hybrid [11, 17]. Additionally, anomaly detection techniques can be categorized in classification, statistical, information theory and clustering based, according to [3, 1, 18].

In the context of anomaly-based schemes, this work proposes a statistical approach based on signal processing techniques for the detection of malicious traffic in computer networks. Inspired by [7, 6], this work models the network traffic using a signal processing formulation as a composition of three components: legitimate traffic, malicious traffic and noise, taking into account the incoming and outgoing traffic in certain types of network ports (TCP or UDP). To the best of our knowledge there is no similar model in the literature. The proposed technique is based on eigenvalue analysis, model order selection (MOS) and similarity analysis. In contrast to [7, 6, 21], MOS and eigenvalue analysis are applied to detect time frames under attack. In addition, we also evaluate the accuracy and performance of the proposed framework applied to a experimental scenario and to the DARPA 1998 dataset [18], which is a well known network traffic dataset. Furthermore, this proposed approach has its accuracy evaluation based on eigen similarity analysis for extracting detailed information about accurate time and network ports under attack.

The performed experiments show that synflood, fraggle and port scan attacks can be detected accurately and with great detail in an automatic and blind fashion, applying signal processing concepts for traffic modeling and through approaches based on MOS and eigen similarity analysis. The main contributions of the proposed framework are the capability to blindly detect time frames under network attack via MOS and eigen analysis, and the detailed identification of the network attack via eigen similarity analysis.

This paper is organized as follows. In Section 3.1, related works are discussed. Section 3.2 presents the data model and the evaluated datasets. Section 3.3 describes the proposed framework for blind and automatic detection of flood and probe attacks. Section 3.4 discusses the experimental validation and presents the results, and Section 3.5 discusses the computational complexity of the proposed framework and evaluates the required processing time for tested scenarios. Section 3.6 draws the conclusions and the suggestions for future work. The **??** presents mathematical concepts of examples of state-of-the-art MOS schemes.

3.1 Related Works

Several methods have been proposed for the identification and characterization of malicious activity in computer networks. Classical methods typically employ data mining [10, 9, 18] and regular file analysis [19] to detect patterns that indicate the presence of specific attacks in network traffic.

Data mining is often used to describe the process of extracting useful information from large databases. Multiple methods of data mining are used in [10, 18] to analyze

data flow in a network with the aim of identifying characteristics of malicious traffic in large scale environments. Researchers have applied data mining techniques in log analysis [9] to improve intrusion detection performance. However, data mining techniques used so far in network analysis require prior collection of large data sets, which is a limitation of several schemes for online analysis [11].

Regular file analysis [19] consists of traffic analysis for detecting known patterns that indicate the presence of attacks, applying statistical analysis to the study of collected traffic. An essential feature of this method is that it depends on prior knowledge of the details of the attacks to be identified, and also depends on previous log collection for traffic analysis and false positives reduction.

Principal Component Analysis (PCA) is a statistical technique commonly used for dimensionality reduction. It uses an orthogonal transformation to convert a set of correlated variables into a set of linearly uncorrelated variables, where the first principal components have the largest variance. PCA has been used in attack detection [2]. However, PCA requires human intervention in order to identify abnormalities based on the eigenvalues profiles, if used without complementary techniques.

Callegari *et al* [24] propose a PCA-based method for identifying the traffic flows responsible for an anomaly detected at the aggregate level and evaluated their proposal through a dataset with synthetic anomalies added in the data. However, Callegari *et al* focus on flood attack detection, not addressing probe attack detection, and their approach relies on visual analysis.

Lee *et al.* [15] propose OverSampling PCA (osPCA), which allows one to determine the anomaly of the target instance according to the variation of the resulting dominant eigenvector obtained by similarity analysis and over sampling. In contrast to Lee *et al.*, the framework applies MOS for detection of time frames under attack and similarity analysis to extract details for detection of time and ports under attack. Additionally, Lee *et al.* only evaluate their proposed scheme for covariance analysis, while we adopt an analysis based on sample covariance of zero mean variables and sample covariance of zero mean and unitary standard deviation variables, for flood and probe attacks, respectively.

Besides being prone to higher errors and false positives, such human intervention makes PCA not useful for real time applications. Therefore, in order to automate the analysis of eigenvalues profile, model order selection (MOS) schemes should be incorporated.

Signal processing techniques have been successfully applied to network anomaly detection [16]. Lu and Ghorbani [16] proposed a network anomaly detection model based

on network flow, wavelet approximation, and system identification theory. However, their work requires a training method to produce a prediction model for normal daily traffic and presents limitations on identification of behaviors without significant outliers, such as port scan attacks. Zonglin *et al* [24] proposed a method to detect traffic anomaly with correlation analysis, where the correlation between traffic signals and the predicted traffic signals are used to reveal anomalies. Zonglin *et al.* [24] evaluated the correlation analysis for anomaly detection, but the work is not applied to probe and flood attack detection, simultaneously.

The data collected from honeypot systems, such as captured traffic and operating system logs, can be analyzed to obtain information about attack techniques, general trends of threats and exploits. Blind automatic detection of malicious traffic techniques have been developed for honeypots in [7, 6]. However, traffic on honeypot is simpler than real network traffic, because there are no running legitimate applications, due to the fact that honeypots emulate behavior of a host within a network to deceive and lure attackers [23]. Since honeypots do not generate legitimate traffic, the amount of data captured in honeypots is significantly lower in comparison to a Network Intrusion Detection System (NIDS), which captures and analyzes the largest possible amount of network traffic [7]. MOS for blind identification of malicious activities in honeypots was proposed by us in [7], which evaluated criteria for selecting the model order, through simulations and comparing the order of the resulting model with the true model order.

The proposed framework does not require either a significant amount of logs to detect attacks, nor prior data collection, in order to make comparisons and evaluate the existence of malicious traffic. The proposed solution is automatic and blind for detection of time frames under probe and flood attacks through MOS and eigen analysis. Moreover, we apply eigen similarity analysis to identify details of time and ports under network attacks.

Several approaches for network attack detections uses the KDD 99 [12, 1, 18, 3] datasets for accuracy and performance evaluation, due to their availability and labeled attacks. Even though the KDD 99 dataset are criticized by for the generation procedure and the risk of over-estimations of anomaly detection due to data redundancy, it still represents one of the few publicly available labeled datasets currently in use today by researchers [18, 3]. NSL-KDD [20] dataset is the refined version of the KDD 99 dataset that redundant data records are removed, in order to avoid biased classifications. Additionally, some approaches uses simulated [4] scenarios or non-public datasets their evaluations. Since the proposed approach relies on a packet level analysis and the KDD 99 and NSL-KDD datasets adopt a traffic aggregation by connections, we consider the

use of a experimental scenario on a real network and the DARPA 1998 dataset, which is the source for the creation of the KDD 99 and NSL-KDD datasets. Note that the proposed approach is not based on learning or classification techniques, which are more susceptible to biased results caused by the issues in the DARPA/KDD datasets.

3.2 Data Model

This section presents details of the experimental scenario and the selected cases of the DARPA dataset, along with a description of the dataset model as a signal superposition of legitimate traffic, noise and malicious traffic. Subsection 3.2.1 describes the environment and scenario adopted in order to reproduce flood and probe attacks. Subsection 3.2.2 presents how network traffic can be modeled as signal superposition. Subsection 3.2.3 details the traffic of synflood, fraggle and port scan attacks, and Subsection 3.2.4 discusses the use of the DARPA dataset for evaluation of the proposed approach.

3.2.1 Analyzed Scenario and Data Collection

The environment of the analyzed scenario is composed of two computers and one router with access to the Internet and to an internal network, where the simulation of legitimate traffic, noise, flood and port scan attacks are performed. During the traffic generation, one computer assumes the role of the attacker, while the other is the victim, according to scenario represented by Figure 3.1.

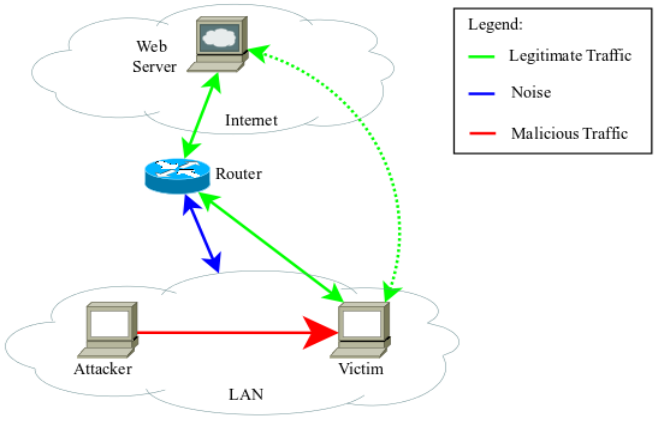

Figure 3.1 Scenario to reproduce legitimate traffic, noise, flood and port scan.

Note that the set of network traffic is modeled as legitimate, noise and malicious traffic, where the victim performs legitimate activities, that can be characterized by web

access. In many organizations this type of traffic is predominant, since most of corporate services are web-based, such as: web pages, customized web-based systems and cloud services. It is possible to characterize the traffic of a DHCP service as an example of noise associated with the transport layer. For malicious traffic, three types of networks attacks are evaluated: synflood, fraggle and port scan. These attacks are reproduced using well-known security tools, such as Nmap[1] to port scan, Metasploit[2] to synflood and Hping[3] to lead the fraggle attack.

A network traffic log is commonly formed by timestamp, protocol, source IP address, source port, destination IP address, destination port and additional information, according to the type of the used transport protocol. The following TCP traffic log is presented in order to exemplify the collected data:

```
21:00:34.099289 IP 192.168.1.102.34712 > 200.221.2.45.80: Flags
[S], seq 2424058224, win 14600, options [mss 1460, sackOK,TS
val 244136 ecr 0,nop,wscale 7], length 0
```

and the following to exemplify UDP traffic log:

```
21:24:42.484858 IP 192.168.1.102.68 > 192.168.1.1.67: BOOTP/DHCP,
Request from 00:26:9e:b7:82:be, length 300
```

In the proposed framework, the goal is to detect the anomalies only taking into account the traffic profile, i.e., specific information such as origin IP or day and time of the attack are not considered. Therefore, from the entire log information, we just consider the timestamp (for sequencing), port type and port number.

3.2.2 Modeling Data

By modeling the dataset as a signal superposition, the network traffic (**X**) can be characterized as a mixture of three components: legitimate traffic (**U**), noise (**N**) and malicious traffic (**A**), according to the following expression:

$$X^{(q)} = U^{(q)} + N^{(q)} + A^{(q)}, \tag{3.1}$$

[1] http://nmap.org
[2] http://www.metasploit.com
[3] http://hping.org

where q represents the q-th time frame, which is a time grouping of network traffic. The matrix $X^{(q)} \in \mathbb{R}^{M \times N}$ consists of M rows and N columns. Each row represents a communication port (TCP port or UDP port), and each column represents time bins having a approtiate size, such as one minute. Each element $x_{m,n}^{(q)}$ stands for the number of times that the port m appears at the n-th minute, at the q-th time frame.

The legitimate traffic $U^{(q)}$ is characterized by the traffic from user's operations. When a user accesses a web page, for example, there is the corresponding TCP/IP traffic to request the page, as well as there is the traffic required to domain name resolution. Figure 3.2 depicts an example of the legitimate traffic obtained during experiments.

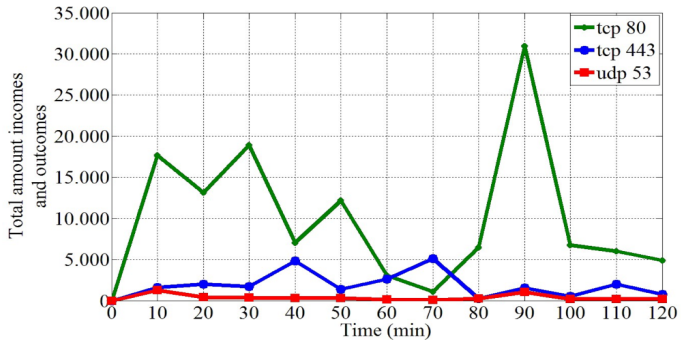

Figure 3.2 Traffic from user's operations, that can be characterized by web access, traffic of well-known applications or network protocols.

The traffic that is not associated with user's operations and with malicious traffic is modeled as noise $N^{(q)}$. The automatic acquisition service of logical IP network address (DHCP) is an example of noise. Independently of any user operation, the machine receives an IP address, since it is configured to perform a DHCP address request. Figure 3.3 depicts an example of noise in a network traffic, represented by traffic to ports 67 and 68.

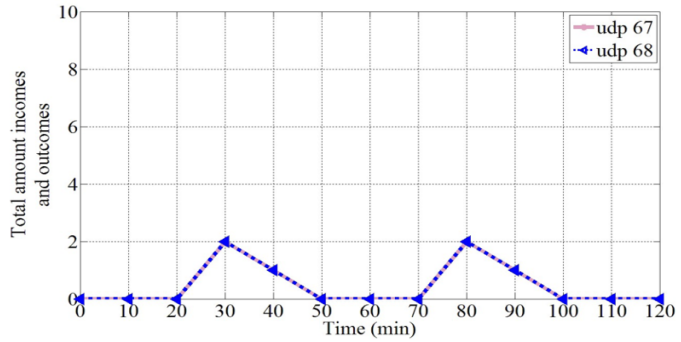

Figure 3.3 Network traffic of user independent operations for network management.

The traffic coming from a malicious activity, such as a synflood or fraggle attack, is represented by the matrix $A^{(q)}$. For this work we only consider the traffic from port scanning and flood attacks.

We define that if the obtained $\#A^{(q)} \neq 0$, then there is malicious traffic in the evaluated time frame q, on the other hand, if the $\#A^{(q)} = 0$, then there is no malicious traffic. This paper shows how to detect the $\#A^{(q)}$, given only the matrix $X^{(q)}$, in order to identify malicious network traffic.

3.2.3 Synflood, Fraggle and Port scan

The network attacks evaluated by this work are: synflood, fraggle and port scan. The first two attacks can be qualified as flood or denial of service (DoS) attacks, while the last one can be qualified as probe or port scanning attack. DoS attempts to block access to system or network resources and is implemented by either forcing targets to be unavailable through the exploiting of system vulnerabilities, or consuming resources through large amount of network traffic, characterizing flood attacks. Probe attacks scan the networks to collect information about host, such as IP addresses, ports and services.

With respect to the synflood attacks, the attacker sends a large quantity and concurrent successive SYN requests to a target, in order to consume resources and cause a DoS. Figure 3.4 depicts an example of a synflood attack carried out in a real computer network. In an interval of ten minutes, more than 210,000 packets are sent as a synflood attack. This network traffic behavior can be considered an abnormal behavior of network traffic, especially since it is concentrated in a short period of time and presents similar outstanding traffic during the time under attack.

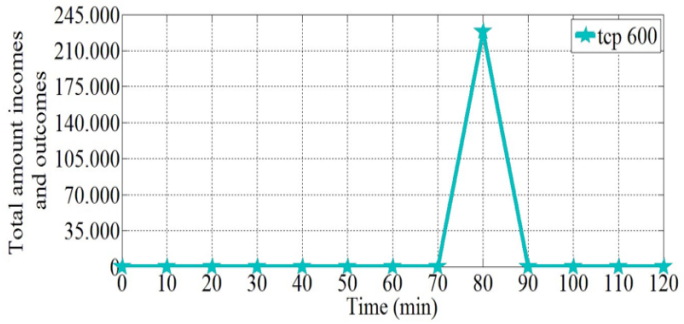

Figure 3.4 A large quantity of SYN requests to a target, in order to cause a DoS.

With respect to the fraggle attack, large packets with UDP echo segments are sent to the broadcast address of a network. Every packet is modified to have the source address

of the victim, in order to implement the source address spoofing technique. Therefore, each host receives a huge amount of requests UDP echo and all of them replies to the IP address of the victim, causing a packet flooding aiming a DoS. This attack can affect the entire network, since all hosts receive several requests UDP echo and respond with the ICMP protocol, therefore each host acts as an amplifier of the attack. This last part of the fraggle attack is not taken into account in this work, because the victim receives ICMP (network layer) packets originated from the hosts that are attacked with flooding packet UDP echo. Figure 3.5 depicts an example of the fraggle attack in a real computer network.

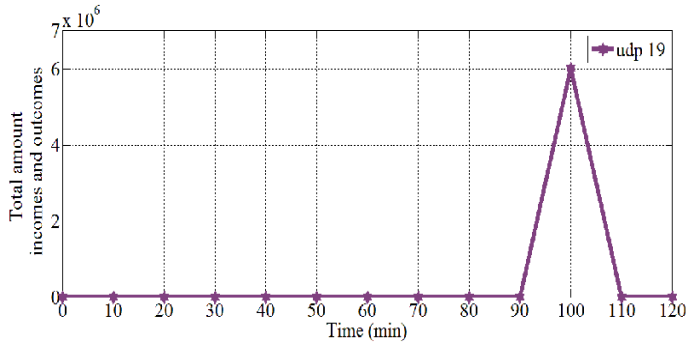

Figure 3.5 Large amount of "UDP echo" requests and replies, causing packet flooding.

More than 6,000,000 malicious packets can be counted in an interval of ten minutes, which can be considered an abnormal network traffic, especially due to the concentrated traffic in a short period of time and due to the similarity of the outstanding traffic.

Port scan is the attempt to establish a connection to TCP and UDP ports to identify what services are running or are in the listening state. There are several available port scanning techniques, including: TCP SYN scan, TCP ACK scan and UDP scan. This work evaluates the use of TCP SYN scan and UDP scan.

In TCP SYN scan, a SYN packet is sent to the destination and two types of responses may occur: SYN/ACK or RST/ACK. In the first case, the destination port is in the listening state, in the second case, the destination port is not listening. At the end of each port scanning, a RST/ACK packet is sent by the system that is performing the port scan. Therefore, a full connection or a complete three-way handshake is never established, which makes the detection of the attack sender more difficult, and requires approaches able to identify probe attacks without connection stablishment. The UDP scan technique sends UDP packets to the destination port, and if it responds with a *ICMP port unreachable* message, then it indicates that the scanned port is closed. On the other hand, if a message is not received, then the port is considered as open.

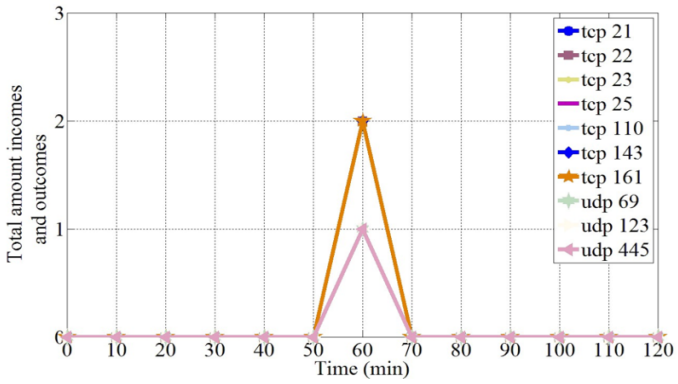

Figure 3.6 Connection attempts in order to identify active ports.

Figure 3.6 depicts an example of the port scan attack in a real computer network. Note that the traffic is composed of two packets for each TCP port and one UDP packet to each port. The incoming and outgoing packets analysis, for each port, shows the high correlation and similarity of TCP and UDP traffic during the simulated port scan attack.

3.2.4 The DARPA Dataset

The DARPA 1998 dataset[4] includes 7 weeks of sniffed traffic saved into raw TCPDUMP packet data, from inside and outside origins, with labeled attacks. The attacks in this dataset can be grouped into: denial-of-service (DoS); remote to local (R2L), which is characterized by unauthorized access from a remote machine; user to root (U2R), which is characterized by unauthorized access to local super-user privileges; and probe attack. Since the proposed approach focus on flood and probe attack, the analysis concentrates on the attacks that present behaviors similar to flood or probe attack. We observe that the most cases of DoS focus on exploit system vulnerabilities instead of on flooding attack. One example is the occurrence of a neptune attack which sends 20 only SYN packets, what is a behavior that differs of the expected flood attack behavior. Therefore, there were selected the cases that simulates several network traffic or numerous connection requests, also known as flooding attack [1, 18], and the cases that scan ports sending just a few packets. From the simulated probe attacks, we select the cases that rely on TCP or UDP connections.

The data modeling follows the method described by the Subsection 3.2.2, with time frames of 20 minutes, packet counting aggregation by minute and considering the traffic to the following ports: 20, 21, 22, 23, 25, 79, 80, 88, 107, 109, 110, 113, 115, 143, 161,

[4]https://www.ll.mit.edu/ideval/data/

389, 443.

3.3 Proposed Framework for Detection and Identification of Network Attacks

This section describes the proposed technique to detect synflood, fraggle and port scan, according to Figure 3.7, which represents the overview of the proposed framework for detection and identification of network attacks. In Subsection 3.3.1 we present the steps for extraction of the largest eigenvelue for each q-th time frame. Next, in Subsection 3.3.2, we show how to apply the eiganvalues on the MOS scheme in order to detect the attack. In Subsection 3.3.3, we present the eigenvalue analysis to identify the time frames detected as under attack, and the Subsection 3.3.4 describes the similarity analysis evaluated for detailed attack identification.

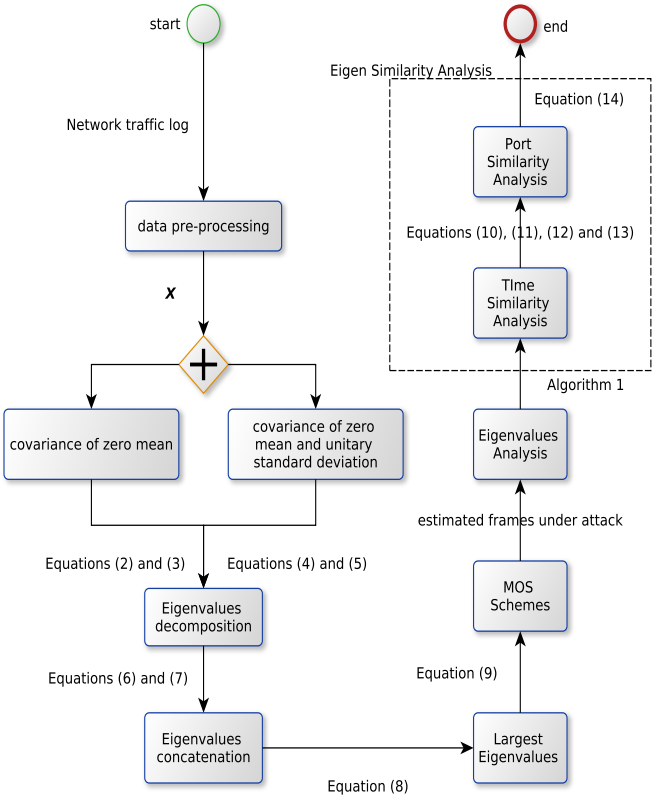

Figure 3.7 Overview of The Framework for Detection and Identification of Network Attacks.

3.3.1 Largest Eigenvalue by Time Frames

The proposed attack detection algorithm starts by the data pre-processing of a network traffic log containing IP, ports and timestamp of senders and receivers. During this step, the desired information is extracted in order to classify and count packets according to the origin and destination ports, and subsequently this information is grouped by minutes and by time frames.

With the data grouped into Q time frames, the framework considers the time variations of the matrix $X^{(q)} \in \mathbb{R}^{M \times N}$, with $q = 1, \ldots, Q$, in order to detect the attack.

According to flood and port scan attacks' behavior, flood attacks and port scan attacks can be characterized as covariance aware attack [13] and correlation aware attack [14], respectively. These characteristics are substantiated by the results obtained through the analysis based on sample covariation of zero mean variables and on covariance of zero mean and unitary standard deviation variables, described in Section 3.4, which shows that the main components of flood attacks are dominated by the variables with more variance and that the traffic associated with port scan attack does not generate many logs, however, it presents high covariance of zero mean and unitary standard deviation variables.

Therefore, to detect flood attacks, it is necessary to calculate the sample covariance matrix $\hat{R}_{yy}^{(q)}$ of the zero mean samples given by

$$y_m^{(q)} = x_m^{(q)} - \bar{x}_m^{(q)}. \tag{3.2}$$

The set of obtained vectors $y_m^{(q)}$ composes the zero mean matrix $Y^{(q)}$, then the sample covariance matrix $\hat{R}_{yy}^{(q)}$ can be calculated as follows

$$\hat{R}_{yy}^{(q)} = \frac{1}{N} Y^{(q)} Y^{(q)^{\mathrm{T}}}. \tag{3.3}$$

For the detection of the port scan attack, the main components are not dominated by the variables with large variance. Moreover, the portscan traffic presents a highly correlated network traffic. In order to exploit such structure, we compute the sample covariance $\hat{R}_{zz}^{(q)}$ whose variables have zero mean and unitary standard deviation as follows

$$z_m^{(q)} = \frac{x_m^{(q)} - \bar{x}_m^{(q)}}{\sigma_m^{(q)}}. \tag{3.4}$$

The set of vectors $z_m^{(q)}$ composes the matrix $Z^{(q)}$, then the sample covariance matrix

$\hat{R}_{zz}^{(q)}$ can be calculated via

$$\hat{R}_{zz}^{(q)} = \frac{1}{N} Z^{(q)} Z^{(q)^{\mathrm{T}}}. \tag{3.5}$$

Once the $\hat{R}_{yy}^{(q)}$ and $\hat{R}_{zz}^{(q)}$ have been obtained for flood and port scan attack detection, respectively, and since the next steps are the same for both sample covariance matrices, we refer to $\hat{R}_{yy}$ and $\hat{R}_{zz}$ as a matrix C. Therefore, the following step of the algorithm is the eigenvalue decomposition (EVD), calculated according to (4.7), in order to obtain the vector of eigenvalues $e^{(q)}$ associated with each matrix, according to (3.7).

$$C^{(q)} = V^{(q)} \Lambda^{(q)} V^{(q)^{\mathrm{T}}}, \tag{3.6}$$

$$e^{(q)} = \mathrm{diag}(\Lambda^{(\mathrm{q})}), \tag{3.7}$$

where the operator $\mathrm{diag}(\cdot)$ extracts the main diagonal of a matrix.

The eigenvalues should be sorted in descending order, i.e., $\lambda_1^{(q)} > \lambda_2^{(q)} > \lambda_3^{(q)} > ... > \lambda_m^{(q)}$. Therefore, the largest eigenvalue of the q-th time frame evaluated for the attack detect is given by $\lambda_1^{(q)}$.

The concatenation of the eigenvalues vector $e^{(q)}$ for $q = 1, \ldots, Q$ is represented by

$$E = \begin{bmatrix} \lambda_1^{(1)} & \lambda_1^{(2)} & \lambda_1^{(3)} & \cdots & \lambda_1^{(Q)} \\ \lambda_2^{(1)} & \lambda_2^{(2)} & \lambda_2^{(3)} & \cdots & \lambda_2^{(Q)} \\ \lambda_3^{(1)} & \lambda_3^{(2)} & \lambda_3^{(3)} & \cdots & \lambda_3^{(Q)} \\ \vdots & \vdots & \ddots & \vdots & \\ \lambda_m^{(1)} & \lambda_m^{(2)} & \lambda_m^{(3)} & \cdots & \lambda_m^{(Q)} \end{bmatrix}. \tag{3.8}$$

Note that since $\lambda_1^{(q)} > \lambda_2^{(q)} > \lambda_3^{(q)} > \cdots > \lambda_{m-1}^{(q)} > \lambda_m^{(q)}$, then the first line of the matrix E contains the largest eigenvalues of each q-th time frame, which is the Greatest Eigenvalue Time Vector (GETV) [21], denoted as

$$e_{\max} = E\{:, 1\} = [\lambda_1^{(1)}, \lambda_1^{(2)} ... \lambda_1^{(Q)}] \tag{3.9}$$

3.3.2 MOS Schemes

Traditionally the MOS schemes are applied for the eigenvalues of the vector $e^{(q)}$. However, the goal here is to detect the variations of the eigenvalues for different values of q.

Therefore, instead of using a certain q, the proposed approach applies MOS schemes for a vector of the largest eigenvalues of each q-th time frame, in order to identify variations and estimate the model order $\hat{d}$, which is the estimated number of time frames under attack. Therefore, $e_{\max}$ is sorted in descending order, producing $\sim e_{\max}$, that is used as input parameter for MOS schemes, according to $\hat{d} = \text{MOS}(\sim e_{\max})$. Note that some MOS schemes may also require the number of minutes that compose a time frame, as $\hat{d} = \text{MOS}(e_{\max}, Q)$. For more information about MOS, we refer to **??**.

In our previous work [21], the accuracy of AIC, MDL, EDC, RADOI, EFT and SURE schemes are evaluated for synflood and port scan attack detection, showing that EDC and EFT are effective for detecting this kind of attacks. The present work extends that evaluation to also analyze the effectiveness of the listed MOS schemes for fraggle attack detection, as shown in Section 3.4.

3.3.3 Eigenvalue Analysis

After applying the MOS schemes to the vector $\sim e_{\max}$, we obtain the estimate of the $\#A$. For instance, in the case of fraggle, synflood and portscan, if $\hat{d} = 1$, , then $\#A = 1$, which means that during the during the Q time frames one attack is present. However, if $\hat{d} = 0$, then $\#A = 0$, and this means that none of these attacks are present. Note that $\hat{d}$ can be greater than 1, indicating the presence of more than one attack.

In Subsection 3.3.2, we obtained only if $\hat{d} = 1$ or $\hat{d} = 0$. However, if $\hat{d} = 1$, the MOS schemes do not provide any information about the q-th time frame under attack. The identification of the q-th time frame under attack can be carried out through a eigenvalues analysis.

The largest eigenvalue analysis for estimating the q-th time frames that are under attack can be expressed according to Algorithm **??**, where $\hat{q}_{\max} \in \mathbb{R}^{\hat{\text{d}}}$ denotes a vector of the q-th time frames under attack, which is the q-th indexes corresponding to the $\hat{d}$ largest eigenvalues of $e_{\max}$. Algorithm **??** initially identifies the largest value of $e_{\max}$, according to Line 2 of Algorithm **??**, and its correspondent index, according to Lines 4 and 5 of Algorithm **??**. Subsequently, the largest value is removed of $e_{\max}$, according to Line 8 of Algorithm **??**, and a new iteration is performed until $e_{\max} = \{\}$.

After the estimation of the $\hat{q}_{\max}$ time frames under attack, it is necessary to obtain more details of the detected attacks, such as the n-th minutes when the attacks happened and the m-th network ports that were attacked. To deal with this problem, the adoption of

a similarity analysis between legitimate traffic and the traffic of time frames estimated as under attack is evaluated, analysing the effectiveness of cosine similarity to highlight abnormalies inserted by network traffic attacks.

3.3.4 Eigen Similarity Analysis

Cosine similarity calculates the cosine of the angle between two vectors, which represents the similarity of values between the selected vectors. Therefore, cosine similarity can be used to evaluate the variation of the most significant eigenvectors of $V^{(q)}$ against the the most significant eigenvectors of time frame detected as under attack, to analyze similarity changes into the most significant eigenvectors caused by the insertion of anomalous traffic [15].

This subsection describes the proposed eigen similarity analysis for detailed attack identification, in complement to the attack estimation carried out through MOS schemes and eigenvalue analysis. In Subsection 3.3.4 we present the eigen similarity analysis for identification of time under attack. Next, in Subsection 3.3.4, we show how to apply the eigen similarity analysis in order to identify network ports under attack.

Time Similarity Analysis

For eigen similarity analyis, we evaluate the cosine similarity in order to identify lacks of similarity between legitimate and malicious traffic, as follows

$$s_n = \frac{|v^{(q)} \cdot v_{(n)}|}{\|v^{(q)}\| \|v_{(n)}\|}, \tag{3.10}$$

where s_n denotes the absolute similarity degree of the n-th minute, $v^{(q)}$ is the most significant eigenvectors of a selected set of minutes without network attack, and $v_{(n)}$ is the most significant eigenvectors obtained after append the target n-th minute of traffic to be performed the flood and port scan attack identification.

The most significant eigenvector $v^{(q)}$, of a time frame q without attack, can be derivated from (4.7) and selected according to the eigenvector of the largest eigenvalue $\lambda_1^{(q)}$, which is the principal component of the evaluated matrix. The same calculation shall be performed in order to obtain the target eigenvectors $v_{(n)}$, calculated from a time frame without attack plus minutes of a time frame estimated as under attack, to evaluate the occurrence of network attacks.

The reference eigenvectors $v^{(q)}$ is calculated from the traffic whithout attack, from a

time frame q composed of Q minutes of legitimate network traffic. For the detailed attack identification, each $x_{(n)}^{(\hat{q})}$ vector of each n-th minutes of the estimated $\hat{q}_{\max}$ time frames shall be individually appended into $X^{(q)}$, as represented by

$$X_n = \{X^{(q)} | x_{(n)}^{(\hat{q})}\}. \tag{3.11}$$

The resultant $X_{(n)}$ is necessary to obtain $v_{(n)}$, through (4.7), for calculating the similarity degree s_n, ranging from 0 to 1, for each n-th minute. The s_n denotes the absolute similarity degree of the n-th minute in comparison to a well-known traffic without attack, detected through MOS schemes and eigenvalue analysis.

The incremental approach for similarity analysis is based on the incremental appending of network traffic into $X^{(q)}$, where the first evaluation is based on (3.11) and the subsequent evaluations is based on (3.12), incrementally appending each n-th minute until $n = N$.

$$X_n = \{X_n | x_{(n)}^{(\hat{q})}\}, \tag{3.12}$$

Figure 3.8 ilustrates the network traffic selection for the incremental approach of eigen similarity analysis, where the $X^{(1)}$ is chosen as reference for similarity analysis of the m-th minutes of the time frame $q = 3$, where one network attack was previously detected.

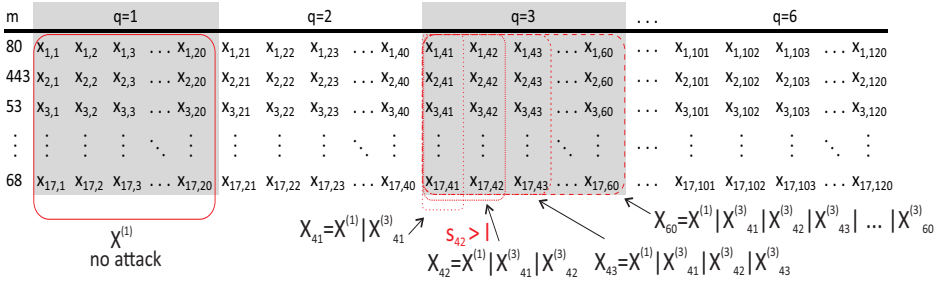

Figure 3.8 Traffic selection for incremental approach.

The eigen similarity analysis starts at $x_{(41)}^{(3)}$ and is incrementally performed until $x_{(60)}^{(\hat{3})}$, in order to calculate the s_n. We assume that $s_n < l$ means an attack identification, according the anomaly on similiarity of s_n to a defined limiar l. Therefore, after obtaining the most significant eigenvector $v^{(q)}$ and the target eigenvectors $v_{(n)}$ for eigen similarity analyis, the s_n is calculated according to (3.10).

If $s_n = 1$, then the two eigenvectors are completely similar and no anomaly is detected.

Smaller values of s_n mean less similarity and can indicate an anomaly, according to a defined threshold l, assuming that if $s_n < l$, then a network attack is identified during the n-th minute. Therefore, the s_n of each n-th minute shall be compared with the threshold l to evaluate if a attack is identified, according to

$$\hat{n}_{(n)} = \begin{cases} 1, & \text{if } s_n < l \\ 0, & \text{otherwise} \end{cases} \tag{3.13}$$

where $\hat{n}_{(n)}$ denotes a vector of n-th minutes detected as under attack.

The eigen similarity analysis can also be applied in an individual fashion, where each n-th minute must be individually appended into $X^{(q)}$, as shown by Figure 3.9.

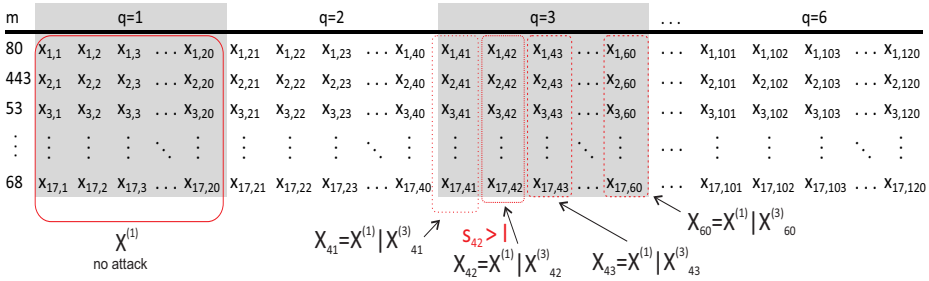

Figure 3.9 Traffic selection for individual approach.

The incremental and the individual approaches can be combined to obtain the incremental individualized approach, where each minute is incrementally appended into the selected $X^{(q)}$ for obtaining $v_{(n)}$ to similarity analysis of the n-th minute, until detect the first n-th minute under attack. Subsequently, X_n becomes the new reference of traffic without network attack and each subsequent minute must have its similarity individually evaluated, as shown in Figure 3.10.

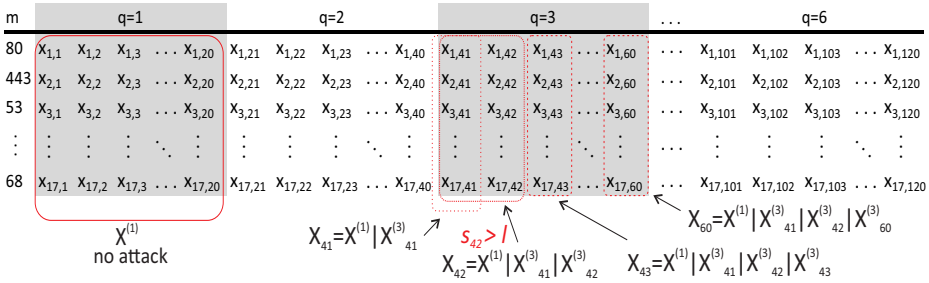

Figure 3.10 Traffic selection for incremental individualized approach.

This approach of incremental similarity analysis followed by individual analysis after

an attack detection allows to identify the attack period, highlighting the first and last time under attack. This identification is possible due to the variation of the most significant eigenvectors, which becomes more significant when compared a traffic under attack against a traffic with no attack, according to results which are discussed in Section 3.4.

Port Similarity Analysis

Given $\hat{n}$, which is the set of n-th minutes under attack, it is still necessary to obtain more details about the identified network attack, such as the network ports that are attacked during each n-th minute identified as under attack. Hence, it is also applied the cosine similarity analysis to identify variation of the most significant eigenvectors, caused by the insertion of anomalous network traffic by a selected m-th port during a n-th minute.

For detection of ports under attack, the $v^{(q)}$ last most significant eigenvectors without attack shall be used as reference for similarity analysis against the $v_{(n)}$ identified as under attack, individually evaluating the cosine similarity of each m-th port of all $\hat{n}$ minutes.

Therefore, $v^{(q)}$ should be calculated from the last $X^{(q)}$ time frame without attack, and $v_{(m,\hat{n})}$ should be calculated from the same traffic appened of all n-th minutes until the identified minute under attack, denoted as X_n.

For similarity analyis, each m-th port of the last n-th minute of X_n, denoted as $x_{(m,n)}$, shall be individually replaced by the traffic of the evaluated m-th port of the $\hat{n}$-th minute under attack, denoted as $x^{(\hat{q})}_{(m,\hat{n})}$, in order to identify significant variation on similarity caused by the traffic of the m-th port.

This approach for detection of ports under attack via similarity analysis is given by

$$\begin{cases} x_{(m,n)} = x^{(\hat{q})}_{(m,\hat{n})} \\ \\ s_{m,\hat{n}} = \frac{|v^{(q)} \cdot v_{(m,\hat{n})}|}{\|v^{(q)}\| \|v_{(m,\hat{n})}\|}, \end{cases} \tag{3.14}$$

where $x^{(\hat{q})}_{(m,\hat{n})}$ denotes the m-th port of the selected n-th minute identified as under attack and $x_{(m,n)}$ denotes the m-th port of the last n-th minute of X_n, which is used to calculate the $v_{(m,\hat{n})}$ most significant eigenvectors that contains the traffic of the m-th port of the $\hat{n}$-th minute identified as under attack.

Once $v^{(q)}$ and $v_{(m,\hat{n})}$ are obtained, then the $s_{m,\hat{n}}$ similarity degree can be calculated in order to identify if the traffic replacement highlights the adition of anomalous traffic by the evaluated m-th port during the $\hat{n}$-th minute previously identified as under attack.

This procedure should be repeated for each m-th target port of $\hat{n}$, in order to individu-

ally identify the network ports under attack during each $\hat{q}$-th time frame.

3.4 Experiments and Results

This section presents the performed experiments and the acquired results. First, in Section 3.4.1, the experimental scenario adopted in the experiments is summarized. Then, Section 3.4.2 shows the results of the largest eigenvalue analysis by time frames for the experimental scenario. In Section 3.4.3 are described the results of the evaluated MOS schemes for attack detection in the simulated dataset. Section 3.4.4 presents the results of the eigenvalue analysis for identification of time frames under attack, Section 3.4.5 shows the results of similarity analysis for detailed flood and port scan identification for the experimental scenario. Section 3.4.6 presents the results of the largest eigenvalue analysis, model order selection and the eigenvalue analysis for flood and probe attack detection in the DARPA 1998 dataset.

3.4.1 Experimental Scenario

The experiment time is 120 minutes, separated into six time frames, with each time frame corresponding to twenty minutes. Therefore, as the time of each sampling period is one minute, then $N = 20$. For each time frame q, a traffic matrix $X^{(q)} \in \mathbb{R}^{17\times 20}$ was obtained, as well as a covariance $\hat{R}_{yy}^{(q)} \in \mathbb{R}^{17\times 17}$ (calculated via (4.4)) and a sample covariance matrix $\hat{R}_{zz}^{(q)} \in \mathbb{R}^{17x17}$, assuming that $q = 1,2,3,4,5$ and 6.

The simulation started at 21:00h, the first time frame was from 21:00h until 21:20h ($q = 1$), the second was from 21:20h until 21:40h ($q = 2$), the third was from 21:40h to 22:00h ($q = 3$), the fourth was from 22:00h until 22:20h ($q = 4$), the fifth was from 22:20h until 22:40h ($q = 5$), and finally, the sixth was from 22:40h until 23.00h ($q = 6$). During the simulation, the victim made legitimate access, and the attacker performed the following attacks: at 21:54h ($q = 3$) was performed a port scan, at the interval ranging from 22:10h to 22:20h ($q = 4$) a synflood attack was simulated, and at the interval from 22:30h to 22:40h ($q = 5$) a fraggle attack was performed.

3.4.2 Largest Eigenvalues Analysis

For the evaluation of MOS Schemes accuracy for flood and port scan detection, the framework defines that it is necessary to obtain the largest eigenvalue of each time frame,

through eigen decomposition from a covariance of zero mean variables or covariance matrix of zero mean and unitary standard deviation variables, calculated from the evaluated traffic, as described in Section 3.3. Through eigenvalue analysis of traffic with flood or port scan attacks, it is possible to visualize a significant difference between the largest eigenvalues and the remain eigenvalues, which can indicate a relationship between an outlier and time frames under attack.

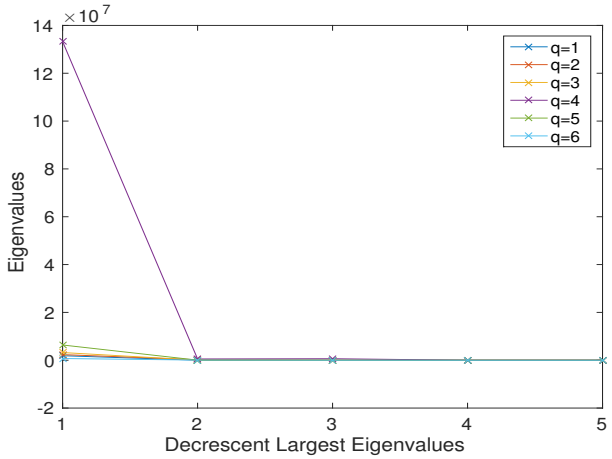

Figure 3.11 Eigenvalues of the sample covariance matrix (synflood).

Figure 3.11 depicts the eigenvalues calculated from sample covariance matrix of the network traffic used to evaluate the synflood attack identification. In Figure 3.11, the largest eigenvalue related to the simulated synflood attack ($q = 4$) stands out significantly from the other eigenvalues.

Figure 3.12 illustrates the eigenvalues calculated from sample covariance matrix of the matrix used for fraggle attack detection. In Figure 3.11, the largest eigenvalue related to the simulated synflood attack ($q = 5$) stands out significantly from the other eigenvalues, in accordance with the result shown in Figure 3.11 for the synflood attack analysis.

Figure 3.13 depicts the eigenvalues calculated from covariance matrix of zero mean and unitary standard deviation variables, of the network traffic matrix evaluated for port scan detection.

As analyzed for the synflood and fraggle attacks, note that the largest eigenvalue, related to this attack ($q = 3$), stands out significantly from the others eigenvalues.

Table 3.1 presents the values of the largest eigenvalues of each time frame q-th for port scan, synflood and fraggle detection.

In Table 3.1, note the significant variation of the eigenvalues associated with attacks,

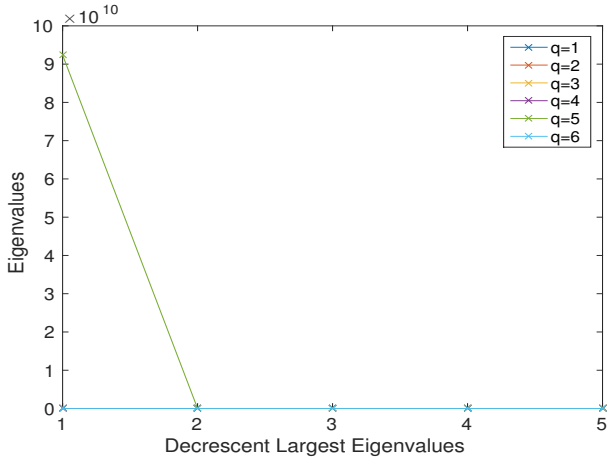

Figure 3.12 Eigenvalues of the sample covariance matrix (fraggle).

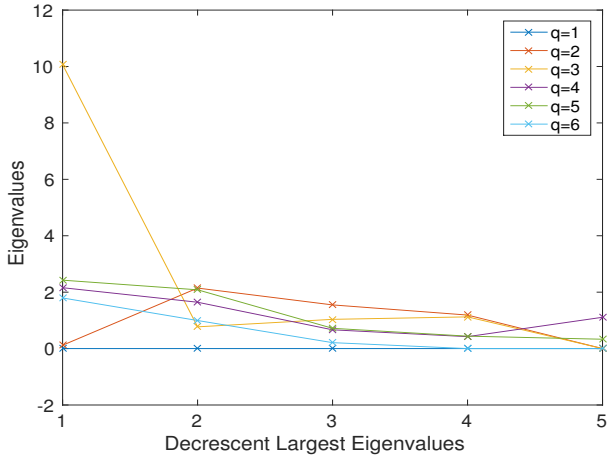

Figure 3.13 Eigenvalues of the covariance matrix of zero mean and unitary standard deviation (port scan).

Table 3.1 Largest Eigenvalue related to attacks detection

Time Frame q	**Vectors GETV**			
	Detection of ***synflood/fraggle***	**Detection of** ***synflood***	**Detection of** ***fraggle***	**Detection of** ***port scan***
1	1887545	1887545	1887545	2,0734
2	2341327	2341327	2341327	2,1451
3	3213867	3213867	3213867	10,0718
4	133238294	133238294	731229	2,1620
5	92384021611	6367983	92384021611	2,4253
6	708335	708335	708335	1,7948

in comparison to the others. At $q = 4$, where the synflood attack occurred, the maximum eigenvalue obtained is approximately 21 times larger than the second one. At $q = 5$,

where the fraggle attack occurred, the maximum eigenvalue obtained is about 29,000 times larger than the second one. At $q = 3$, where the port scan attack occurred, the maximum eigenvalue obtained is approximately 4 times larger than the second one. In the last case, for port scan attack detection, although the largest eigenvalue presented no too large variance to the second one, if compared to synflood or fraggle attacks, it clearly deviates from the remaining largest eigenvalues.

These results highlight that all q-th time frames where a network attack was simulated, present high significant variance between the largest eigenvalue and the remaining eigenvalues, obtained from sample covariance matrix, for flood detection, or from covariance matrix of zero mean and unitary standard deviation variables, for port scan detection. Therefore, we propose to apply the vector of the largest eigenvalues to MOS schemes in order to evaluate their accuracy for identification of time frames under attack, motivated by the fact that it is relevant to apply MOS schemes to automate the attack detection process, taking into account the characteristics of the evaluated eigenvalues.

3.4.3 MOS Schemes Evaluation

In [21], we evaluate the accuracy of AIC, MDL, EDC, RADOI, EFT and SURE MOS schemes [5, 21] for synflood and port scan attack detection. In this work we extend that evaluation for fraggle attack detection, applying the same schemes to fraggle attack detection over the traffic presented in Section 3.2, as results shown in Table 3.2.

Table 3.2 MOS schemes applied to port scan and flood detection

Type of analysis q	**MOS schemes (estimated model order** $\hat{d}$**)**						**(d)**
	AIC	**MDL**	**EDC**	**RADOI**	**EFT**	**SURE**	
Detection of synflood (presence of attack)	2	1	**1**	5	**1**	4	**1**
Detection of synflood (absence of attack)	1	1	**0**	1	**0**	3	**0**
Detection of fraggle (presence of attack)	1	1	**1**	5	**1**	4	**1**
Detection of fraggle (absence of attack)	1	1	**0**	1	**0**	3	**0**
Detection of port scan (presence of attack)	1	1	**1**	1	**1**	9	**1**
Detection of port scan (absence of attack)	0	0	**0**	1	**0**	1	**0**
Detection of synflood/fraggle (presence of attack)	2	2	**2**	5	**2**	5	**2**
Detection of synflood/fraggle (absence of attack)	1	1	**0**	1	**0**	3	**0**

Note that $\hat{d} = 1$, if there is attack, while $\hat{d} > 1$ indicates more than one attack. An

example of this could be seen for attack detection via EFT for traffic containing synflood and fraggle attacks, showing $\hat{d} = 2$, which indicates the presence of two attacks, as expected by the d real values of Table 3.2.

In Table 3.2, two MOS schemes outperforms from the others, EDC and EFT. Efficient Detection Criterion (EDC) and Exponential Fitting Test (EFT) are the most effective schemes, correctly estimating the number of attacks in comparison to the expected values for effetctive attack detection, as defined by the column of real values in Table 3.2. The AIC and MDL schemes are satisfactory only for port scan detection, however SURE and RADOI schemes did not show effective results for port scan or flood detection.

Although EDC and EFT presented the same accuracy on the evaluation, the EDC scheme requires less processing time than EFT, which is an important criteria to select EDC as the MOS scheme for flood and port scan detection on the remain experiments.

According to Table 3.2, EDC and EFT estimated correctly the number of attacks of a time frame vector, indicating that ocurred $\hat{d}$ network attacks, but not providing additional details, what highlights the necessity of complementary approaches in order to estimate the time and ports under attack. Hence, we propose apply eigen analysis to estimate the q-th time frames under attack and eigen similarity analysis to estimate the minutes and ports under attack.

3.4.4 Eigenvalue Analysis

According to the results presented in Section 3.4.2, the largest eigenvalue stands out significantly from the others eigenvalues of an evaluated q-th time frame. This behavior can also be observed in the largest eigenvalues analysis, according to results presented in Table 3.1, where it is possible to observe that the $\hat{d}$ largest eigen values of the time frames under attacks stand out significantly from the others largest eigenvalues.

Therefore, we conclude that the $\hat{d}$ largest eigenvalues correspond to the respectives q-th time frames under attack, which is denoted by $\hat{q}_{\max}$ and can be calculated according to Algorithm **??**.

3.4.5 Eigen Similarity Analysis

This paper proposes applying eigen similarity analysis to detect time and ports under attack, from each q-th time frames under attack defined by $\hat{q}_{\max}$. Hence, the proposed framework is applied to the time frames where $q = 3$, $q = 4$ and $q = 5$ to respectively evaluate its effectiveness for port scan, synflood and fraggle attack detection.

Time Analysis

Three approaches were evaluated for eigen similarity analysis: incremental, individual and incremental individualized approaches. For the incremental individualized approach, each minute is incrementally appended into the selected $X^{(q)}$ for obtaining $v_{(n)}$ to similarity analysis of the n-th minute, until detect the first n-th minute under attack. Subsequently, X_n became the new reference of traffic without network attack and each subsequent minute must have its similarity individually evaluated. For the incremental approach, each n-th minute must be incrementally appended into $X^{(q)}$, for obtaining the next eigenvectors $v_{(n)}$ for individual time similarity analyis. For the individual approach, each n-th minute must be individually appended into $X^{(q)}$, without incremental append, but doing individual appended into $X^{(q)}$ for obtaining the next eigenvectors $v_{(n)}$ for individual similarity analyis.

Table 3.3 presents the results of the evaluation of three approaches for similarity analysis of eigenvectors for port scan detection.

Table 3.3 Eigen Similarity Analysis for Port Scan Detection

Time Frame q	**Time** n	**Similarity Analysis**			**Attack?**
		Incremental Individualized	**Incremental**	**Individual**	
3	1	0.9946	0.9946	0.9946	no
3	2	0.9934	0.9934	0.9999	no
3	3	0.9912	0.9912	0.9999	no
3	4	0.9888	0.9888	0.9999	no
3	5	0.9856	0.9856	0.9998	no
3	6	0.9840	0.9840	0.9999	no
3	7	0.9824	0.9824	1.0000	no
3	8	0.9794	0.9794	0.9999	no
3	9	0.9673	0.9673	0.9926	no
3	10	0.9674	0.9674	0.9997	no
3	11	0.9733	0.9733	0.9993	no
3	12	0.9702	0.9702	0.9993	no
3	13	0.9677	0.9677	0.9999	no
3	14	0.9646	0.9646	0.9998	no
3	15	0.0216	0.0216	0.0276	yes
3	16	0.9621	0.0209	1.0000	no
3	17	0.9611	0.0199	0.9998	no
3	18	0.9612	0.0191	0.9999	no
3	19	0.9613	0.0186	0.9998	no
3	20	0.9638	0.0190	1.0000	no

Table 3.3 shows the evaluation of the time frame $q = 3$, when the port scan attack was simulated, considering the incremental individualized, incremental and individual approaches for eigen similarity analysis. According to the presented results, it is possible

to observe the high similarity between network traffic without attack, which was larger than 0.9610 for all evaluated cases, and emphasize the expressive low similarity when evaluated the traffic with the simulated port scan attack ($n = 15$), which was lower than 0.0276 for all evaluated approaches.

Comparing the approaches for similarity analysis, it is possible to observe that all approaches highlight the low similarity when evaluated the traffic under attack. However, the incremental approach figured out low similarity for times without attack, where $n = 16, 17, 18, 19, 20$, what indicates that the incremental approach can produce false posive results. This behavior occurs because the incremental approaches appends all selected traffic into the reference traffic for comparison against the original reference traffic, what makes more evident the first lack of similarity but reduces the changing detection capability after an attack detection.

Table 3.4 presents the results of the evaluation of the similarity analysis of eigenvectors for synflood detection. It shows the evaluation of the time frame $q = 4$, when the synflood attack is simulated, considering the incremental individualized, incremental and individual approaches for eigen similarity analysis. According to the results, it is possible to observe the high similarity between network traffic without attack, which is larger than 0.9907 for all evaluated cases, and emphasize the expressive low similarity when evaluated the traffic with synflood attack (between $n = 11$ and $n = 20$), which is lower than 0.1244 for all evaluated approaches.

The incremental approach produces better results if compared with other evaluated approaches, with lower values and maximum of 0.0185 for times under attack, but this approach presents change detection limitation after the first outlier of similarity, as shown in Table 3.3 for port scan detection.

Comparing the incremental individualized and the individual approaches for eigen similarity analysis, it is possible to observe that the incremental individualized approach obtain lowest values for almost all cases, except for the time $n = 14$, where incremental individualized approach identified a larger similarity than the individual approach. The incremental individualized appends information about each evaluated traffic, therefore it incorporates traffic behaviors that can reduce the outlier capability detection, as ocurred for the time $n = 14$.

Table 3.5 presents the results of the eigen similarity analysis evaluation for fraggle detection.

For fraggle attack detection, the lack of similarity between legitimate and malicious traffic was more evident than for the evaluation of synflood and port scan detecion. This

Table 3.4 Eigen Similarity Analysis for Synflood Detection

Time Frame q	Time n	Similarity Analysis			Attack?
		Incremental Individualized	Incremental	Individual	
4	1	1.0000	1.0000	1.0000	no
4	2	0.9999	0.9999	1.0000	no
4	3	0.9997	0.9997	0.9999	no
4	4	0.9998	0.9998	1.0000	no
4	5	0.9965	0.9965	0.9908	no
4	6	0.9975	0.9975	1.0000	no
4	7	0.9977	0.9977	1.0000	no
4	8	0.9980	0.9980	1.0000	no
4	9	0.9987	0.9987	0.9999	no
4	10	0.9991	0.9991	1.0000	no
4	11	0.0085	0.0085	0.0284	yes
4	12	0.0162	0.0120	0.0343	yes
4	13	0.0248	0.0158	0.0427	yes
4	14	0.1243	0.0185	0.1041	yes
4	15	0.0082	0.0162	0.0103	yes
4	16	0.0404	0.0070	0.0580	yes
4	17	0.0397	0.0007	0.0573	yes
4	18	0.0408	0.0042	0.0584	yes
4	19	0.0408	0.0079	0.0584	yes
4	20	0.0477	0.0092	0.0757	yes

Table 3.5 Eigen Similarity Analysis for Fraggle Detection

Time Frame q	Time n	Similarity Analysis			Attack?
		Incremental Individualized	Incremental	Individual	
5	1	1.0000	1.0000	1.0000	no
5	2	0.9999	0.9999	1.0000	no
5	3	1.0000	1.0000	1.0000	no
5	4	0.9999	0.9999	1.0000	no
5	5	0.9993	0.9993	0.9997	no
5	6	0.9993	0.9993	0.9997	no
5	7	0.9994	0.9994	1.0000	no
5	8	0.9995	0.9995	1.0000	no
5	9	0.9995	0.9995	1.0000	no
5	10	0.9995	0.9995	1.0000	no
5	11	0.0031	0.0031	0.0021	yes
5	12	0.0019	0.0025	0.0009	yes
5	13	0.0030	0.0026	0.0020	yes
5	14	0.0030	0.0027	0.0020	yes
5	15	0.0030	0.0028	0.0020	yes
5	16	0.0012	0.0025	0.0002	yes
5	17	0.0030	0.0026	0.0020	yes
5	18	0.0030	0.0026	0.0020	yes
5	19	0.0030	0.0027	0.0020	yes
5	20	0.0069	0.0023	0.0083	yes

behavior can be explained by the number of packets generated through the fraggle attack simulation, that was significative larger than the number of packets generated during the synflood simulation. Considering the three approaches, the largest value for times under attack was 0.0083, while the shortest value for times without attacks was 0.9993.

Therefore, considering the evaluation for port scan, synflood and fraggle detection, the incremental approach can produce false positive results, while the individual and incremental individualized approaches produce quite similar results, even though the individual approach be more simple and require less memory and processing time.

These results highlight the capability of change detection based on similarity between legitimate and malicious traffic from flood or port scan attacks, endorsing the effectiveness and safety for adoption of threshold for attack detection through eigen similarity analysis.

Port Analysis

Given $\hat{N}$, which is the set of estimated n-th minutes under attack, it is possible to apply cosine similarity analysis to identify variation of the most significant eigenvectors, caused by the insertion of anomalous network traffic by a selected m-th port, during a n-th minute. Therefore, the incremental individualized and individual approaches of eigen similarity analysis were evaluated, for detection of ports under flood and port scan attacks, according to results presented in following tables. For this evaluation, the v last most significant eigenvectors without attack was used as reference for similarity analysis against each target port m-th.

Table 3.6 presents the results of the evaluation of eigen similarity analysis for detection of ports under port scan attack, showing only the time frame $q = 3$ and minute $n = 15$, due to the simulated port scan attack occurred only at this time, although the remain time frame has been completely evaluated and presented high similarity to the reference of traffic without network attack.

The incremental individualized approach presented more sensibility to anomaly detection than the individual approach, the former produced the identification of a low similarity of 0.0298 for almost all ports under attack, unless the port 21, although the simulation has attacked this port. The individual approach was not able to identify low similarity for ports under attack, resunting in values of 0.9997 for ports with anomalous traffic and 0.9999 for ports without network attack.

For the evaluation of the proposed approaches for identification of ports under synflood and fraggle attack, all minutes of each time frame, in which one attack location was estimated, were analyzed. Even though, due to space limitations, only the results of the

Table 3.6 Eigen Similarity Analysis for Detection of Ports Under Port Scan Attack (q=3 and n=15)

Port p	Approaches		Attack?
	Incremental Individualized	Individual	
80	0.9999	0.9999	no
443	0.9999	0.9999	no
53	0.9999	0.9999	no
21	0.9999	0.9997	yes
22	0.0298	0.9997	yes
23	0.0298	0.9997	yes
25	0.0298	0.9997	yes
110	0.0298	0.9997	yes
143	0.0298	0.9997	yes
161	0.0298	0.9997	yes
69	0.0298	0.9997	yes
123	0.0298	0.9997	yes
445	0.0298	0.9997	yes
600	0.9999	0.9999	no
19	0.9999	0.9999	no
67	0.9999	0.9999	no
68	0.9999	0.9999	no

firt minute where a low similarity was idenfified will be shown such as where $n = 11$. Nevertheless, the results obtained for the evaluation of traffic without attack presented high similarity to the reference traffic, with similarities close to 0.9999, and the evaluation of the other minutes under attack presented results quite similar to the resuts shown in the Tables 3.7 and 3.8.

Table 3.7 presents the results of the evaluation of eigen similarity analysis for detection of ports under synflood attack, showing only the time frame $q = 4$ and minute $n = 11$.

According to results presented in Table 3.7, both approches identifies low similarity for the traffic of port 600, which is the target port of the simulated synflood attack, but the incremental individualized approach identifies the lowest similarity and presents better sensibility to identification of synflood attack through eigen similarity analysis assisted by threshold definition.

Table 3.8 presents the results of the evaluation of eigen similarity analysis for detection of ports under fraggle attack, showing only the time frame $q = 5$ and minute $n = 11$.

The results for the avaluation of ports under fraggle attack, shown in Table 3.8, were similar to the results obtained for synflood analysis, with the identification of low similarity for traffic of the port under attack. Nevertheless, for fraggle analysis, the individual approach identified the lowest similarity, that is 0.0004 while the incremental individualized approach obtained a similarity of 0.0031.

Table 3.7 Eigen Similarity Analysis for Detection of Ports Under Synflood Attack (q=4 and n=11)

Port p	Approaches		Attack?
	Incremental Individualized	Individual	
80	1.0000	1.0000	no
443	1.0000	1.0000	no
53	1.0000	1.0000	no
21	1.0000	1.0000	nos
22	1.0000	1.0000	no
23	1.0000	1.0000	no
25	1.0000	1.0000	no
110	1.0000	1.0000	no
143	1.0000	1.0000	no
161	1.0000	1.0000	no
69	1.0000	1.0000	no
123	1.0000	1.0000	no
445	1.0000	1.0000	no
600	0.0077	0.0427	yes
19	1.0000	1.0000	no
67	1.0000	1.0000	no
68	1.0000	1.0000	no

Table 3.8 Eigen Similarity Analysis for Detection of Ports Under Fraggle Attack (q=5 and t=11)

Port p	Approaches		Attack?
	Incremental Individualized	Individual	
80	1.0000	1.0000	no
443	1.0000	1.0000	no
53	1.0000	1.0000	no
21	1.0000	1.0000	no
22	1.0000	1.0000	no
23	1.0000	1.0000	no
25	1.0000	1.0000	no
110	1.0000	1.0000	no
143	1.0000	1.0000	no
161	1.0000	1.0000	no
69	1.0000	1.0000	no
123	1.0000	1.0000	no
445	1.0000	1.0000	no
600	1.0000	1.0000	no
19	0.0031	0.0004	yes
67	1.0000	1.0000	no
68	1.0000	1.0000	no

The incremental individualized approach was able to detect low similarity for all evaluated scenarios and types of network attack, while the other approaches presented false positives or low sensibility to eigen similarity analyis for network attack detection. This approach is able to gradually and incrementally adapt to network traffic changing,

preserving the sensibility to identify outliers or anomalies by time or network port, and reducing the occurrence of false positives.

According to the shown significant lack of similarity between legitimate and malicious traffic, it is possible to adopt safe thresholds for flood and port scan detection through eigen similarity analysis.

3.4.6 DARPA Scenario

This subsection presents a summarized view of results obtained from the application of the proposed framework, focusing on the largest eigenvalue analysis, model order selection and the eigenvalue analysis, for flood and probe attack detection in the DARPA 1998 dataset. Since the proposed framework is detailed in Section 3.3 and in Subsections 3.4.1, 3.4.2, 3.4.3, 3.4.4 and 3.4.5, here the focus is on the parameter selection, dataset evaluation and results for flood and probe attack identification.

The DARPA dataset includes 7 weeks of sniffed traffic saved into raw packet data. The traffic and the labeled attacks are grouped by week and day, with information of the number and types of attacks per day, but also providing the start time for each labeled attack. For this evaluation, an evaluation per day was performed, considering the network traffic of 24 hours split into Q time frames of 60 minutes ($N = 60$) and aggregate by minute and by port number. For each time frame q, a traffic matrix $X^{(q)} \in \mathbb{R}^{17\times 20}$ is obtained, considering the ports 20, 21, 22, 23, 25, 79, 80, 88, 107, 109, 110, 113, 115, 143, 161, 389 and 443.

Since the proposed framework focus on flood and probe attack detection, only the attacks with behavior similar to flood or probe attack were evaluated. Initially all DoS and probe attacks were selected, but it was observed that the most cases of DoS focus on exploit system vulnerabilities instead of flooding attack, and most of probe attacks focus on ICMP intead of port scanning. Therefore, for evaluation of the proposed approach for flood and probe attack detection, it is necessary to select cases that implements flood or port scan behaviors. The following week-day-attack cases were selected:

1. week3-thursday-neptune;
2. week4-friday-portsweep;
3. week5-thursday-neptune;
4. week5-thursday-portsweep;

Table 3.9 Results of the attack detection evaluation

Solution	Attack Type	Metric	Result
Proposed Work	Flooding	True Positive	100.00 %
Proposed Work	Flooding	False Positive	60.00 %
Proposed Work	Flooding	Misclassification	50.00 %
Proposed Work	Probe	True Positive	76.92 %
Proposed Work	Probe	False Positive	18.52 %
Proposed Work	Probe	Misclassification	32.73 %
Callegari *et al* [24]	Flooding	True Positive	82.00 %
Callegari *et al* [24]	Flooding	False Positive	-
Callegari *et al* [24]	Flooding	Misclassification	-
Lu and Ghorbani [16]	Overall	True Positive	94.67 %
Lu and Ghorbani [16]	Overall	False Positive	-
Lu and Ghorbani [16]	Overall	Misclassification	-
Lu and Ghorbani [16]	Portsweep	True Positive	50.00 %
Lu and Ghorbani [16]	Portsweep	False Positive	-
Lu and Ghorbani [16]	Portsweep	Misclassification	-

5. week5-friday-portsweep;

6. week6-wednesday-neptune;

7. week6-thursday-neptune;

8. week7-wednesday-portsweep.

Table 3.9 presents the evaluated results for attack detection, considering rates of TP, FP [8] and misclassification, which is defined as $\frac{(FN+FP)}{(TP+FP+FN+TN)}$ [3]. The analysis based on sample covariation of zero mean variables is evaluated for flooding behavior of netpune attacks, obtaining rates of 100.00 % for true positive (TP) detection and 60.00 % for false positive (FP) detection from 30 time frames. The results also show 50.00 % of misclassification rate , which attempts to estimate the probability of disagreement between the true and predicted cases by dividing the sum of FN and FP by the total number of pairs observed. The result for FP and misclassification analysis is poor due to the legitimate traffic of DARPA dataset presents high number of packets per time from one source to one target, with no variation on IP source or target port. This observation corroborates with previous evaluations of the DARPA dataset that highlight issues regarding traffic redundancy.

The analysis based on covariance of zero mean and unitary standard deviation variables was evaluated for port scan attacks, including probe attacks and DoS attacks that

send few packets for several ports in order to exploit some vulnerability. The results show rates of 76.92 % for TP detection and 18.52 % for FP detection from 94 time frames. The observed misclassification rate for this scenario is 32.73 %. It was observed that all FN cases are probe attacks with a time delay between scanning one port and start scanning the next port, what can be called as sparse probe attacks. Cases with a delay of one minute or more were not detected by the proposed approach.

The performance of detection rate of flooding attacks is compared with the method proposed by Callegari *et al* [24]. This work is a statistical method, based on PCA, without training or learning methods, even though it relies on visual analysis for principal components selection. The best detection rate of [24] was 82.00 % for detection of synthetically added flood attacks, while this current proposal obtains 100.00 % of detection rate for detection of flood attacks of DARPA dataset.

Due to the lack of statistical techniques without training or learning methods for detection of probe attacks, this proposed approach is compared to the Lu and Ghorbani's [16] proposal, which is a network anomaly detection model based on signal processing techniques that uses DARPA dataset for evaluation. The results of [16] show the best detection rate of 94.67 % in terms of general attack instance detection, but shows a case case with 50.00 % of attack instance detection for the portsweep attack. The proposed approach presents 76.92 % of detection rate measured specifically for probe attacks, without the requirement of learning or training methods, in contrast to Lu and Ghorbani's [16] work.

3.5 Performance Evaluation

This section discusses the computational complexity and the performance evaluation of the proposed framework, focusing on the main steps, which are the eigenvalues decomposition (EVD), largest eigenvalues analysis, application of MOS scheme and eigen similarity analysis, according to Figure 3.7 and equations presented in Section 3.3.

3.5.1 Complexity Analysis

The EVD, calculated according to (4.7), requires the previous calculation of covariance matrix, according to Equations (4.3), (4.4), (4.5) and (4.6). The covariance matrix calculation is $O(M^2N)$ and the EVD is $O(N^3)$, where M denotes the number of network ports and N denotes the period time. Therefore, the computational complexity for all

steps for EVD can be represented as $O(M^2N+N^3)$ and yields an $O(N^3)$ upper bound on the worst-case running time for EVD.

EDC and EFT are the MOS schemes that presented accuracy on the evaluation for the network attack detection. The computational complexity evaluation for MOS focuses on EDC scheme, since EDC requires less processing time than EFT but presents the same accuracy for the evaluated scenario. EDC scheme is $O(Q\log Q+Q+Q\log Q)$ and its worst-case running time can be represented as $O(Q\log Q)$, where Q denotes the number of time frames.

The largest eigenvalue analysis is $O(\hat{d}Q)$, where $\hat{d}$ denotes the number of time frame under attack, according to Algorithm **??**. Subsequently, the eigen similarity analysis relies on EVD and cosine similarity analysis, which is $O(N^2)$, for $\hat{d}$ time frames, therefore the eigen similarity analysis is which is $O(\hat{d}(M^2N+N^3+N^2))$ and yields an $O(N^3)$ upper bound on the worst-case running time for eigen similarity analysis.

Therefore, the proposed framework is $O(N^3+Q\log Q+\hat{d}Q+N^3)$ and its worst-case running time is $O(N^3)$. The computational complexity of EVD is predominant in the framework, but the approach splits the data into time frames with period time N, which makes possible to limitate the growth of N even for evaluations of cases with total time larger than N, reducing the impact caused by the computational complexity of EVD.

3.5.2 Processing Time Analysis

For better understanding the scalability and impact of configurations of N, M and Q, the processing time required for different scenarios of parameter configurations and dataset were evaluated, measuring the processing time of:

1. Eigen analysis based on sample covariance of zero mean;
2. Eigen analysis based on sample covariance of zero mean and unitary standard deviation;
3. EDC MOS scheme;

The performance evaluation focus on the main steps, which are discussed in subsection 3.5.1 regarding the complexity analysis. The data modeling is also a time consuming step, however its processing can be optimized through distributed processing techniques, such as MapReduce, achieving high throughput for packet counting or even for deep packet inspection [22].

Table 3.10 Processing time of the main steps for anomaly detection

Traffic Time (hour)	Frame Size (min)	Num. Ports	1-time (ms)	2-time (ms)	3-time (ms)
16	10	17	0.7900	0.8100	0.0650
16	20	17	0.5250	0.5950	0.0100
16	60	17	0.9700	1.1400	0.0250
16	120	17	0.6050	0.6100	0.0050
16	60	34	1.2750	1.2200	0.0050
16	120	34	1.1200	1.1700	0.0050
20	10	17	2.7950	2.8950	1.1000
20	20	17	2.0700	2.0200	0.3500
20	60	17	1.0250	1.0450	0.0650
20	120	17	1.0000	1.0700	0.0350
20	60	34	2.9650	3.2100	0.0400
20	120	34	2.9950	3.1150	0.0200
22	10	17	4.7250	4.0850	1.4600
22	20	17	2.3200	2.6800	0.2450
22	60	17	1.0700	1.1200	0.0300
22	120	17	0.9900	1.0500	0.0250
22	60	34	3.0850	3.1250	0.0650
22	120	34	2.8100	2.9600	0.0250

The experiments were performed in a desktop computer with a Intel Core i7-4510U 2.00GHz and 16 GB of RAM, considering: variations on the network traffic time; the frame size denoted as N; the number of network ports denoted as M; the mean processing time for eigen analysis based on sample covariance of zero mean, denoted as 1-time; the mean processing time for eigen analysis based on sample covariance of zero mean and unitary standard deviation, denoted as 2-time; and the mean processing time for EDC MOS scheme, denoted as 3-time. The mean time calculations was obtained from 200 measurement repetitions, in order to obtain reliable values.

Table 3.10 presents the measured results. The experiment considered traffic time of 16, 20 and 22 hours, according to the selected traffic time per day available by the DARPA dataset. Note that the processing time increases according to the increment in traffic time, around 2 or 3 times for 1-time and 2-time, but the worst measured processing time is 4.7250 milliseconds.

According to Table 3.10, the processing time increases with the frame size N decreasing, therefore it is possible to evaluate the frame size that produces better identification rates and acceptable processing time. The number of ports evaluated during the proposed

scheme is also an important variable regarding processing time optimizations, since the significant increase of processing time observed between scenarios considering 17 or 34 ports, with growth between 7 % and 199 %.

3.6 Conclusion and Future Works

This paper models the network traffic as a signal processing formulation for applying to the framework for detection and identification of network attacks, which is based on eigenvalue analysis, model order selection (MOS) and eigen similarity analysis.

The proposed framework is evaluated and the experimental results show that synflood, fraggle and port scan attacks can be detected accurately and with great detail in an automatic and blind fashion, applying signal processing concepts for traffic modeling and through approaches based on MOS and eigen similarity analysis. The main contributions of this work were: the extension of an approach based on MOS combined with eigen analysis to blindly detect time frames under network attack; the proposal and evaluation of an eigen similarity based framework to identify details of network attacks, presenting accuracy of timely detection and identification of TCP/UDP ports under attack, as well as presenting acceptable complexity and performance regarding the processing time.

This paper evaluated the effectiveness of MOS schemes for fraggle attack detection, extending our previous work [21] and showing that the analysis of the largest eigenvalues by time frames can be applied to detect the number of port scanning, and flood attacks, but still requiring more information for detailed attack detection. Therefore, we proposed a novel approach for detailed network attack detection, based on eigen similarity analysis.

The incremental individualized approach of eigen similarity analysis, is able to detect low similarity for all evaluated scenarios and types of network attack, while the other approaches present false positives or low sensibility to eigen similarity analyis for network attack detection. Therefore, the incremental individualized approach is able to gradually and incrementally adapt to network traffic changing, preserving the sensibility to identify outliers or anomalies by time or network port, and reducing the occurrence of false positives.

According to the significant similarity difference between legitimate and malicious traffic, it is possible to adopt safe thresholds for flood and port scan detection through eigen similarity analysis.

Future research is directed to improvements for obtaining better false positive rates, as well as for make the proposed framework able to identify sparse probe attacks or

subtle behaviors, such as exfiltration or covert communication, considering the evaluation of a flow-based analysis and novel datasets. Distributed or parallel processing can also be evaluated to analyze the scalability and processing capacity for monitoring high thoughput network traffic. Additionally, future research can evaluate the application of the proposed approach to different attack types and domains, considering cases that are aware to behavioral analysis.

4

Offline Mode for Corporate Mobile Client Security Architecture

Life is really simple, but we insist on making it complicated.

—CONFUCIUS

4.1 Introduction

Cloud computing is a new rapidly evolving paradigm in the world of distributed networking and computation. The basic features of the cloud environment is providing the elastic, on-demand and secure services for the end-users. While the first two requirements are rather well conceptualized and supported by the majority of the cloud platforms in use, security is a serious concern of the cloud providers and governmental organizations as well as academia and research community [**?** **?** **?**]. Although for the small and medium-sized enterprises (SME) the cloud environment is often the most cost-effective and easily scalable solution, the security and privacy of the sensitive data in cloud platforms are also not fully conceptualized leading to obscure and incomplete security paradigms and solutions.

A common practice to provide a cloud solution with stable security is to use a specific type of cloud service, such as Cloud Access Security Broker (CASB) or Cloud Access Control (CAC). These services are specifically designed to bring security at a single access point and provide the coordination of the most important security measures. Gartner [**?**] estimates that such systems will be used by 85 % of companies by 2020. The reason for this is that the organization of security measures at a single control point allows to control and to monitor the level of cloud protection much more effectively. The basic features of

the CASB are: discovery of cloud services, encryption along with tokenization for better search properties, access control, Data Loss Prevention (DLP) services, authentication, and auditing and alerting services [**?**]. The protection of the confidential data, according to the standards of CASB deployment should be provided elsewhere, i.e. in transit and at the end-user, i.e. end-client, side [**?**].

Additional security issues and requirements have to be considered when mobile clients are actively used in corporate cloud environment [**?**]. Today more and more organizations and enterprises are functioning in the Bring-Your-Own-Device (BYOD) paradigm. The uncontrolled usage of the mobile devices represents a serious risk to the development of secure SME cloud platforms being the bottleneck of the corporate information security system (ISS). While the enterprise cloud infrastructure based on the web interface can be protected by powerful third-party services, such as CASB and CAC, the corporate mobile client is usually light-weighted and generally less protected.

The protection scheme used on a mobile device should be both computationally secure as well as resource-constrained due to battery power limitations [**?**]. Therefore, encrypting files and generating keys on a mobile device is not considered a good solution. On the other hand, the protection schemes with good computational qualities lack the security analysis in many cases [**?**]. The common practice is the shadow user activities monitoring [**?**]. However, the mobile device usage stays unprotected in all the proposed scenarios while in offline mode.

Suppose that a SME uses CASB to protect data at rest, i.e. while stored on the server-side, in transit, communicating with server, and, in use, while the mobile client is connected to the network. In this case, when the mobile client goes offline with the sensitive corporate data on board all powerful cloud-based tools cannot help and the mobile client has to secure itself with its own limited resources. Moreover, due to the resources constraint, there is a crucial difference in strategy of online and offline mode protection. For example, offline mode does not allow performing the extensive computation and encryption on the mobile device. Observing the above described landscape, this paper outlines the concept of the offline mobile cloud security.

This paper proposes a novel approach based on powerful cryptographic preventive methods, such as secret sharing (SSS) [**?**] and Attribute-based Encryption (ABE) [**?**] for securing mobile clients. Moreover, our proposed approach includes the user behavior analysis based on Model Order Selection (MOS) [21], in order to detect possible threats, and to reduce the risks and the harm of the most common threats, which are the expired user misusing password and the intruder attack. The key expiration period is safely

incorporated into the proposed system solution in order to enhance security. Additionally, the behavioral analysis can indicate well known malicious behaviors, their variations, as well as novel attacks, which present low or high variance in comparison to legitimate user behaviors. The main target of our proposed solution is to provide a maximum defense at the minimal resource cost.

The paper is structured as follows. Section 4.2 analyzes the most common security problems in the mobile cloud environment and the solutions for offline protection in the BYOD world. Section 4.3 outlines the mobile security architecture of the proposed solution. Section 4.4 presents the detailed scheme of the proposed solution to the problem of offline mobile client security. Section 4.5 explains the algorithms for the core security methods: AES, ABE, SSS and MOS. Section 4.6 discusses the security proofs including the outline of adversary model, the common threat scenarios, the data modeling, the complexity analysis and dicussion of the practical implementation results. Section 4.7 concludes the paper.

4.2 Related works

The increasing usage of BYOD demands more sophisticated data protection services compared to ordinary computing environment. A common practice is to provide additional contextual methods apart from authentication, DLP services, and encryption, which can be at rest, in transit and in use [**? ? ? ?**]. The contextual methods increase the security of the mobile client at a maximum level with minimum resource requirements. The most commonly used are:

1. Using geolocation of the device to trace its usage;
2. Setting up the expiration period of an app;
3. Setting up the expiration period of client pass pin;
4. Setting up the counter of failed tries;
5. Secured transfer politics between apps;
6. Restricting access to the corporate app;
7. Restricted or prohibited offline access;
8. Logging and auditing.

Preventing data leakage on the mobile device is a crucial security problem. Therefore, it is necessary to take additional control and protection measures for the confidential data on the mobile devices that leave the boundaries of the organization. Generally, the most sensitive and confidential data should not be permitted to be transferred to the mobile device. However, what if the SME need to allow their employees to work on such devices and even use them on the offline mode for the convenience and traffic reducing, or even for a particular characteristic of the mobile client or the business itself?

From the theoretical point of view of this problem, there are several surveys, whose common point is the mobile cloud as a rapidly developing paradigm that poses many security and complexity problems [**?** **?** **?** **?** **?** **?** **?** **?** **?** **?**]. Kulkarni and Khanai [**?**] discuss the most important threats related to Mobile Cloud Computing and argue that there is a need for a lightweight secure framework that provides security with minimum communication and processing overhead on mobile devices.

An analysis of the new models of mobile cloud computing and new ways of using data storage services is presented in [**?** **?** **?** **?**]. Commonly, the models and protection schemes concentrate on the encryption properties and either perform the computations on their own [**?** **?**] or use the cloud provider to off-load the cryptographic operations [**?** **?**]. Obviously, it is natural the mobile client cannot handle all operations securely without the assistance of a cloud provider, due to resources constraint and battery limitation.

The necessity to use schemes that function without putting load on a provider arises when it is desired to make the mobile client less dependent on the cloud provider, i.e. corporate client continues to provide the secure access to the sensitive data without connection to the network. As discussed in [**?**], all the currently known schemes of encryption, performing the computation, either use a cloud provider [**?**], a third party trusted agent [**?** **?**], a combination of both [**?**] or ad hoc approaches [**?**]. In some cases, they concentrate on computational complexity without taking care of user privacy and security [**?**]. Therefore, according to [**?** **?**], the state-of-the-art mobile cloud security models do not consider the problem of the offline security mode.

In many cases the industrial providers of the mobile security API avoid the problem by completely forbidding the offline access to the protected app, i.e. SAP Mocona, which operates as a secure client wrapping layer [**?**]. Due to various constraints such as traffic load, travelling and ease of access, the SME business procedure may require such access. To the best of our knowledge, the offline mode security problem has not yet been deployed, neither in academia nor in the industry [**?** **?** **?** **?**]. Therefore, the main concern of this proposal is the protection of the mobile client and its data in offline mode, when

the functions of data protection cannot be offloaded to a cloud or a trusted party.

4.3 The mobile security architecture

The approach proposed in this work describes and implements the complete lifecycle of the mobile client security infrastructure. The mobile security processes depend on the key expiry period, and are used to access the protected storage. Once the user keys expire, the user is requested to enter his valid credentials, i.e. PIN and password. The mobile client then sends the credentials to the server for verification. Once the new set of access keys is received, the user can access the protected files in the offline mode, without the access to the server. This means that no further communication with the server is needed until the key expires.

The core set of functions and protocols can be divided into three sets of operations as shown in Figure 4.1.

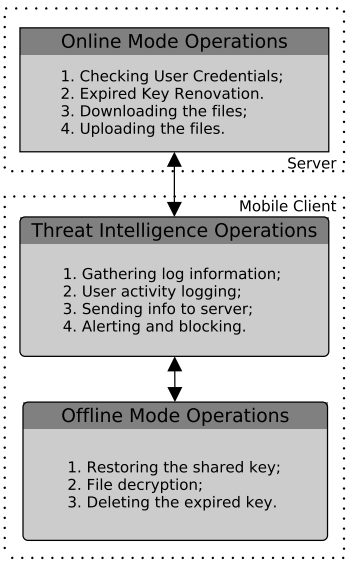

Figure 4.1 The core set of functions and protocols of the mobile cloud security infrastructure

Figure 4.1 describes the mobile client protection both in online and offline mode. In online mode, the client has the possibility to connect to the server and the security of the client is enhanced by the server-backed up mechanisms. On the other hand, in offline mode the client's security is supported by the standalone mechanisms. Additionally, the mobile client protection is enhanced by the threat intelligence unit providing the constant monitoring and analysis.

Figure 4.2 depicts the client-side protection mechanisms. The client should support 4 subsystems: i) encryption subsystem that contains the procedures of encryption and decryption; ii) storage subsystem that provides the downloaded shares and key storage protection; iii) threat intelligence unit that provides the constant monitoring, and iv) the communication subsystem. In short, all security procedures are connected to 4 groups of operations: file request and receiving; encryption and decryption; file and key storing; monitoring and analysis.

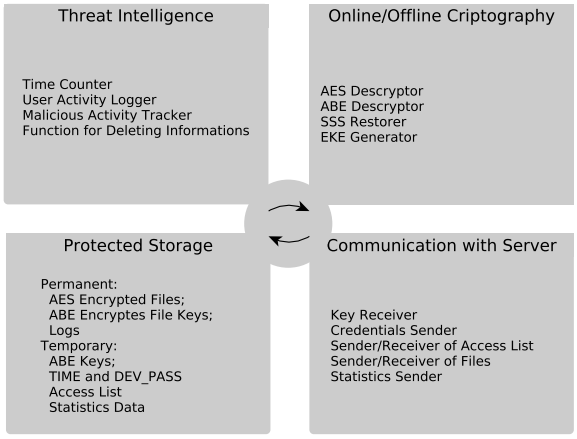

Figure 4.2 The Mobile Client Architecture

This architecture consists of the modules of cryptographic functions, threat intelligence infrastructure, communication with server and storage.

The cryptographic functions include the decryption (for AES and ABE) key restoration for SSS and EKE token generation procedure. While the AES-ABE hybrid encryption scheme functions both in the online and offline modes, the SSS is used only in the offline mode, while the EKE generation and executions is used in the online mode. These operations are described in Subsections 4.5.1-4.5.3 and are part of the Figure 4.1.

The threat intelligence infrastructure takes into account simple actors such as the time counter for the key expiry period, the counter of unsuccessful tries in order to protect from brute force attacks, and MOS-inspired statistics analyzer. Functions such as alerting and deleting the expired key belong to this block as well. These functions are described in Subsection 4.5.4.

The communication with server includes the separate sender and receiver to check the user credentials and receive new keys, which acts based on the EKE protocol. The remaining data like access list and files can be sent via unprotected channels.

4.4 The proposed solution for offline mobile security

This section proposes an approach for the mobile client protection in which the security is supported both in online and offline modes. Currently and to the best of our knowledge, the systems of mobile client protection follow a model where the protected client can operate only when it is connected to the cloud, which is not always convenient for the end-user. The basic principles of the mobile client protection herein proposed are:

1. Optimized communication with the cloud when the mobile client does not need to be constantly connected to the server due to the resource constraint and necessity to secure this communication;

2. Optimized combination of the security mechanisms so that the mobile client does not need to perform complex computation like encryption and key generation due to its resource constraint;

3. Behavioral analysis of user's operations on mobile client, which can indicate anomalous or automated activities performed by attackers.

The most important security issues in the proposed model arise when the client goes to the offline mode and the user is still allowed to get the access to the protected SME documents. In this case, the server can neither monitor the user activity nor provide the protection methods. The security should be performed at the mobile client. Additionally, the maximum protection should be provided at the minimum resource cost.

In the online mode the mobile client uses the secure communication with the server in order to verify the validity of user's credentials. On the contrary, the offline protection model should be approached independently. Thus, the proposal is that the authentication/authorization mechanisms in the offline mode should utilize the derived proof of the user identity. The requirements for the proof are as follows:

1. The proof is derived from the previous session, in order to verify that the user is still authorized;

2. The proof should not give access to user password, i.e. it should not be stored on the app;

3. The proof should be temporary and have an expiration period;

4. It should not be directly used in communication with the server, in order to prevent the malicious user from mimicking;

5. It should be resilient to offline dictionary attacks;

6. It has to stay effective both in the scenarios of the malicious outsider and leakage of information when the formerly authorized user leaves the group.

Such proof cannot be stored on the client, as it is not possible to guarantee its protection in the offline mode. Therefore, the most effective way to secure the proof on the offline mode without performing complex computation is to share this proof between the mobile client and the user in a protected manner. In this case, there is no need to store the function of the user password. The additional argument against the traditional password verification is the necessity to check the PIN, which is very small, so the construction of valid one-way function resilient to the offline dictionary attacks is difficult.

The proposed architecture for offline mobile security is represented in Figure 4.3.

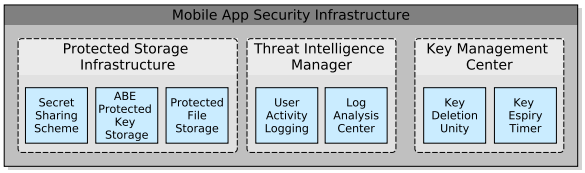

Figure 4.3 Proposed Architecture for Offline Mobile Security

The Figure 4.3 describes the proposed architecture for offline mobile security, showing the modules responsible for securing the mobile client, which includes:

1. **Protected Storage**: the storage is protected with the shared user key and contains the ABE keys giving access to the file keys which allow decrypting the stored files. The outline of the hybrid encryption scheme is presented in the Section 4.5.

2. **Threat Intelligence Manager (TIM)**: most attacks incur into significant variation on the legitimate behavior of information systems, or they adopt well-known patterns that can be easily detected by monitoring the system in the case of the offline mode. Signal processing techniques have been successfully applied to anomaly detection [**? ?**] and have become a solution to a problem of improving detection accuracy, adaptability and computational cost for application on resource-constrained scenarios. Therefore, signal processing can be applied in offline

mobile client security, for evaluating anomalies on user's behavior, according to the scenarios in Section 4.6. Moreover, Model Order Selection (MOS), which is an effective signal processing technique to separate noise components from the principal components, can be applied into anomaly and attack detection [21], to identify and separate malicious behaviors from the legitimate ones. The TIM is an internal module of the mobile client that implements offline anomaly detection through signal processing techniques.

3. **Key Management Center**: it includes the functions for maintaining the key expiry period and deleting the expired keys.

4.4.1 Offline mode workflow

The user performs the following operations in the offline mode:

1. Enters the PASS and the 4-digit PIN;
2. Views the list of authorized files kept on the mobile storage;
3. Open the protected encrypted files;
4. Modify the protected files and save them in an unprotected storage.

The mobile client performs all the cryptographic calculations in the shadow. These calculations include the key storage, key restoring, decryption and showing the decrypted files in the mobile client area. Note that the mobile client does not check the password or PIN validity as it does not hold the verification proof for the above. Also, the mobile client keeps track of the user activities and the key expiry. The complete list of the mobile client activities is presented in the Figure 4.4.

To finalize the description of the offline mode, Figure 4.5 shows the complete workflow of the proposed mobile application in the offline-mode.

The core cryptographic module of the implemented solution is based on the combination of AES-ABE-SSS methods, described in the Section 4.5. The key feature of the offline security is that the mobile client does not actually store any part of the user password to be verified. The mobile client combines its own key with the user share (PIN and password-derived) in order to restore the initial KEY_SET_KEY. If the user provides the wrong share the mobile client will not be able to recognize it, but will decrypt the files incorrectly. Additionally, the password entering is tracked and too many tries in a short time are considered a threat.

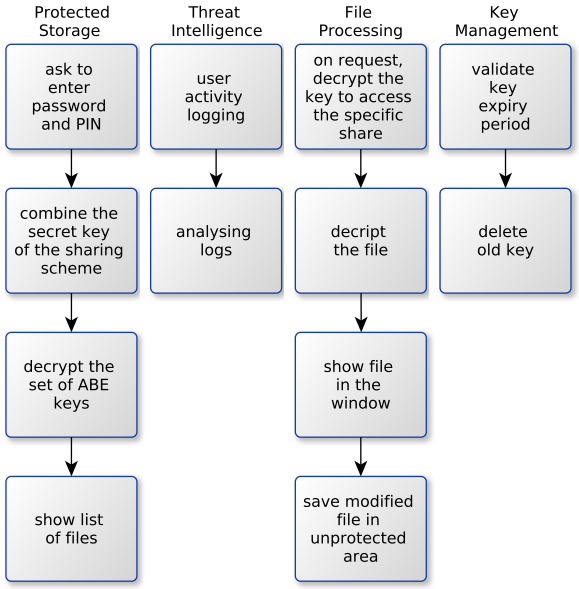

Figure 4.4 Offline Mode Operations

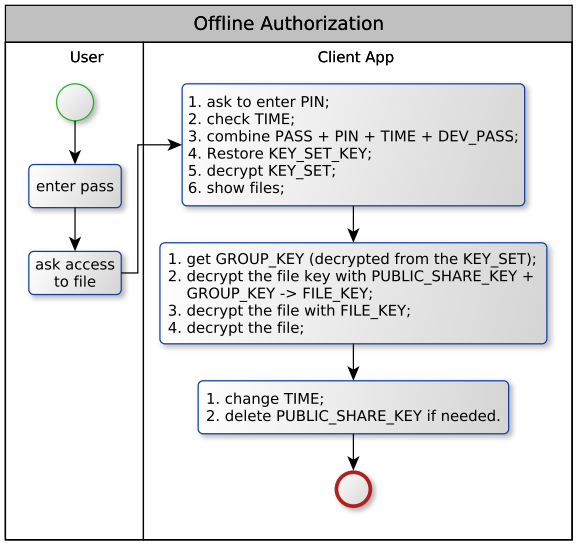

Figure 4.5 Offline mode authorization workflow

4.4.2 Offline Behavioral Analysis

In the proposed client security architecture, the Threat Intelligence Manager (TIM) is responsible for receiving logged user operations, perform feature extraction, data modeling and malicious behavior analysis in order to identify possible threats, in offline mode. Figure 4.6 depicts the TIM for offline behavioral analysis.

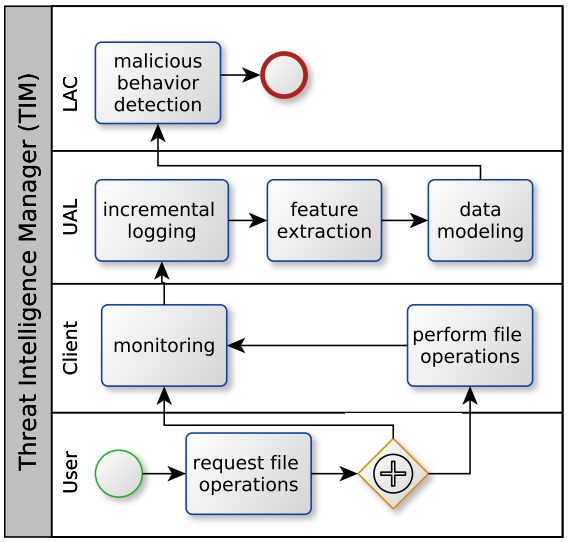

Figure 4.6 The Threat Intelligence Manager Workflow

As depicted in Figure 4.6, users request operations are logged so that the main features can be monitored in the mobile client. The user behavior, trying or effectively executing operations, shall be incrementally captured and logged, making possible to monitor the main features that can reveal malicious behaviors, as well as to identify unexpected behaviors that can reveal possible threats. Therefore, the user operations are monitored by the client app, which sends the information to the User Activity Logging (UAL).

The UAL is responsible for the incremental logging of activities of the mobile client, feature extraction and data modeling for malicious behavior detection, through the Log Analysis Center (LAC). As an internal module of the mobile client, the UAL implements monitors of selected events of the application, such as a login attempt or a file decryption, and logs the desired information for further analysis. The logged information shall be decomposed into selected features and modeled as matrices, composed of the number of occurrences of the selected features by its location and by time. The resultant data is submitted to the LAC, for anomaly detection.

The LAC performs the behavioral analysis through eigenvalue analysis and MOS schemes, which identify anomalies on sparse, subtle or abrupt number of user operations. The malicious behavior detection is detail described in 4.5.4.

4.5 The algorithms, key usage and data protection methods

In the proposed approach, the kernel encryption scheme in the mobile client is a combination of several methods of security. The files are encrypted with 128/256-bit AES, while the permanent file keys are encrypted with the attribute-based encryption. The set of expiring ABE keys corresponding to the set of files accessible by user in encrypted with a single expiring AES key (KEY_SET_KEY). This key is expiring and is splited by the server into 4 parts (2 are stored on the device and 2 belong to the user) by the method of secure secret sharing. The encryption workflow is outlined in the Figure 4.7.

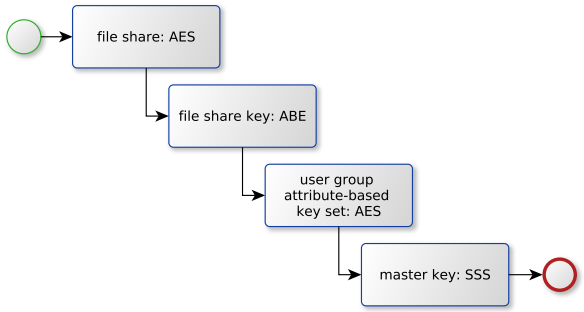

Figure 4.7 Encryption workflow

The key hierarchy is explained by Figure 4.8 where the key represented in Figure 4.7 serves for decrypting the key for each level represented in Figure 4.8. As it is possible to be observed in Figure 4.8, the master key on the upper level is computed by means of SSS (Subsection 4.5.3). It serves for decrypting the ABE key storage (section 4.5.2). The ABE key allows the user finally decrypt the permanent file share keys. Further, the file share can be decrypted by means of AES.

This proposed encryption scheme is called as a hybrid encryption since it uses a combination of different encryption methods in order to support all requirements to the encryption scheme used in the proposed infrastructure. Then it has:

1. AES provides a fast encryption of the data files;
2. ABE permits to support the authorization policy on the encryption level;
3. SSS allows protecting the key storage by avoiding the necessity to store the key proof.

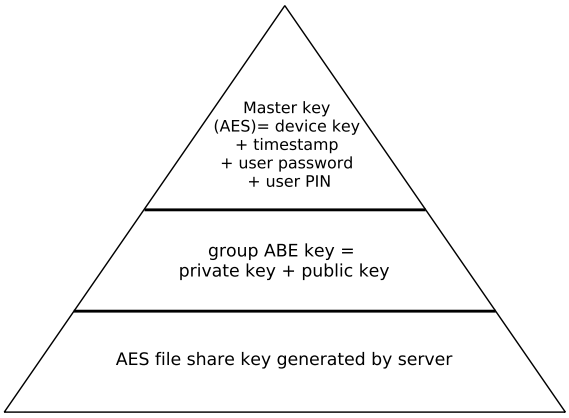

Figure 4.8 Encryption key hierarchy

In Subsections 4.5.1-4.5.3, a detailed description of the building blocks of the hybrid encryption is provided, and the proposed malicious behavior detection scheme is detail described in 4.5.4.

4.5.1 AES file encryption

The protected files are encrypted on the server side with the secure 128/256 bit AES encryption, which is currently an industrial standard. Nevertheless, it is possible to encrypt the file storage with Blowfish or Serpent which also provide a high level of safety [**?**].

The symmetric encryption on a server side is performed in two steps. First there is a single AES key for preserving the server storage. Second, when the encrypted file is sent from the server to the mobile client another unique AES key is generated, which serves for the encryption on client side. In other words, along with the encrypted file the user gets his own unique AES key for decrypting the files received. This key is protected by attribute-based public key encryption (ABE).

The scheme of the server-side AES encryption is represented in Figure 4.9.

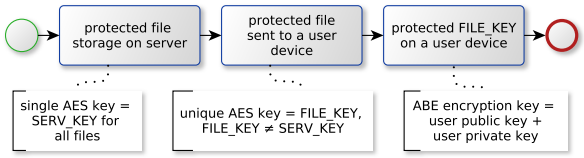

Figure 4.9 Server-side encryption

Usually, the AES encryption key is used as a session key. In our setting, it is not desirable to re-encrypt the files stored in the device unless there is a certain specific condition (for example, the user leaves the domain or a specific group in the domain (the group of users in the domain contains the users with the equal access rights, i.e. the same set of accessible file shares) and the file should not be accessible for this user anymore). This is why AES key in the presented notation is permanent and defined as FILE_KEY. The randomly generated FILE_KEY is unique for each file or set of files stored on a client.

4.5.2 ABE encryption to protect the FILE_KEY

With each user session, the permanent FILE_KEY (unique AES key) is re-encrypted. The set of FILE_KEYs is protected with the corresponding ABE keys. The unique ABE model that we propose supports the attribute policy based on user groups and on file shares, i.e. the key attributes correspond both to the groups and the file shares.

The model supports the simple selective ABE scheme [**?** **?**]. The selective scheme for attribute-based encryption is as follows: if at least one attribute in the set $\{t_i\}_U$ is equal to the attribute in the set $\{t_i\}_M$, the corresponding user U can decrypt the text M. In other words, as soon as the user and share have one attribute in common – the user can get access to the share. The components of the ABE encryption are:

1. **Master-key (MK)** which is kept safely on server and accessible only for the domain administrator and is given by $MK = (t_1, t_2, \cdots, t_n, y)$, where the values t_i are randomly selected from the huge group Zp. They are the private keys corresponding to the group attributes. Note, that this is different from the usual PK encryption: the private keys are controlled by the admin and not by the users.

2. **Public key (PK)** depends on the master key values and is kept in clear allowing users to access the information: $PK = (g^{t_1}, g^{t_2}, \cdots, g^{t_n}, e(g,g)^y)$. Here $e(g,g)$ is the bilinear pairing function corresponding to an elliptic curve. In order to explain the properties of bilinear paring function, let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_1$ and e be a bilinear map, e: $\mathbb{G}_1 X \mathbb{G}_1 \rightarrow \mathbb{G}_2$. The bilinear map e has two properties, the non-degeneracy, which defines that $e(g,g) \neq 1$, and the bilinearity, which defines that for all $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u,v)^{ab}$ [**?**]. Here $w \in \mathbb{Z}$ is the unique security parameter allowing to control the key generation for each user group.

3. Secret user **KEY_SET** depends on his attribute set. Here each D_i (**GROUP_KEY**) serves for decryption of the data of a single group of users, for example, related to some project: $\{t_i\}_U \rightarrow D = \{D_i = g^{\frac{yw}{t_i}}\}$.

4. Encrypted text M, in this context, $M = FILE\_KEY$, or the permanent AES symmetric key, which allows to avoid the file re-encryption. The encryption procedure is performed by a multiplication. The set of the public keys E_i (PUBLIC_SHARE_KEY) corresponding to the set of groups able to access the text is kept along with the encrypted text E: $E = Me(g,g)^{ys}, \{E_i = g^{\frac{t_i s}{w}}\}, \forall i, t_i \in \{t_i\}_U$. Here $s \in \mathbb{Z}$ is the unique security parameter allowing to control the key generation for each file share.

5. Decryption is division: $M = \frac{E}{Y^s}$.

In order to perform this operation the user needs the pair of private key D_i and public key E_i corresponding to the attribute t_i:

$$Y^s = e(g,g)^{ys} = e(E_i, D_i) = e(g^{\frac{yw}{t_i}}, g^{\frac{t_i s}{w}}) = e(g,g)^{ys}. \tag{4.1}$$

The result of decryption is the FILE_KEY - the symmetric AES key that permits to decrypt the contents of protected file.

4.5.3 Secret sharing scheme to protect the key storage

The attribute-based private keys D_i should be protected while being stored in the device memory. Therefore, server encrypts the set of D_i with a single AES key before sending it to the user device. This AES key (master key) is denoted by the value KEY_SET_KEY in our notation. KEY_SET_KEY is a secret value and it is split by the secure method of polynomial modular secret sharing [**?** **?**] into the set of 4 shares: KEY_SET_KEY = PASS + PIN + TIME + DEV_PASS.

The adversary cannot get any information of the KEY_SET_KEY unless he possesses all 4 key parts. Here the values PASS and PIN are predefined similar to the construction in [**?**].

The PASS and PIN values are predefined, similar to the key storing construction based on the polynomial secret sharing scheme (SSS) in [**?**].

In this SSS, the following notation is used: $S(x) \in F_2[x]$ denotes the secret polynomial with coefficients from the finite field F_2 to be computed by pooling the shares together. $s_1(x)$, $s_2(x) \in F_2[x]$ denote the shares of the SSS, that are used to compute the secret $s(x)$.

The proposed authentication system is based on the shared storing of the user key. Also, the mobile client acts as a dealer in the SSS. Using the SSS ensures that the key can only be accessed by an authenticated user. The participants of the (2, 2)-threshold SSS are the user and device. The user share $s_1(x)$ is computed based on the PIN and the PASS entered by the user. Additionally, the current time value TIME is used in the calculation of the share. Let $s(x) = d$ and $s_1(x) = f(PIN + PASS + TIME)$, where f is a one-way (hash) function that transforms the data into the string of the desired length:

$$\begin{cases} S = s \bmod p_0 \\ S = s_1 \bmod p_1 \end{cases} \tag{4.2}$$

According to the CRT:

$$S \equiv s_1 p_0 {p_0}^{-1}{}_{p1} + s_1 p_1 {p_1}^{-1}{}_{p0} + \bmod p_0 p_1 \tag{4.3}$$

Here, p_0 and $p_1 \in F_2[x]$ denote the public moduli of the polynomial SSS [10, 11] and $s \in F_2[x]$ denotes the intermediate secret value. The formulae for these moduli, secret and participants' shares directly follow from the definition of polynomial SSS construction.

Thus calculated DEV_PASS is written to the permanent device memory. The user share is not saved. Otherwise, it would allow an attacker to locally validate the restored private key.

4.5.4 MOS for Threat Intelligence

In the context of anomaly-based schemes for attack detection, the proposed behavioral analysis approach applies signal processing techniques, such as Principal Component Analysis and Model Order Selection schemes [21], for automatic detection of attacks or malicious behaviors.

Model Order Selection is an effective signal processing technique for several applications, allowing separating the only noise components from the principal components applying a rank reduction of the data.

Applying MOS to the analysis of user operations can be effective in order to reveal the occurrence of malicious behavior during an offline session. MOS for threat intelligence requires that the target features, which can be user operations, should be modeled as a matrix composed by the number of occurrences by location and time, and grouped into Q time frames. Therefore, the framework considers the time variations of the matrix $\mathbf{X}^{(q)} \in \mathbb{R}^{M \times N}$, with $q = 1, \ldots, Q$, in order to detect the occurrence of malicious behaviors.

For example, one element of $\mathbf{X}^{(q)}$ can represent the number of file readings on folder m during the minute n, from file operations logged by the mobile client.

The MOS schemes can rely on sample covariance of zero mean variables (called as zero mean covariance for the sake of simplicity) and sample covariance of zero mean and unitary standard deviation (called as zero mean and standardized covariance for the sake of simplicity) variables, where the former is useful to identify abnormalities caused by large amounts of operations during a period, while the latter is applied to identify anomalies on sparse or subtle number of file operations.

Classical approaches to model order selection require the computation of the sample covariance matrix $\hat{\mathbf{R}}_{yy}^{(q)}$ and of its eigenvalues, obtained from the measurement matrix $\mathbf{X}$ of the zero mean samples given by

$$\mathbf{y}_m^{(q)} = \mathbf{x}_m^{(q)} - \bar{\mathbf{x}}_m^{(q)}. \tag{4.4}$$

The set of obtained vectors $\mathbf{y}_m^{(q)}$ composes the zero mean matrix $\mathbf{Y}^{(q)}$, then the zero mean covariance matrix $\hat{\mathbf{R}}_{yy}^{(q)}$ can be estimated as follows

$$\hat{\mathbf{R}}_{yy}^{(q)} = \frac{1}{N}\mathbf{Y}^{(q)}\mathbf{Y}^{(q)^{\mathrm{T}}}, \tag{4.5}$$

where $\hat{\mathbf{R}}_{yy}^{(q)}$ means the estimation of the sample covariance matrix from the measured zero mean matrix $\mathbf{Y}^{(q)}$ over N minutes of the time frame q.

For MOS based on sample covariance of zero mean and unitary standard deviation, in order to identify anomalies with sparse or subtle behavior, it is required, for each variable, to make the standard deviation unitary as follows

$$\mathbf{z}_m^{(q)} = \frac{\mathbf{x}_m^{(q)} - \bar{\mathbf{x}}_m^{(q)}}{\boldsymbol{\sigma}_m^{(q)}}. \tag{4.6}$$

The set of vectors $\mathbf{z}_m^{(q)}$ composes the matrix $\mathbf{Z}^{(q)}$, then the zero mean and standardized covariance matrix $\hat{\mathbf{R}}_{zz}^{(q)}$ can be calculated via

$$\hat{\mathbf{R}}_{zz}^{(q)} = \frac{1}{N}\mathbf{Z}^{(q)}\mathbf{Z}^{(q)^{\mathrm{T}}}. \tag{4.7}$$

Once the $\hat{\mathbf{R}}_{yy}$ or $\hat{\mathbf{R}}_{zz}$ have been obtained for MOS in order to anomaly detection, for the sake of simplicity, we refer to $\hat{\mathbf{R}}_{yy}$ or $\hat{\mathbf{R}}_{zz}$ as a matrix $\mathbf{C}$. Therefore, the next step of the algorithm is the eigenvalue decomposition (EVD), calculated according to

$\mathbf{C}^{(q)} = \mathbf{V}^{(q)}\blacksquare^{(q)}\mathbf{V}^{(q)^{\mathrm{T}}}$, in order to obtain the vector of eigenvalues e, as following:

$$\mathbf{e}^{(q)} = \mathrm{diag}(\blacksquare^{(\mathrm{q})}), \tag{4.8}$$

The eigenvalues should be sorted in descending order, as defined by $\lambda_1^{(q)} > \lambda_2^{(q)} > \lambda_3^{(q)} > \cdots > \lambda_m^{(q)}$, to make possible the selection of the first eigenvalue in the obtained sequence, represented by $\lambda_1^{(q)}$, which is the largest eigenvalue of the data evaluated for attack detection.

The process of obtaining the $\mathbf{X}^{(q)} \in \mathbb{R}^{M\times N}$ and the matrix $\mathbf{C}^{(q)}$, finding the largest eigenvalue for each q-th time frame, should be repeated until $q = Q$, in order to obtain the largest eigenvalue of all time frames, as presented by

$$\mathbf{E} = \begin{bmatrix} \lambda_1^{(1)} & \lambda_1^{(2)} & \lambda_1^{(3)} & \cdots & \lambda_1^{(Q)} \\ \lambda_2^{(1)} & \lambda_2^{(2)} & \lambda_2^{(3)} & \cdots & \lambda_2^{(Q)} \\ \lambda_3^{(1)} & \lambda_3^{(2)} & \lambda_3^{(3)} & \cdots & \lambda_3^{(Q)} \\ \vdots & \vdots & \ddots & \vdots & \\ \lambda_m^{(1)} & \lambda_m^{(2)} & \lambda_m^{(3)} & \cdots & \lambda_m^{(Q)} \end{bmatrix}. \tag{4.9}$$

Since $\lambda_1^{(q)} > \lambda_2^{(q)} > \lambda_3^{(q)} > \cdots > \lambda_{m-1}^{(q)} > \lambda_m^{(q)}$, then the first line of the matrix $\mathbf{E}$ contains the largest eigenvalues of each q-th time frame, which is the expected input for MOS schemes and can be expressed as

$$e_{\max} = E\{:,1\} = [\lambda_1^{(1)}, \lambda_1^{(2)} \cdots \lambda_1^{(Q)}] \tag{4.10}$$

Once obtained the largest eigenvalues of each q-th time frame, it is possible to apply a selected MOS scheme to estimate the model order $\hat{d}$, which is the estimated number of time frames with malicious behavior. Therefore, $e_{\max}$ is used as input parameter for MOS schemes, according to the equation

$$\hat{d} = \mathrm{MOS}(e_{\max}) \tag{4.11}$$

Note that some MOS schemes may also require the amount of time that compose a time frame, such as $\hat{d} = \mathrm{MOS}(e_{\max}, M)$. For more information about MOS schemes, interested readers are referred to [5].

4.6 Results and analysis

This section provides the detailed analysis of the results the proposed approach regarding security and complexity analysis.

4.6.1 Security analysis

The security analysis of the proposed model was performed both from the point of implemented cryptographic mechanisms and from the user behavioral analysis. Two common attack scenarios were analyzed. First, the malicious outsider trying to infect or steal the important files. Second, the malicious expired user trying to steal the important files.

Adversary model

The proposal is to use a restricted adversary model. In other words, the actions of the adversary are predicted within the outlines model and scenarios. For example, it is not possible to guarantee the security of the decrypted document once the user copies its content to another file.

In the analyzed scenarios, it is assumed that the adversary can:

1. Steal the user keys one-by-one while the mobile client performs decryption, if the attacker is outsider;
2. Use expired key to decrypt the files, if the attacker is insider;
3. Steal the encrypted files.

It is also assumed that the adversary can not:

1. Perform offline dictionary attack and cryptanalysis, i.e. does not have the possibility to verify user credentials apart from entering them in the mobile client;
2. Steal the decrypted files.

Security analysis

In the offline mode the mobile client does not store the user password (see Section 4.4.1 for details), i.e. no information about the password leaks, and therefore there is no

possibility for the malicious user to check if the password he is trying to enter is correct or not. The possible scenario for the information leakage in the case of the malicious outsider is if:

(a) The hacker steals all the parts of KEY_SET_KEY;

(b) The hacker steals the KEY_SET;

(c) The hacker try the brute force offline dictionary attack on all the previous values, they have to belong to ONE TIME SESSION (the values of ”a” and ”b” belong to one period of time i.e. TIME, DEV_PASS, KEY_SET);

(d) Steal the permanent FILE_KEY (this is protected by the KEY_SET);

(e) Steal the file and try to decrypt it with offline dictionary attacks.

Still, the hacker has to get 4 values from one session: TIME, DEV_PASS, KEY_SET, FILE_KEY. At the same time he should try the offline dictionary attack on PIN+PASS. Moreover, the 4 values provide access only to 1 single file. So practically, it is very difficult to perform such attack due to the key expiration period. In accordance with the conditions 1 and 3 of our adversary model, the above actions are not possible for such adversary in the conditions presented.

The temporary nature of all parameters obliges the user to connect to server when necessary and prevents the malicious actions from the user side. The possible scenario when malicious or expired user wishes to prolong his old credentials is:

(a) Steal the file and try to decrypt it with offline dictionary attacks.

(b) He has to be able to combine;

(c) He has to steal the KEY_SET synchronized with his credentials;

(d) He has to steal the protected FILE_KEY (also synchronized) and encrypted file;

(e) He has to do everything without the mobile client (because the client checks the TIME and renews the PUBLIC_SHARE_KEY).

Basically, for a malicious user there is practically no way to use the mobile client with the old keys. The fact that the client does not contain any data to be checked or validated does not prevent the user from seeing the contents of decrypted files, but with the wrong password the decrypted files will be different from the original ones.

The mobile client still has to count the tries (to prevent the hacker to perform brute force attack) within one session. In accordance with the conditions 2 and 4 of the adversary model, the above actions are not possible for such adversary.

Common threat scenarios

This section provides the detailed description of the common scenarios in which the log and behavioral analysis is provided. The behavioral analysis can help to keep the user or administrator informed of the threat and take actions, as well as it can be useful in order to implement threat preventions or reactive actions to avoid threat propagation.

Scenario 1. *An attacker uses a valid password to perform operations on a bulk of files.*

The session time defines the period when operations can be performed until the next session renewing. During this period, it is still necessary to identify attacks and malicious behavior on file operations, in order to avoid fast attacks to perform unauthorized access to information or data modification. Some attacks present behavioral patterns based on abrupt number of operations, such as the ransomware attack, which is a growing attack [**?**] that blocks the access to valuable resources and requires a payment in order to unblock the content. The access to the resources can be blocked by the attacker through some techniques, when the content is encrypted by the attacker, the ransomware attack can be called cryptoransomware [**?**].

MOS schemes based on zero mean covariance analysis are effective to reveal abrupt changing of behaviors over time [21], making possible to identify intense malicious behaviors on offline mode of mobile clients, such in case of ransomware attack or bulk access to sensitive data.

The large number of operations over time is a well-known pattern of some attacks, due to the efforts on security measures to make the attacks infeasible over time. In this context, the operations can also be evaluated in contrast to the estimated required time for operations done by legitimate behaviors, such as the evaluation of the mean time between operations, highlighting the occurrence of infeasible behaviors in comparison to legitimate user activities.

Sparse or subtle file operations, with low number of operations distributed over different files or directories, during short period of time can indicate anomalies in contrast to the required time for legitimate directory navigation. MOS and zero mean and standardized covariance analysis can be suitable if applied to evaluate the time and location of operations, in order to identify unreachable navigation, if compared to legitimate navigation

The MOS based on zero mean covariance analysis indicates abnormalities caused by large amounts of operations during a period. Subsequently, the eigenvalue analysis highlights massive or concentrated operations over time or folder location, which is evaluated by MOS schemes in order to identify the number of malicious behaviors during the evaluated time.

This threat scenario, where an attacker uses a valid password and session to perform operations on a bulk of files, can have its steps described as:

(a) The hacker has access to the mobile client and is able to perform operations;

(b) The session time is valid;

(c) The hacker tries to perform legitimate operations, such as file decryption, encryption, reading, writing or directory navigations;

(d) The mobile client incrementally append each operation attempt time into the logging;

(e) The MOS module evaluates the logging of legitimate operations, applying zero mean and standardized covariance analysis to identify anomalies on sparse or subtle number of file operations, highlighting the occurrence of infeasible behaviors in comparison to legitimate user activities;

(f) The MOS module evaluates the logging of legitimate operations, applying zero mean covariance analysis to identify abnormalities caused by massive operations during the session time.

Scenario 2. *Usage of expired password to perform unauthorized operations.*

In the offline mode, the session time is used to restrict the operations during a specified period, although it is possible to manipulate the current time in mobile device, to emulate a period in which the session was valid. The log analysis by MOS can deal with this kind of threat, through the incremental logging of the time when each operation was performed, followed by the behavioral evaluation of operations over time.

The incremental logging assumes that new logged operations shall have equal or bigger time than the last logged operation, the violation of this rule means that the system is out of sync and indicates a malicious behavior. Additionally, a large amount or sparse operation performed at the same time, or during a short period, can indicate the use of backtrack techniques to maintain a valid session during necessary time to perform an

attack. Massive, subtle or sparse malicious operation performed during a valid session time can be identified by MOS schemes based on covariance analysis.

Applying MOS to the analysis of the time between user operations can be effective in order to reveal the occurrence of malicious behavior during an offline session. The MOS based on zero mean and standardized covariance analysis identifies anomalies on sparse or subtle variation in the number of file operations, since the eigenvalue analysis highlights the unexpected number of sparse (such as file operations on diverse folders) or subtle operations. Consequently, the result of the eigenvalue analysis is applied to MOS schemes, in order to identify the occurrence of malicious behaviors during the valid session.

The MOS based on zero mean covariance analysis indicates abnormalities caused by large amounts of operations during a period. The eigenvalue analysis based on the zero mean covariance matrix highlights massive or concentrated operations over time or location, which is evaluated by MOS schemes in order to identify the number of malicious behaviors during the evaluated time.

This threat scenario, where the attacker uses expired password to perform unauthorized operations, can have its steps described as:

(a) The hacker steals the operating system;

(b) The hacker modifies the time of the operating system to a period when the session was valid;

(c) The hacker has access to the mobile client and is able to perform operations;

(d) The hacker tries to perform legitimate operations, such as file decryption, encryption, reading, writing or directory navigations;

(e) The mobile client incrementally append each operation attempt time into the logging;

(f) The mobile client verifies if one logged time is older than the last operation time. If it is true, the MOS module classifies the evaluated operation as malicious;

(g) The MOS module evaluates the logging of legitimate operations, applying zero mean and standardized covariance analysis to identify anomalies on sparse or subtle number of file operations;

(h) The MOS module evaluates the logging of legitimate operations, applying zero mean covariance analysis to identify abnormalities caused by massive operations during the session time;

Data Modeling for Behavioral Analysis

MOS schemes are used in order to identify anomalous behavior that can indicate an attack and be used to prevent or avoid attack propagation. Therefore, it is necessary to analyze the data that can be collected from user operations on mobile client, to identify features that can be modeled and submitted to MOS schemes, according to described in Section 4.5.4.

The selected features shall be modeled as matrices which represents a signal superposition containing noise, legitimate and malicious behavior [21], grouped into time frames $\mathbf{X}^{(q)} \in \mathbb{R}^{M \times N}$, with $q = 1, 2, 3, \ldots, Q$, where M defines the decomposition of a selected feature, N defines the time decomposition and represents the number of occurrences of the feature m during the time n.

In offline mode, the user is still allowed to get access to operations that do not require communication with the server side. These operations and their selected features shall be incrementally logged by the UAL of the mobile client, in order to be analyzed by the TIM to identify malicious behaviors.

This work proposes to evaluate the following features, which represents events of the user operating the mobile client.

File Access (Time and File System Location), i.e. data access to selected files in offline mode, accessing the data stored on the mobile client. The file access feature can be decomposed into more detailed features, which are:

1. file decryption;
2. decrypted file reading;
3. decrypted file execution.

Therefore, it is necessary to generate three matrices for the following malicious behaviors analysis:

(a) massive file access, which can reveal data leakage and be identified by MOS schemes based on zero mean covariance analysis;

(b) low file access into several folders, characterized by sparse operations that can reveal unreachable navigation performed by automated file accesses in order to avoid the massive file access characterization;

(c) Malicious sparse file accesses can be identified by MOS schemes based on zero mean and standardized covariance analysis.

File Update (Time and File System Location), i.e. writing operations into selected files in offline mode, writing the data stored on the mobile client. The update feature can be decomposed into:

1. file encryption;
2. decrypted file writing.

Therefore, it is necessary to generate two matrices for malicious behaviors analysis, such as:

(a) massive file update, which can reveal ransomware or similar attacks and be identified by MOS schemes based on zero mean covariance analysis;

(b) low number of file update into several folders, characterized by sparse operations that can reveal unreachable navigation performed by automated file accesses in order to avoid the massive file access characterization. Malicious sparse file accesses can be identified by MOS schemes based on zero mean and standardized covariance analysis.

File Download (Start Time, End Time and File System Location), i.e. download requests in online mode, evaluated by the mobile client. The file download feature shall be modeled as the matrix of number downloads by file location over time, in order to perform malicious behaviors analysis, such as:

1. massive data leakage or similar attacks, which can be identified by MOS schemes based on zero mean covariance analysis;
2. low number of file download from several folders, characterized by sparse operations, which can reveal unreachable navigation performed by automated file download in order to avoid the massive file download characterization. Malicious sparse file download can be identified by MOS schemes based on zero mean and standardized covariance analysis.

Table 4.1 Speed of decryption on a client device

Device Model	File set size	Mean time of getting decrypted stream (ABE)	Mean time of decrypting content (AES)	Mean time of copying file without decrypt	Mean decryp (the w
SM-T320 Galaxy Tab Pro 8.4	10 mb	1041 ms	4230 ms	298 ms	5271 r
SM-T320 Galaxy Tab Pro 8.4	1 mb	811 ms	319 ms	23 ms	1130 r
SM-T320 Galaxy Tab Pro 8.4	1 kb	1078 ms	1 ms	0 ms	1080 r
GT-I9190 Galaxy S4 Mini	1 b	1167ms	2 ms	0 ms	1169 r
GT-I9190 Galaxy S4 Mini	1 kb	1141ms	2 ms	0 ms	1144 r
GT-I9190 Galaxy S4 Mini	1 mb	1138 ms	368 ms	35 ms	1506 r
GT-I9190 Galaxy S4 Mini	10 mb	1131 ms	3618 ms	324 ms	4750 r

Table 4.2 Speed of receiving the list of files

Files quantity	Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Error (%)	Throughput	KB/sec	Avg. Bytes
1	1000	131	29	321	31.56	0.00%	74	161.32	2233
10	1000	154	37	269	32.24	0.00%	63.4	659.79	10656
100	1000	396	68	1055	87.56	0.00%	25	2323.2	95308
1000	1000	2879	475	4634	496.02	0.00%	3.4	3178	946311

File Upload (Start Time, End Time and File System Location), i.e. upload requests in online mode, evaluated by the mobile client. The file upload feature can reveal attempts of ransomware or similar attacks and be identified by MOS schemes based on covariance analysis. Therefore, it is necessary model the matrix of number uploads by file location over time, in order to perform malicious behaviors analysis, such as:

1. massive file upload, similar to ransomware attack, which can be identified by MOS schemes based on zero mean covariance analysis;

2. low number of file upload to several folders, characterized by sparse operations, which can reveal unreachable navigation performed by automated file upload in order to avoid the massive file upload characterization. Malicious sparse file upload can be identified by MOS schemes based on zero mean and standardized covariance analysis.

4.6.2 Complexity analysis

The complexity analysis depends on the operation that the user and the mobile client perform in order to keep the app protected. In the offline mode the mobile client performs

Table 4.3 Speed of generating the user keys

Label	Samples	Average (ms)	Min (ms)	Max (ms)	Std. Dev.	Throughput	KB/sec	Avg. B
Login 5 shares available	1000	982	225	1990	271.88	20.1	146.36	7459
Login 10 shares available	1000	1140	340	4176	459.3	17.3	223.88	13218
Login 25 shares available	1000	1267	887	2303	192.75	15.5	294.55	19466

Table 4.4 Data Modeling and Eigenvalue Decomposition Time

Device	Log Size (MB)	Window (min)	Modeling (ms)	Avg. Eig. (ms)	Std. Eig. (ms)	Eig. Min. (ms)	Eig. Max. (ms)
Galaxy GT-I9300	6	60	107	209.52	18.58	183	276
Galaxy GT-I9300	6	40	115	227.26	18.13	191	289
Galaxy GT-I9300	6	20	89	268.14	21.94	229	315
Galaxy GT-I9300	6	10	90	347.42	24.11	304	421
Galaxy GT-I9300	4.1	60	20	60.90	15.19	37	106
Galaxy GT-I9300	4.1	40	20	68.72	15.71	43	114
Galaxy GT-I9300	4.1	20	34	89.04	16.78	54	133
Galaxy GT-I9300	4.1	10	21	117.24	14.36	96	171
Galaxy GT-I9300	1.4	60	10	159.82	15.82	125	197
Galaxy GT-I9300	1.4	40	10	168.06	15.90	139	220
Galaxy GT-I9300	1.4	20	11	204.4	20.46	176	269
Galaxy GT-I9300	1.4	10	13	259.00	21.34	220	315
Galaxy Tab SM-T800	6	60	7	59.30	6.55	54	74
Galaxy Tab SM-T800	6	40	8	62.56	7.05	56	80
Galaxy Tab SM-T800	6	20	10	73.28	8.59	65	95
Galaxy Tab SM-T800	6	10	8	93.48	9.13	83	130
Galaxy Tab SM-T800	4.1	60	11	18.64	4.51	16	38
Galaxy Tab SM-T800	4.1	40	11	19.64	5.12	17	38
Galaxy Tab SM-T800	4.1	20	12	25.12	5.55	21	46
Galaxy Tab SM-T800	4.1	10	12	32.32	7.29	27	55
Galaxy Tab SM-T800	1.4	60	4	49.08	6.01	42	62
Galaxy Tab SM-T800	1.4	40	5	51.42	7.36	44	74
Galaxy Tab SM-T800	1.4	20	5	51.12	7.80	54	91
Galaxy Tab SM-T800	1.4	10	7	75.24	7.71	65	90

the following actions:

1. Combine the PASS + PIN + TIME + DEV_PASS = KEY_SET_KEY → SSS secret restoring;
2. Decrypt the KEY_SET with the KEY_SET_KEY → symmetric 128/256 AES decryption;
3. Select the SHARE_KEY from the KEY_SET → no calculation;
4. Decrypt the FILE_KEY with the SHARE_KEY → ABE decryption;
5. Decrypt the file with the FILE_KEY → symmetric 128/256 AES → decryption;
6. Modify the TIME periodically → timer;
7. Count the tries within the TIME → count;
8. Modify or delete PUBLIC_SHARE_KEY → no calculation;
9. Malicious behavior detection → eigenvalues calculation.

It is easy to check that in the offline mode the mobile client does not perform complex calculations and does not use the resources extensively due to the fact that the initial key is shared and the mobile client only performs decryption, which is not a time-consuming operation. Thus, the proposal supports the concept of the light-weighted client, i.e. the most consuming actions are ABE and AES decryption.

Similarly, in the online mode the mobile client does not perform resource-consuming actions apart from J-PAKE construction which is used to renew the user/app master key (DEV_PASS):

1. Generates keys for J-PAKE → this can be a resource-consuming action. In the future we suggest to replace J-PAKE with SIS-based PKE [**? ?**];
2. Sends and receives data in the clear;
3. Performs actions 1)-7);
4. Performs the action 8) and sends the log data for server side analysis.

The proposed concept of mobile client security has been implemented in the Storgrid protected cloud environment [**?**]. Therefore, the approach is correlated with the practical usability requirements: the corporate user continues to use the mobile storage app in offline and does not need to reload the files every time the key is renewed. This methodology can be used in other mobile clients. The common advantage is that the mobile client performs the operations both in the offline and online mode and uses the key expiry and ABE to protect the privacy of the corporate data.

Table 4.1 presents the results of mobile client security workflow testing and demonstrates that the proposed scheme decrypts the data with acceptable speed. One can compare the speed of ABE/AES decryption (keys and actual data) and the speed of the whole decryption process.

Table 4.2 demonstrates the results of testing the speed of the mobile client for receiving and displaying the list of files. The client was tested for displaying up to 1000 files. Around 100 requests were generated from 10 threads. It is possible to observe that the client receives and displays the list of files depending on the user attributes with acceptable average speed, showing that the proposed protection scheme is usable.

Table 4.3 demonstrates the results of performance testing of the keys generation procedure on server. It is possible to observe that the performance does not decrease drastically with the increase of particular user shares. Each key sample generated contains the domain member code, key data and the sharing key value for each file sharing code. It is possible to conclude that the proposed client is not overloaded with calculations due to the carefully selected mathematical operations. It can be successfully used and provides acceptable level of security.

The log analysis of the Log Analysis Center (LAC) has been implemented and evaluated for offline anomaly detection in mobile clients, making it possible to apply anomaly detection techniques in a lightweight fashion, considering low processing requirements for deal with the resource constraints of mobile clients. The evaluation considered the required processing time for anomaly detection from log analysis, measuring the data modeling time through the UAL, the eigenvalue decomposition time and the required time for the EDC MOS scheme execution, which is the scheme that requires less processing capacity and provides more anomaly identification accuracy [5, 21].

The experiments were performed in two mobile devices, Galaxy GT-I9300 and Galaxy Tab SM-T800, with variations of log size and window size. The Galaxy GT-I9300 has Quad-core 1.4 GHz Cortex-A9 processor and 1 GB RAM, while the Galaxy Tab SM-T800 has its processing capacity composed by Quad-core 1.9 GHz Cortex-A15 and quad-core

1.3 GHz Cortex-A7, and 3 GB RAM.

Table 4.4 presents the data modeling time and the processing time of eigenvalues decomposition calculations to be applied to anomaly detection from user operation logs of Storgrid mobile client. The information presented by column are the device model, the log size in megabytes, the window size in minutes, the data modeling time in milliseconds, the average of eigenvalue decomposition time in milliseconds, the standard deviation of eigenvalue decomposition time in milliseconds, the minimum of eigenvalue decomposition time in milliseconds and the maximum of eigenvalue decomposition time in milliseconds.

The results show that the lower window size leads to the larger eigenvalue decomposition time, but the largest eigenvalue decomposition time, which was the maximum of 421 milliseconds with average of 347.42 milliseconds. This result highlights an acceptable speed even for the worst evaluated scenario, which is the case using a Galaxy GT-I9300 for processing 6MB with window size of 10 minutes.

Table 4.5 presents the processing time of EDC MOS calculations applied to anomaly detection from user operation logs of Storgrid mobile client. Table 4.5 respectively presents the device model, the log size in megabytes, the window size in minutes, the average of EDC calculation time in milliseconds, the standard deviation of EDC calculation time in milliseconds, the minimum of EDC calculation time in milliseconds and the maximum of EDC calculation time in milliseconds.

It is possible to observe that the processing time increases with the window size decreasing, similar to the results for eigenvalue decomposition time. The longest processing time measured is lower than 200 milliseconds, even considering window size of 10 minutes or processing 6 MB of user operation log. This result represents an acceptable processing time for anomaly detection in mobile clients.

4.7 Conclusion and future work

An important security issue faced by corporations that use cloud-based systems is how to provide security mechanisms to support offline corporate mobile clients. Once a mobile client releases the connection with the corporate cloud, no security measure implemented in the cloud infrastructure assures the protection of sensitive data stored in the mobile device. Aware of this problem and its importance, this work presented a proposal to address the offline mobile security problem combining a different cryptographic methods. Moreover, this proposed approach also prevents malicious user behavior by applying a

Table 4.5 EDC MOS scheme processing time for anomaly detection

Device	Log Size (MB)	Window (min)	Avg. EDC. (ms)	Std. EDC. (ms)	Min. EDC. (ms)	Max. EDC. (ms)
Galaxy GT-I9300	6	60	5.27	4.04	3	20
Galaxy GT-I9300	6	40	10.78	6.37	6	34
Galaxy GT-I9300	6	20	32.62	12.44	21	88
Galaxy GT-I9300	6	10	115.08	17.45	88	158
Galaxy GT-I9300	4.1	60	5.68	4.18	3	23
Galaxy GT-I9300	4.1	40	10.76	5.31	7	27
Galaxy GT-I9300	4.1	20	37.58	10.30	23	61
Galaxy GT-I9300	4.1	10	125.98	18.56	101	191
Galaxy GT-I9300	1.4	60	4.92	3.49	3	17
Galaxy GT-I9300	1.4	40	9.00	4.23	6	25
Galaxy GT-I9300	1.4	20	30.14	9.21	19	62
Galaxy GT-I9300	1.4	10	100.62	15.83	69	163
Galaxy Tab SM-T800	6	60	1.84	0.65	1	3
Galaxy Tab SM-T800	6	40	3.26	1.24	2	7
Galaxy Tab SM-T800	6	20	10.90	2.40	9	21
Galaxy Tab SM-T800	6	10	41.86	7.33	34	60
Galaxy Tab SM-T800	4.1	60	1.85	0.60	1	3
Galaxy Tab SM-T800	4.1	40	3.62	1.10	2	8
Galaxy Tab SM-T800	4.1	20	12.04	2.79	9	22
Galaxy Tab SM-T800	4.1	10	40.16	6.48	35	60
Galaxy Tab SM-T800	1.4	60	1.98	0.89	1	6
Galaxy Tab SM-T800	1.4	40	3.30	1.16	2	7
Galaxy Tab SM-T800	1.4	20	10.48	2.90	8	21
Galaxy Tab SM-T800	1.4	10	34.52	4.08	30	45

MOS-based analytic method.

As prove of concept, a fully working mobile application was developed to test the proposed security solution and acquired results provide evidence that besides achieving the desired security features, the solution also has positive results in terms of performance. This fact is due to the usage of lightweight operations and the optimized combination of the selected security methods. The proposed approach is a practical application to be used in the corporate mobile environment. It is implemented as a fully working mobile client and can be used for any type of enterprise. Also, part of concept is seed for new security solutions for big data apps.

Future works in the area can further explore enhancements in the analytics methods as well as to extended the approach to be used by mobile devices with even more severe resources constraints.

5

Tensor-Based Discriminative Dictionary Learning

Tensor-Based Discriminative Dictionary Learning for Fraud Detection

6

Conclusion and Future Work

The softest things in the world overcome the hardest things in the world.

—LAO TZU

Distributed systems has been adopted for building modern Internet services and cloud computing infrastructure. The detection of error causes, diagnose and reproduction of errors of distributed systems are challenges that motivate efforts to develop less intrusive mechanisms for monitoring and debugging distributed applications at runtime.

Network traffic analysis is one option for distributed systems measurement, although there are limitations on capacity to process large amounts of network traffic in short time, and on scalability to process network traffic where there is variation of resource demand.

In this dissertation we proposed an approach to perform deep inspection in distributed applications network traffic, in order to evaluate distributed systems at a data center through network traffic analysis, using commodity hardware and cloud computing services, in a minimally intrusive way. Thus we developed an approach based on MapReduce, to evaluate the behavior of a JXTA-based distributed system through DPI.

We evaluated the effectiveness of MapReduce to implement a DPI algorithm and its completion time scalability to measure a JXTA-based application, using virtual machines of a cloud computing provider. Also, was deeply evaluated the performance of MapReduce for packet-level analysis and DPI, characterizing the behavior followed by MapReduce phases, its processing capacity scalability and speed-up, over variations of

input size, block size and cluster size.

6.1 Conclusion

With our proposed approach, it is possible to measure the network traffic behavior of distributed applications with intensive network traffic generation, through the offline evaluation of information from the production environment of a distributed system, making it possible to use the information from the evaluated indicators, to diagnose problems and analyse performance of distributed systems.

We showed that MapReduce programming model can express algorithms for DPI, as the Algorithm **??**, implemented to extract application indicators from the network traffic of a JXTA-based distributed application. We analysed the completion time scalability achieved for different number of nodes in a Hadoop cluster composed of virtual machines, with different size of network traffic used as input. We showed the processing capacity and the completion time scalability achieved, and also was showed the influence of the number of nodes and the data input size in the processing capacity for DPI using virtual machines of Amazon EC2, for a selected scenario.

We evaluated the performance of MapReduce for packet level analysis and DPI of applications traffic, using commodity hardware, and showed how data input size, block size and cluster size cause relevant impacts into MapReduce phases, job completion time, processing capacity scalability and in the speedup achieved in comparison against the same execution by a non distributed implementation.

The results showed that although MapReduce presents a good processing capacity using cloud services or commodity computers for dealing with massive application traffic analysis, but it is necessary to evaluate the behaviour of MapReduce to process specifics data type, in order to understand its relation with the available resources and the configuration of MapReduce parameters, and to obtain an optimal performance for specific environments.

We showed that MapReduce processing capacity scalability is not proportional to number of allocated nodes, and the relative processing capacity decreases with node addition. We showed that input size, block size and cluster size are important factors to be considered to achieve better job completion time and to explore MapReduce scalability, due to the observed variation in completion time provided by different block size adopted. Also, in some cases, the processing capacity does not scale with node addition into the cluster, what highlights the importance of allocating resources according with the workload and input data, in order to avoid wasting resources.

We verified that packet level analysis and DPI are Map-intensive jobs, due to Map phase consumes more than 70% of the total job completion time, and shuffle phase is the second predominant phase. We also showed that using whole block as input for Map functions, it was achieved a poor completion time than the approach which splits the block into records.

6.2 Contributions

We attempt to analyse the processing capacity problem of measurement of distributed systems through network traffic analysis, the results of the work presented in this dissertation provide the contributions below:

1. We proposed **an approach to implements DPI algorithms through MapReduce**, using whole blocks as input for Map functions. It was **shown the effectiveness of MapReduce for a DPI algorithm** to extract indicators from a distributed application traffic, also it was **shown the completion time scalability of MapReduce for DPI**, using virtual machines of a cloud provider;

2. We developed *JNetPCAP-JXTA* [**?**], an open source **parser to extract JXTA messages from network traffic traces**;

3. We developed *Hadoop-Analyzer* [**?**], an open source **tool to extract indicators from Hadoop logs and generate graphs** of specified metrics.

4. We **characterized the behavior followed by MapReduce phases for packet level analysis and DPI**, showing that this kind of job is intense in Map phase and highlighting points that can be improved;

5. We **described the processing capacity scalability of MapReduce for packet level analysis and DPI**, evaluating the **impact caused by variations in input size, cluster size and block size**;

6. We **showed the speed-up obtained with MapReduce for DPI**, with variations in input size, cluster size and block size;

7. We **published two papers** reporting our results, as follows:

   (a) Vieira, T., Soares, P., Machado, M., Assad, R., and Garcia, V. *Evaluating Performance of Distributed Systems with MapReduce and Network Traffic*

Analysis. In ICSEA 2012, The Seventh International Conference on Software Engineering Advances. Xpert Publishing Services.

(b) Vieira, T., Soares, P., Machado, M., Assad, R., and Garcia, V. *Measuring Distributed Applications Through MapReduce and Traffic Analysis*. In Parallel and Distributed Systems (ICPADS), 2012 IEEE 18th International Conference on, pages 704 - 705.

6.2.1 Lessons Learned

The contributions cited are of scientific and academic scope, with implementations and evaluations little explored in the literature. However, with the development of this work, some important lessons were learned.

During this research, different approaches for evaluating distributed systems of cloud computing providers were studied. In this period, we could see the importance of the performance evaluation in a cloud computing environment, and the recent efforts to diagnose and evaluate system at production environment of a data center. Also, the growth of the Internet and resource utilization make necessary solutions to be able to evaluate large amounts of data in short time, with low performance degradation of the evaluated system.

MapReduce has grown as a general purpose solution for big data processing, but it is not a solution for all kind of problems, and its performance is dependent of several parameters. Some researches has been done in order to improve MapReduce performance, through analytical modelling, simulation and measurement, but the most relevant contributions in this direction was guided by realistic workload evaluations, from large MapReduce clusters.

We learned that although the facilities provided by the MapReduce for distributed processing, its performance is influenced by the environment, network topology, workload, data type and by several specific parameter configurations. Therefore, an evaluation of the MapReduce behavior using data of a realistic environment will provide more accurate and wide results, while in controlled experiments the results are more restricted and limited to the evaluated metrics and factors.

6.3 Future Work

Because of time constraints imposed on the master degree, this dissertation addresses some problems, but some problems are still open and others are emerging from current results. Thus, the following issues should be investigated as future work:

- **Evaluating of all components of the proposed approach**. This dissertation evaluated the *JNetPCAP-JXTA*, the *AppAnalyzer* and its implementation to evaluate a JXTA-based distributed application, it is necessary to evaluate the *SnifferServer*, *Manager* and the whole system working together, analysing their impact into the measured distributed system and the scalability achieved;

- Development of a **technique for the efficient evaluation of distributed systems through information extracted from network traffic**. This dissertation addressed the problem of processing capacity for measuring distributed systems through network traffic analysis, but it is necessary an efficient approach to diagnose problems of distributed systems, using information of flows, connections, throughput and response time obtained from network traffic analysis;

- Development of a **analytic model and simulations**, using information of MapReduce behavior for network traffic analysis, measured by this dissertation, to reproduce its characteristics and enable the evaluation and prediction of some cases of MapReduce for network traffic analysis;

Bibliography

[1] Ahmed, M., Mahmood, A. N., and Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, **60**, 19–31.

[2] Almotairi, S., Clark, A., Mohay, G., and Zimmermann, J. (2009). A technique for detecting new attacks in low-interaction honeypot traffic. In *Internet Monitoring and Protection, 2009. ICIMP'09. Fourth International Conference on*, pages 7–13. IEEE.

[3] Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. (2014). Network anomaly detection: methods, systems and tools. *IEEE Communications Surveys & Tutorials*, **16**(1), 303–336.

[4] Callegari, C., Gazzarrini, L., Giordano, S., Pagano, M., and Pepe, T. (2011). A novel pca-based network anomaly detection. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE.

[5] da Costa, J. P. C. L., Thakre, A., Roemer, F., and Haardt, M. (2009). Comparison of model order selection techniques for high-resolution parameter estimation algorithms. In *Proc. 54th International Scientific Colloquium (IWK'09), Ilmenau, Germany*.

[6] da Costa, J. P. C. L., de Freitas, E. P., David, B. M., Serrano, A. M. R., Amaral, D., and de Sousa Jr, R. T. (2012). Improved blind automatic malicious activity detection in honeypot data. In *The International Conference on Forensic Computer Science (ICoFCS)*.

[7] David, B. M., da Costa, J. P. C. L., Nascimento, A. C., Amaral, D., Holtz, M., and de Sousa Jr, R. T. (2011). Blind automatic malicious activity detection in honeypot data. In *The International Conference on Forensic Computer Science (ICoFCS)*.

[8] Fleiss, J. L., Levin, B., and Paik, M. C. (2013). *Statistical methods for rates and proportions*. John Wiley & Sons.

[9] Ghourabi, A., Abbes, T., and Bouhoula, A. (2010). Data analyzer based on data mining for honeypot router. In *Computer Systems and Applications (AICCSA), 2010 IEEE/ACS International Conference on*, pages 1–6. IEEE.

[10] He, W., Hu, G., Yao, X., Kan, G., Wang, H., and Xiang, H. (2008). Applying multiple time series data mining to large-scale network traffic analysis. In *2008 IEEE Conference on Cybernetics and Intelligent Systems*, pages 394–399.

[11] Huang, C.-T., Chang, R. K. C., and Huang, P. (2009). Editorial: Signal processing applications in network intrusion detection systems. *EURASIP J. Adv. Signal Process*, **2009**, 9:1–9:2.

[12] Ji, S.-Y., Jeong, B.-K., Choi, S., and Jeong, D. H. (2016). A multi-level intrusion detection method for abnormal network behaviors. *Journal of Network and Computer Applications*, **62**, 9–17.

[13] Jin, S. and Yeung, D. S. (2004). A covariance analysis model for ddos attack detection. In *Communications, 2004 IEEE International Conference on*, volume 4, pages 1882–1886. IEEE.

[14] Lakhina, A., Crovella, M., and Diot, C. (2005). Mining anomalies using traffic feature distributions. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 217–228. ACM.

[15] Lee, Y.-J., Yeh, Y.-R., and Wang, Y.-C. F. (2013). Anomaly detection via online oversampling principal component analysis. *Knowledge and Data Engineering, IEEE Transactions on*, **25**(7), 1460–1470.

[16] Lu, W. and Ghorbani, A. A. (2009). Network anomaly detection based on wavelet analysis. *EURASIP J. Adv. Signal Process*, **2009**, 4:1–4:16.

[17] Mudzingwa, D. and Agrawal, R. (2012). A study of methodologies used in intrusion detection and prevention systems (idps). In *Southeastcon, 2012 Proceedings of IEEE*, pages 1–6. IEEE.

[18] Osanaiye, O., Choo, K.-K. R., and Dlodlo, M. (2016). Distributed denial of service (ddos) resilience in cloud: review and conceptual cloud ddos mitigation framework. *Journal of Network and Computer Applications*, **67**, 147–165.

[19] Raynal, F., Berthier, Y., Biondi, P., and Kaminsky, D. (2004). Honeypot forensics. In *Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC*, pages 22–29. IEEE.

[20] Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A.-A. (2009). A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*.

[21] Tenório, D. F., da Costa, J. P. C. L., and de Sousa Jr, R. T. (2013). Greatest eigenvalue time vector approach for blind detection of malicious traffic. In *The International Conference on Forensic Computer Science (ICoFCS)*.

[22] Vieira, T. P. d. B., Fernandes, S. F. d. L., and Garcia, V. C. (2013). Evaluating mapreduce for profiling application traffic. In *Proceedings of the First Edition Workshop on High Performance and Programmable Networking*, HPPN '13, pages 45–52, New York, NY, USA. ACM.

[23] Zakaria, W. Z. A. and Kiah, M. L. M. (2012). A review on artificial intelligence techniques for developing intelligent honeypot. In *Proceeding of: 8th International Conference on Computing Technology and Information Management, At Seoul, Korea*.

[24] Zonglin, L., Guangmin, H., Xingmiao, Y., and Dan, Y. (2009). Detecting distributed network traffic anomaly with network-wide correlation analysis. *EURASIP J. Adv. Signal Process*, **2009**, 2:1–2:11.