112 學年度
國立成功大學 資訊工程學系碩士班
甄試審查資料

申請人：洪胤勛

就讀學校：國立暨南國際大學 資訊工程學系

目錄

一、 自傳

成長背景

學生洪胤勛，來自彰化縣，高中就讀彰化高中。在高一時，我們課程裡面有電腦課，那是我第一次接觸到程式語言。當時，我們從 C++的 Hello world 開始，到後來去解程式競賽的題目，我知道能夠把之前所學的數學知識透過程式語言來實現化，讓我非常興奮。到了高一下時，我們有一堂可以自由選擇的課，其中有一堂課是 C++進階，我毅然決然選擇了那門課。在其中，我學習到了 C++的 STL，這也讓我對資料結構有了初步的認識，其中的學習讓我在高中三年間有個更明確要選擇資工這條路的目標，最後也很幸運考到了暨大資工。

大學時期

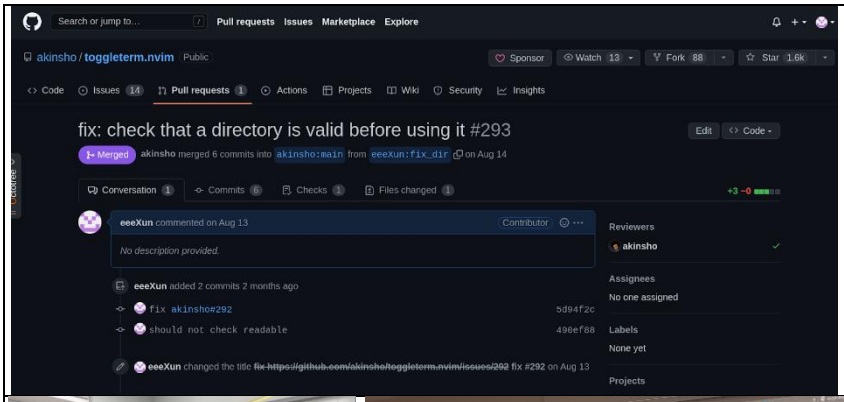
在大一上學期的「計算機概論」裡，雖說在高中時已經有一定的程式基礎，但我對這門課仍完全不鬆懈。在高中所學的基礎很多部份都不夠扎實，所以下課時常常帶著問題去找老師，在課後也花了許多時間複習和預習，最後也沒有辜負自己的努力，拿了 **100** 分。除了計算機概論，我還修了通識 Python 程式設計，在這堂課，不同於以往程式的結果只能呈現在黑色的終端，我學到了如何讓程式呈現在圖形介面上，輸出在圖形介面就會比輸出在終端介面麻煩許多，但這也讓我學會如何 divide and conquer，最後期末也做出俄羅斯方塊，並拿到 **99** 分。

大一下的「程式設計」讓我收穫滿滿，這是我第一次接觸 Object-Oriented Programming，而且每週也至少花 5 個小時以上在這堂課上，讓我學到如何更好地把程式封裝起來。在這堂課裡，老師曾經用黑色的終端進行操作，這不讓我驚訝，讓我驚訝的是竟然可以在裡面編輯，後來去問老師，才知道他是 ssh 到 Linux 使用 vim，那時我甚麼都不懂，但也讓我二年級暑假有了目標，就是要學習 Linux 和 vim。

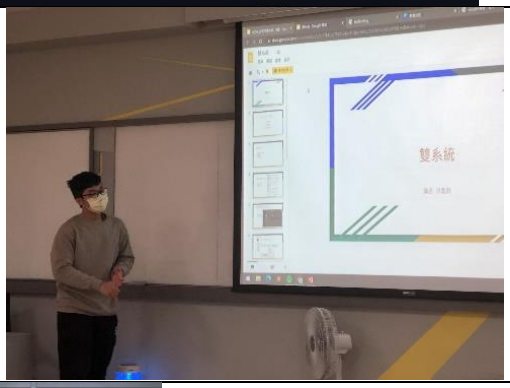
大二暑假，我開始在自己的筆電裝了雙系統，當然其中是一次又一次地把筆電的硬碟搞壞，但這一次次的失敗對我來說使我學到了更多知識。之後我便開始學習了 Linux 的基本指令、shell script、vim、make 和 Makefile……。在學習 vim 中，發現了一些有趣的插件，能幫 vim 做到一些額外的事，但在使用期間有時會發現一些 Bug，一開始會常常發 issue，後來發覺，與其當個製造問題的人，不如當個解決問題的人，因此開始學習如何去 trace code，並且發送 Pull Request，這也讓我有了參與社群的快樂。在大二，我修習了「弗列斯科網頁程式設計」這門課，在這堂課裡，我學到了網頁是如何運作，如何在 Python 上利用 Flask 提供網頁服務，還有資料庫建立等等。在期末專題中，也與同學一起建立了課程討論區，自己也在其中負責了資料庫的設計與查詢，並且自學了 container，來讓我們的服務能快速建立起來。

大三是我大學最充實的一年，我加入 ACM（Association for Computing Machinery）學生分會，並擔任副會長。我們時常舉辦讀書會和電腦學習相關的活動，並在暑假期間，與同學和學弟妹組隊一起參加了 HPC-AI Advisory Council 舉辦的亞太地區 RDMA（Remote Direct Memory Access）黑客松，在參賽過程中學到如何透過 RDMA 來提升電腦溝通的速度，最後在歐亞地區全部 41 個參賽隊伍中，拿到**第三名**的佳績。在專題實作，我進入了吳坤熹老師的 PEARL（Protocol Engineering and Application Research Laboratory）實驗室，學習了許多網路與微服務（Kubernetes）等知識，並將專題成果撰寫成一篇論文「可延展式與容錯性 WebAPI 安全管制機制設計」，投稿至 TANET 2022 會議。在大四暑假，參與了校內的「獎勵大學部學生參與專題研究計畫」，研究 Secret Sharing 相關領域，並以「Secret Sharing with Multi-cover Steganographic Audio Files」為題目，投稿於 TANET 2022 會議。

課外活動



熱愛使用 vim，並喜歡參與社團討論，也在 vim 的周邊插件多達 **10 個以上**的 Pull Request 被 merge 進去。左圖是修改原本插件本身不當操作會出現 Bug 的問題。



大三時擔任 ACM 副會長，在線上曾舉辦讀書會，線下的則曾舉辦電腦拆解，指導同學們理解裡面元件，與安裝電腦的作業系統。

大三暑假參與 HPC-AI Advisory Council 舉辦的亞太地區 RDMA 黑客松，在其中學習到 RDMA 透過 OS kernel bypass 來讓網路傳遞更快速，最終獲得**第三名**佳績。

二、 專題與論文

1. 可延展式與容錯性 WebAPI 安全管制機制設計（專題&論文）

探討傳統 Web Server 與 WebAPI Gateway 的差異，並對這兩者的運作模式做效能上的分析。WebAPI Gateway 能減少公有 IP 位址暴露的數量，但同時也會有單點故障的風險，因此本研究使用虛擬化 container 的技術，讓 API Gateway 跑在多個 container 上，並使用 Kubernetes 來分擔 container 的壓力和確保 container 的正常運作。最後將專題成果整理論文，投稿至 TANET 2022 的年會。

2. Secret Sharing with Multi-cover Steganographic Audio Files（論文）

祕密共享（secret sharing）為一種分享資料的技 術，它會把資料拆成若干持份(sharing)，並且把 sharing 分給不同的人。若要回復原本的資料，則需要湊齊大於或等於一定門檻(threshold)數量的 sharing，否則不能還原出原本的資料。本論文將探討如何安全的把這些 sharing 分送給不同的人，並導入隱寫術，將 sharing 藏在聲音中，讓這些 sharing 在不易被察覺的情況下發送給不同的人。最後將研究成果整理成論文，投稿至 TANET 2022 的年會。

三、 研究計畫

隨著科技的進步，人們上網的時間越來越長，網路已經是生活中不可或缺的一部分，這也使得網路傳輸的安全顯得格外重要。我在大四暑假，研究 secret sharing 中，發現原來不是只有 Shamir's Secret Sharing，還有 Blakley's Secret Sharing、Counting-based Secret Sharing 或 Matrix-based Secret Sharing。其中也有針對圖片的 Visual Secret Sharing 或是針對聲音的 Audio Secret Sharing。我有意往這方面研究，發掘其在網路上的應用場景。

在我研究的「Secret Sharing with Multi-cover Steganographic Audio Files」題目，探討了 secret sharing 與隱寫術。在研究過程中，曾瀏覽到有篇論文提到在圖片中使用 LSB 的藏法會相當危險，因為圖片鄰近 pixel 的值通常會連續，他們的 LSB 值也會連續，但事實是否如此，如果圖片雜訊較多或是經過壓縮，那結果是否還會一樣呢? 而聲音的 LSB 是否真的沒有連續性?我也有意往這方面探索。

同時我也對資訊安全相關領域想多嘗試。這學期我修習了「C 語言安全程式設計」，目前學到許多相關知識，像是 Buffer Overflow 會造成的問題，也期待自己能在這堂課收穫滿滿。而我也會在大四剩餘的時間裡，學習資安相關的知識。

四、 其他文件

Certificate of Achievement

awarded to

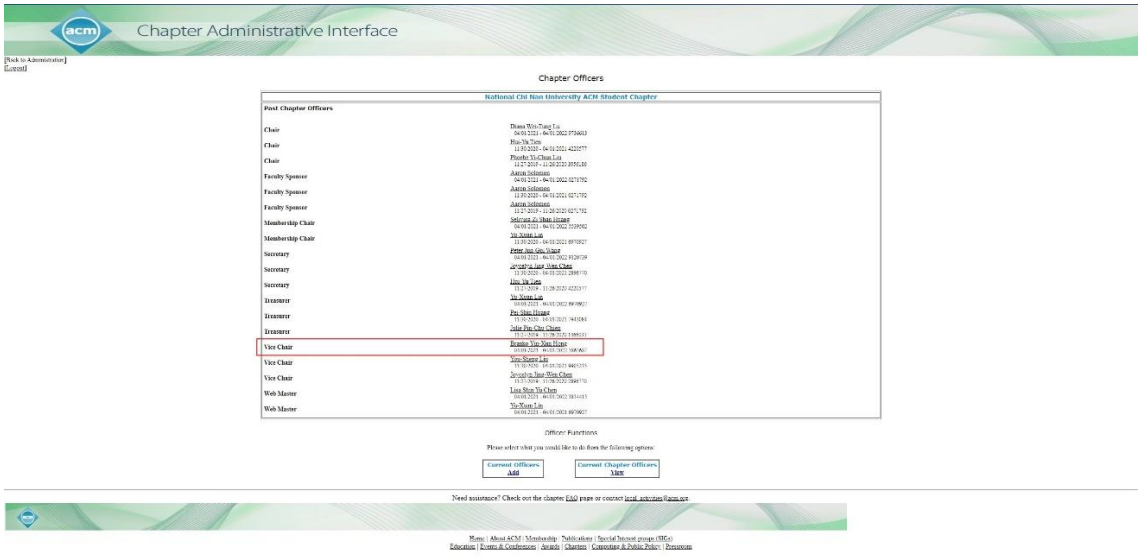
Mr. Yin-Hsun Hong

National Chi Nan University
1ide1die1die
The 2021 Taiwan Online Programming Contest
23 October 2021

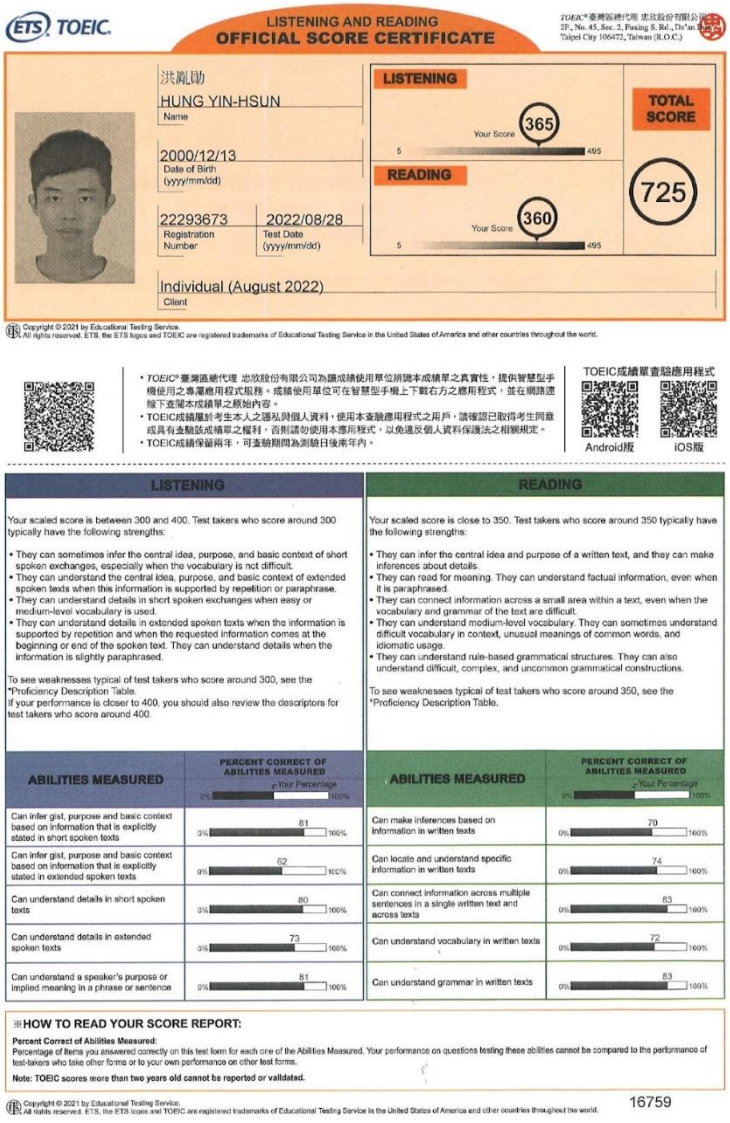
Honorable Mention

William B. Poucher, Ph. D.
ICPC Executive Director

參與 ICPC



ACM 副會長



ETS TOEIC

LISTENING AND READING
OFFICIAL SCORE CERTIFICATE

TOEIC® 臺灣區總代理 忠欣股份有限公司
2F., No. 45, Sec. 2, Fuxing S. Rd., Da'an Dist.,
Taipei City 105472, Taiwan (R.O.C.)

洪胤勛
HUNG YIN-HSUN
Name

2000/12/13
Date of Birth
(yyyy/mm/dd)

22293673
Registration Number

2022/08/28
Test Date
(yyyy/mm/dd)

Individual (August 2022)
Client

LISTENING
Your Score 365
5 495

READING
Your Score 360
5 495

TOTAL SCORE
725

Copyright © 2021 by Educational Testing Service.
All rights reserved. ETS, the ETS logos and TOEIC are registered trademarks of Educational Testing Service in the United States of America and other countries throughout the world.

• TOEIC® 臺灣區總代理 忠欣股份有限公司為讓成績使用單位辨識本成績單之真實性，提供智慧型手機使用之專屬應用程式服務。成績使用單位可在智慧型手機上下載右方之應用程式，並在網路連線下查閱本成績單之原始內容。
• TOEIC成績屬於考生本人之個私與個人資料，使用本查驗應用程式之用戶，請確認已取得考生同意或具有金驗該成績單之權利，否則請勿使用本應用程式，以免違反個人資料保護法之相關規定。
• TOEIC成績保留兩年，可查驗期間為測驗日後兩年內。

TOEIC成績單查驗應用程式
Android版 iOS版

LISTENING

Your scaled score is between 300 and 400. Test takers who score around 300 typically have the following strengths:

- They can sometimes infer the central idea, purpose, and basic context of short spoken exchanges, especially when the vocabulary is not difficult.
- They can understand the central idea, purpose, and basic context of extended spoken texts when this information is supported by repetition or paraphrase.
- They can understand details in short spoken exchanges when easy or medium-level vocabulary is used.
- They can understand details in extended spoken texts when the information is supported by repetition and when the requested information comes at the beginning or end of the spoken text. They can understand details when the information is slightly paraphrased.

To see weaknesses typical of test takers who score around 300, see the *Proficiency Description Table.
If your performance is closer to 400, you should also review the descriptors for test takers who score around 400.

ABILITIES MEASURED	PERCENT CORRECT OF ABILITIES MEASURED (Your Percentage)
Can infer gist, purpose and basic context based on information that is explicitly stated in short spoken texts	81
Can infer gist, purpose and basic context based on information that is explicitly stated in extended spoken texts	62
Can understand details in short spoken texts	80
Can understand details in extended spoken texts	73
Can understand a speaker's purpose or implied meaning in a phrase or sentence	81

READING

Your scaled score is close to 350. Test takers who score around 350 typically have the following strengths:

- They can infer the central idea and purpose of a written text, and they can make inferences about details.
- They can read for meaning. They can understand factual information, even when it is paraphrased.
- They can connect information across a small area within a text, even when the vocabulary and grammar of the text are difficult.
- They can understand medium-level vocabulary. They can sometimes understand difficult vocabulary in context, unusual meanings of common words, and idiomatic usage.
- They can understand rule-based grammatical structures. They can also understand difficult, complex, and uncommon grammatical constructions.

To see weaknesses typical of test takers who score around 350, see the *Proficiency Description Table.

ABILITIES MEASURED	PERCENT CORRECT OF ABILITIES MEASURED (Your Percentage)
Can make inferences based on information in written texts	70
Can locate and understand specific information in written texts	74
Can connect information across multiple sentences in a single written text and across texts	83
Can understand vocabulary in written texts	72
Can understand grammar in written texts	83

※HOW TO READ YOUR SCORE REPORT:

Percent Correct of Abilities Measured:
Percentage of items you answered correctly on this test form for each one of the Abilities Measured. Your performance on questions testing these abilities cannot be compared to the performance of test-takers who take other forms or to your own performance on other test forms.

Note: TOEIC scores more than two years old cannot be reported or validated.

16759

Copyright © 2021 by Educational Testing Service.
All rights reserved. ETS, the ETS logos and TOEIC are registered trademarks of Educational Testing Service in the United States of America and other countries throughout the world.

多益證明

五、 論文一

可延展式與容錯性 WebAPI 安全管制機制設計

洪胤勛　吳坤熹
國立暨南國際大學 資訊工程學系[1]
{s108321019,solomon}@ncnu.edu.tw

摘要

本研究將探討傳統的 Web Server 與 WebAPI Gateway 之間的差異，並對這兩者的運作模式做效能上的分析。WebAPI Gateway 能減少公有 IP 位址暴露的數量，但同也會有單點故障的風險。因此本研究會使用虛擬化 container 的技術，讓 API Gateway 跑在多個 container 上，並使用 Kubernetes 來分擔 container 的壓力和確保 container 的正常運作。

I. 前言

在一個分工的時代，工程師也要懂得如何分工。在軟體工程裡面，有個專有名詞叫做 API（Application Programming Interface）。它是一個溝通的介面，旨在讓工程師們專心在他們的任務，而不需要憂慮其他層面的問題。比如說一個人用電腦時，他只要知道鍵盤、滑鼠可以輸入資料，螢幕可以輸出結果，但他並不需要去知道電腦是如何運作的。如果是網頁的開發，通常會分為前端和後端，前端負責設計使用者介面，後端負責資料的儲存，前後端之間通常會透過 HTTP 或 HTTPS 來溝通，並使用 JSON 或 XML 格式來傳遞資料，此種模式稱為 WebAPI。

在使用網頁時，通常使用者（client）會需要去 WebAPI 服務端（也就是 server）提取資料。當使用者需要多筆資料，而那些資料可能是由不同的服務端所提供的，那架構就會像圖1。但這架構會造成一個大問題，每個服務端都必須要有一個公用的 IP 地址，也就表示所有人都可以直接連到伺服務器。尤其在 IoT 的佈建當中，許多的感測器是由外包廠商負責建置，再透過公開的4G 網路傳回伺服器。IoT 設備由4G 取得的是浮動的 IP 位址，因此實務上伺服端都必須擁有固定的公開 IP 位址，以讓 IoT 設備傳回資料。如果有心人士想竊取伺服器的資料，或是想讓伺服器癱瘓，這種架構極有可能會讓有心人士藉由伺服器的公開 IP 位址達成他們的目的。

於是後來有了圖2這樣的 Gateway 架構。Gateway 扮演了使用者連到服務端間的中繼站，使用者需要的資料都間接透過 Gateway 取得，服務端要給使用者的資料也都透過 Gateway 傳送。公用的 IP 地址只需要 Gateway 這台電腦擁有，真正提供服務的服務端可以只使用私有 IP 地址。這解決了剛剛所提到的問題，服務端不再需要暴露在所有人都可以直接連線的公開網路中。除了上述的好處之外，Gateway 還能讓服務端的管理者清楚掌握當前有哪些電腦正在對外提供服務，並把沒在提供服務的電腦給關機，減少了被入侵的機會。

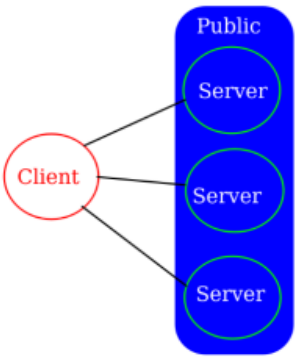


圖1. 傳統 Web Server

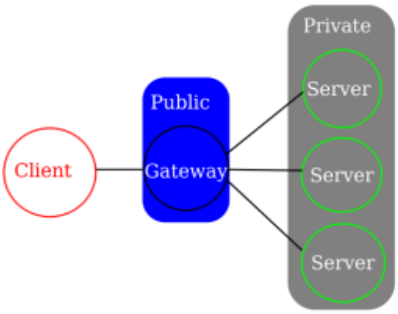


圖2. WebAPI Gateway

II. 研究問題

雖然上述的架構解決了把服務端暴露在公用 IP 的環境，但相對地，也讓風險全部都承擔在 Gateway 上。萬一，Gateway 停止運作，那麼使用者也沒辦法正常存取服務。針對此問題，勢必要有個自動化的機制，讓停止

[1] 本研究成果感謝國立暨南國際大學與埔基醫療財團法人埔里基督教醫院產學合作之「埔暨計畫」(111-PuChi-AIR-006) 經費贊助

運作的 Gateway 能夠自動重啟。因此本研究將使用 Gateway 與 Kubernetes 搭配，讓因意外而停止運作的 Gateway 自動重啟。

網路上有許多常見的 API Gateway 開源項目，本研究將對以下三個 API Gateway：Express Gateway，API Umbrella，Kong API Gateway 進行效能上的分析與比較。

III. 實驗所需

A. API Gateway

1. 此實驗將比較 Express Gateway [1]，API Umbrella [2]，Kong API Gateway [3]這三個 API Gateway 的效能。
2. 在建置 API Gateway 時所使用的平台，將分別使用實體主機、VM 虛擬機、以及 Docker 的容器（container）技術。

B. Docker

相較於虛擬機[4]（VM），Docker [5]的 container 是個類似虛擬的技術，但卻比虛擬機更為輕巧，更能夠快速地建立起來。傳統的虛擬機著重在將硬體設備給虛擬化，而 container 則著重在將作業系統給虛擬化。如圖3所示，左邊為傳統的虛擬機，右邊則為 container 的架構。我們可以看到，左邊的虛擬機可以把一台電腦的硬體資源分配給多台虛擬機，但每台虛擬機都需要有作業系統在上面執行。右邊的 container 一樣是把硬體資源分配給 container，但值得注意的是，每個 container 並不需要再架起一個作業系統，這也是為甚麼 container 比起虛擬機更為輕量，更能夠快速建立起來的原因。

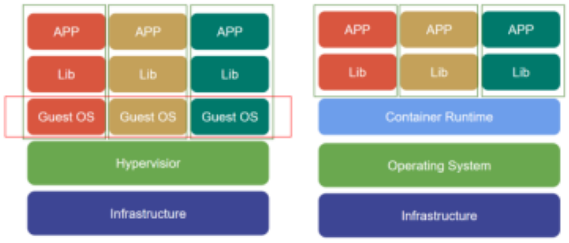


圖3. VM vs. Container

C. Kubernetes

Kubernetes [6]簡稱為 K8s，由 Google 設計出來，現在已屬於 Cloud Native Computing Foundation。K8s 中，一個基本單位為 Pod，Pod 裡面可以有一個或多個 container，但通常只會有一個。在運行 K8s 時，有兩個主要角色，如圖4所示，為 Master 和 Worker。Master 和一個或多個 Worker 組合起來就稱為 Cluster。使用者會在 Master 端下達指令，告訴 K8s 我們要架起服務的需求共要幾個 Pod。Master 則會根據 Worker 的狀態，來決定如何把所有的 Pod 分配到各個 Worker 上。K8s 中還有一個物件稱為 Deployment，它的工作是保持 Pod 的數量；當有 Pod 若因意外故障，Deployment 則會自動啟動新的 Pod，讓 Pod 維持在一定的數量。

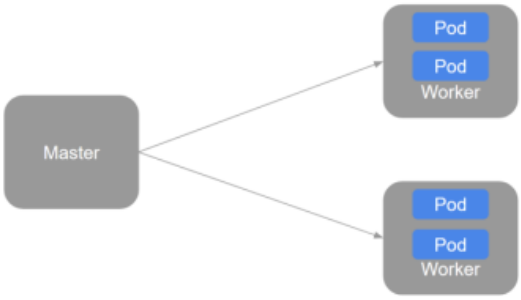


圖4. Kubernetes 架構

D. K6

K6 [7]為壓力測試的工具，它能夠使用 CLI 或 JavaScript 編寫自己定義的測試內容，例如要有多少個虛擬使用者去做請求，還有這段測試要執行多久。

K6執行完時，會輸出請求傳送時間、等待時間、接收時間等等，之後可以選擇多種輸出方式，如圖5。預設會在終端機有摘要輸出，（平均值，最大值，最小值……），也能決定是否要把這些摘要輸出給存起來。除了上述的摘要輸出，K6也能在測試時，用 time-series 的方式把每個時間點的結果給存起來，而這裡又分為 stream 的方式（每紀錄一筆就輸出到資料庫）或是結束時再把整個結果寫入到檔案裡。

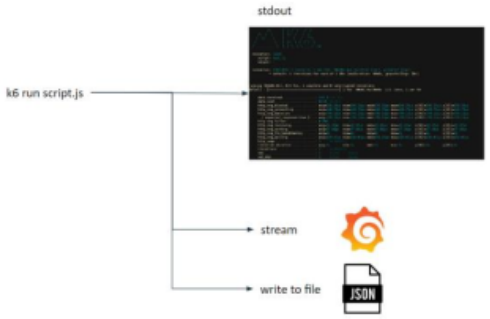


圖5. K6輸出方式

IV. 實驗步驟

A. 直接針對 API 的來源做壓力測試

圖6. API 來源測試

B. 使用 API Gateway，來轉傳 API，並做壓力測試

圖7. Gateway

C. 把 API Gateway 部署在 K8s 上

圖8. Gateway 在 K8s 上

V. 實驗結果

底下實驗圖中，橫軸代表時間線，縱軸代表虛擬使用者的數量；藍線代表直接對 API 來源的請求時間，紅線代表透過在 VMware 上的 API Gateway 的請求時間，黃線則是代表在 K8s 的 API Gateway 的請求時間。

圖9和圖12的黃線分別是 Express Gateway 和 Kong API Gateway 在 K8s 使用一個 Pod 執行的結果；圖10和圖13的黃線分別是 Express Gateway 和 Kong API Gateway 在 K8s 使用兩個 Pod 執行，並且兩個 Pod 在相同主機的結果；圖11和圖14分別是 Express Gateway 和 Kong API Gateway 在 K8s 使用兩個 Pod 執行，並且兩個 Pod 在不同主機的結果。

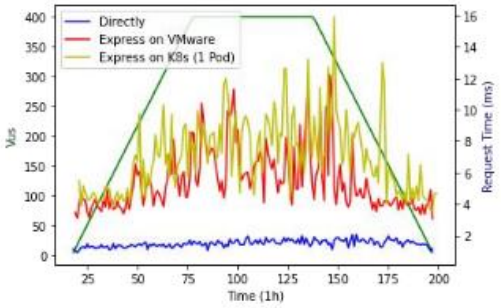


圖9. Express 比較圖

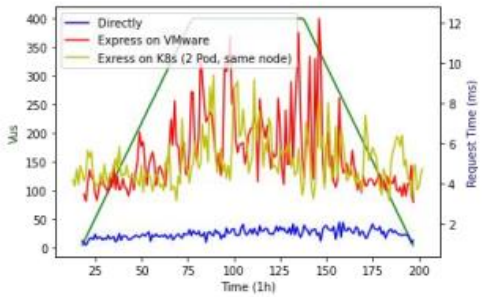


圖10. Express 比較圖

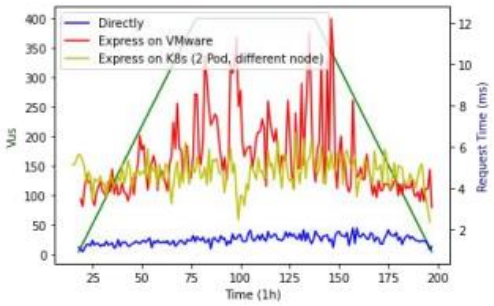


圖11. Express 比較圖

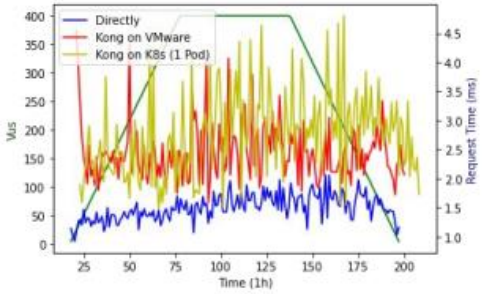


圖12. Kong 比較圖

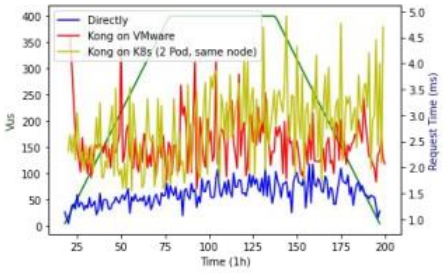


圖13. Kong 比較圖

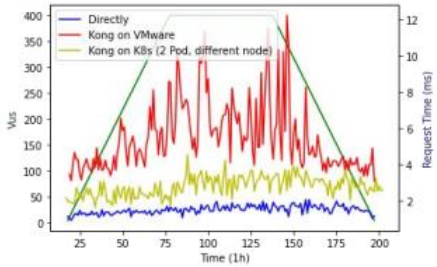


圖14. Kong 比較圖

VI. 結論

Express Gateway 和 Kong API Gateway 在請求時間的比較，明顯是 Kong 比較好，在社群的活躍度上，Kong API Gateway 也是勝過 Express Gateway。且在設定 API Gateway 方面，Express Gateway 只能透過修改設定檔來配置，Kong API Gateway 可以透過發送 HTTP/HTTPS request 來作到 CRUD（Create，Read，Update，Delete）。且還有第三方插件，能讓 Kong API Gateway 在網頁上做設定。

在裸機上的 API Gateway 效能最好，因為所需要 network hopping 數最少，且可以直接使用電腦上的硬體資源。而 Container 則需要透過 Linux 的 cgroup [8]來分配， 且不是100%的分配，所以效能略低，但不至於太差。且 Container 的優勢就在於能快速安裝且啟動，安裝設定若希望調整，就可以重新另創一個新的容器，不用像裸機一樣需要擔心是否會搞亂環境。

在 K8s 的比較上，單個 Pod 的 Request 時間比裝在 VMware 上來的慢。當 Pod 提升到兩個的時候，當這兩個 Pod 都在相同的 Node 上，Request 時間已經與在 VMware 上差不多；當兩個 Pod 在不相同的 Node 上，可以看到 Request 時間已經相比在 VMware 上較低。

綜合以上實驗，我們可以看出，利用 Kubernetes 的自動重啟功能，可以讓 API Gateway 的運作更穩健，故障重啟的時間更加縮短。

參考文獻

[1] M. Bawane, I. Gawande, V. Joshi, R. Nikam, and S. A. Bachwani, "A Review on Technologies used in MERN stack," *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, 2022.

[2] A. Gámez Díaz, P. Fernández Montes, and A. Ruiz Cortés, "Towards SLA-driven API gateways," *XI JORNADAS DE CIENCIA E INGENIERÍA DE SERVICIOS*, 2015.

[3] R. Xu, W. Jin, and D. Kim, "Microservice security agent based on API gateway in edge computing," *Sensors*, vol. 19, no. 22, p. 4905, 2019.

[4] R. P. Goldberg, "Survey of virtual machine research," *Computer*, vol. 7, no. 6, pp. 34-45, 1974.

[5] C. Anderson, "Docker [software engineering]," *IEEE Software*, vol. 32, no. 3, pp. 102-c3, 2015.

[6] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE cloud computing*, vol. 1, no. 3, pp. 81-84, 2014.

[7] V. Seifermann, "Application performance monitoring in microservice-based systems," Bachelor, Institute of Software Technology Reliable Software Systems, University of Stuttgart, 2017.

[8] M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose, "Performance evaluation of container-based virtualization for high performance computing environments," in *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2013: IEEE, pp. 233-240.

六、 論文二

Secret Sharing with Multi-cover Steganographic Audio Files

洪胤勛　吳坤熹
國立暨南國際大學 資訊工程學系[1]
{s108321019,solomon}@ncnu.edu.tw

摘要

祕密共享（secret sharing）[1, 2]為一種分享資料的技術，它會把資料拆成若干持份(sharing)，並且把 sharing 分給不同的人。若要回復原本的資料，則需要湊齊大於或等於一定門檻(threshold)數量的 sharing，否則不能還原出原本的資料。本論文將探討如何安全的把這些 sharing 分送給不同的人，並導入隱寫術，將 sharing 藏在聲音中，讓這些 sharing 在不易被察覺的情況下發送給不同的人。

I. 前言

隨著資安的議題日益重要，許多加解密的原理與工具更顯得它們的重要性。有些機密性的資料，不能夠直接赤裸裸的流通在網路上，因此有些人選擇給資料做了加密，防止資訊外洩。但在某些情況下，做了加密反而「欲蓋彌彰」。比如在一個被監控的網路環境中，或是網路會流經不受信任的節點，若你的封包攜帶了加密的資料，管理者通常能很快發現此封包異常，因此封包可能就此被丟棄又或是被選出來破解。為了避免封包被遺棄或是被別人嘗試破解，有人會選擇把機密的資料藏在一些不容易發現的地方，像是圖片或是聲音，甚至是封包的表頭，此方法又稱隱寫術（steganography）。

Secret sharing 在資安的領域也相當重要，它與加解密不同，secret sharing 為一種分享資料的技術，在這種架構下，它解決了祕密資料掌握在單一個人手上的保密性問題。比如需要多人授權的密碼，像是金庫的密碼，老闆可能會有緊急狀況，無法親自到金庫，因此需要部下幫他開啟，但如果把密碼只交給一個人，那將非常危險，因此理想狀況是需要多人才能開啟；又或是階層式授權的密碼，給經理和主管有開啟的權力，但也不能讓單一個經理或是單一個主管直接擁有密碼，而是讓一個經理和一個主管，或是三個主管決定要打開金庫時，才能得出密碼。這些問題都可以靠 secret sharing 來解決。

Secret sharing 同時也解決機密資料掌握在單一個人手上的另兩個問題。第一為資料的穩健性（robustness），在單一資料的架構下，當系統被駭客入侵，資料被竊取走，那就表示機密一定被駭客知曉；但在 secret sharing 的架構下，即使駭客竊走一份資料，他也無法得知祕密到底為何，他必須去找到一定數量的資料才有辦法還原出原始的機密資料。因此 secret sharing 有助於達成較佳的縱深防禦（defense in depth）。第二為資料的可靠性（reliability），在單一資料的架構下，資料若不小心遺失或是被破壞掉了，那祕密就此消失，再也無法取得；但在 secret sharing 的架構下，即使遺失一份或損壞一份資料，仍然可以靠其他的資料得以復原。

Secret sharing 的做法在1979年相繼被 Adi Shamir [1]和 George Blakley [2]發表。而他們的解法又常分別被稱為 Shamir's Secret Sharing 與 Blakley's Secret Sharing。 Shamir's Secret Sharing 使用多項式來達成目的，而 Blakley's Secret Sharing 使用有限域的幾何空間。雖然兩者都有 threshold 的設計，但 Blakley's Secret Sharing 較複雜且效率較低[1]，因此本論文將著重在 Shamir's Secret Sharing。

本論文除了使用 Shamir's Secret Sharing 外，為了讓資料不被輕易察覺，還使用隱寫術，把產生的 sharing 藏在聲音檔裡面，並且派送至不同目的地。

II. 研究動機

Secret sharing 與一般的單一祕密架構不同，它會拆分成很多不同的 sharing，因為要收集到多份的 sharing 才能解開祕密，這使得它要被破解更困難；因為要破壞掉多份的 sharing，才能使祕密真正被揭露出來，這讓外人要破解它更艱辛。以上兩個原因，讓 secret sharing 比起單一祕密的架構更加強韌，更可以信任。

但也因為祕密會被拆分成多個 sharing，這讓傳遞和保存 sharing 時，需要花費額外的資源來儲存和傳送這些 sharing。如果 sharing 赤裸裸地呈現在封包內容裏，那麼可能在這些 sharing 到達目的地前，就會被偵測並刪除了，這並不是我們想要的結果。因此，本論文研究如何讓 sharing 能夠在安全、不易被察覺的情況下抵達目的地。

[1] 本研究成果感謝國立暨南國際大學與埔基醫療財團法人埔里基督教醫院產學合作之「埔暨計畫」(111-PuChi-AIR-006) 經費贊助

III. 相關研究

A. Securing matrix counting-based secret-sharing involving crypto steganography [3]

這篇的作者使用了 matrix-based secret sharing [4]，一種由 counting-based secret sharing [5]改進而來的 secret sharing 方法。它把由祕密產生的 sharing，使用三原色（RGB）的圖片，並使用兩種不同的隱寫術。第一種使用 LSB（Least Significant Bit），第二種使用了 DWT（Discrete Wavelet Transform）。

B. Secret sharing with multi-cover adaptive steganography [6]

這篇的作者提出了 multi-cover，跟傳統只藏在單一張圖片不同，作者把 sharing 隱藏到不同的圖片裡。藏的地方越多，單一張圖片儲存的資訊越少，因此也更不容易被發現。

C. Cyber warfare: steganography vs. steganalysis [7]

在此篇論文裡，作者針對圖片最低有效位元（Least Significant Bit，LSB）的隱寫術做了隱寫分析。他發現，通常在一張圖片上，一個點的 LSB 會與其鄰近點的 LSB 相似。因此只要抽取出整張圖片的 LSB，形成 LSB-plane，並針對此做比較，就可以很輕易發現這張圖片裡面是否藏有訊息，因為藏有訊息的 LSB-plane 中將會有某些部分是不連續的。

IV. 研究目標

本論文將使用 secret sharing 來提升祕密的穩健性和可靠性。除了 secret sharing 外，同時使用隱寫術來隱藏 sharing，讓 sharing 不容易被發現。

在 Cyber warfare [7]這篇文章裡，作者提到使用 LSB 把資訊藏在圖片裡面是件危險的事，因為此方法很容易被偵測。為此，本論文將使用聲音取代圖片來進行隱寫術，聲音除了沒有 LSB 連續的特徵以供偵測外，聲音也不像圖片一樣，能夠靜態地仔細觀察整張圖片；聲音只能在連續播放的過程中聆聽。

除了聲音之外，本論文還設計了一個 server 與多個 clients 的架構。server 把已經藏在音檔的 sharing 送到多個不同 client，如圖1所示，此方式讓想竊取機密的駭客，必需要能夠同時竊聽不同連線，才能有破解的一絲希望，因此更難去執行。

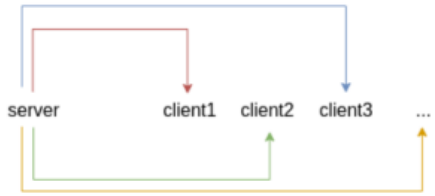


圖1. Server 與 client

V. 技術背景

A. Shamir's Secret Sharing

Shamir's Secret Sharing 的演算法共有兩個步驟。第一步驟是將原本的機密資料 D 拆成 n 份 sharing，並決定需要幾份 (k) 以上的 sharing 才能還原，故$1 \le k \le n$，並以 k 的值來決定一個多項式的最高冪為 k–1，而零次項的值則為 D。

第二步則從這 n 份 sharing 中湊得 k 份 sharing，並將它還原為原本的祕密，若湊得的份數少於 k 份，則無法還原。拆成 n 份 sharing 與需要 k 份 sharing 才能組回去原本祕密的情境，又稱為(k,n) threshold。

以下將以 threshold 為3，sharing 數 n=5，祕密 D=34為例。在 Shamir's Secret Sharing 的第一步驟中，將產生一個多項式，由於 k 為3，因此需要產生的多項式的最高次方為2，而零次項的值為34，其它次項的值則可隨機產生，以公式(1)為例，

$$F(x) = 4x^2 - 21x + 34 \qquad (1)$$

由於 n 為5，因此需要在這平面上隨機找5個點，F(2)=8、F(3)=7、F(4)=14、F(5)=29、F(6)=52。

而在 Shamir's Secret Sharing 第二步驟中，使用了拉格朗日差值法（Lagrange Interpolation）。如果收集到了3個 sharing，F(3)=7、F(5)=29、F(6)=52，並代入拉格朗日差值法(2)，即可得到以下式子，簡化後即可得出(1)，得出 D=34。

$$F(x) = 7\frac{(x-5)(x-6)}{(3-5)(3-6)} + 29\frac{(x-3)(x-6)}{(5-3)(5-6)} + 52\frac{(x-3)(x-5)}{(6-3)(6-5)} \qquad (2)$$

B. 聲音取樣

在音檔中，以下幾個是控制聲音格式的重要參數：channel 、sample、frame 、sample rate、sample format。

1. channel 代表著總共有幾個聲道，例如若是要輸出在電話的話筒，那麼 channel 數就設定為1，如果要輸出在有左右聲道的耳機，那麼 channel 數就設定為2。

2. sample 為儲存聲音的一個最基本單位，其大小由 sample format 控制，用來表示聲音當下的狀態。
3. frame 為所有聲道，在某個時間點的 sample 所成集合。如果是單聲道，那在某個時間點的一段 frame 裡就只有一個 sample。如果是雙聲道，那在某個時間點的一段 frame 裡就會有兩個 sample。
4. sample rate 為聲音的採樣頻率，單位為赫茲（Hertz，簡寫為 Hz）。sample rate 越高，代表聲音的品質越好，但也代表資料量越大。電話的頻率為八千赫茲，而 CD（Compact Disc）的音質大都是44100赫茲。
5. sample format 為呈現每個 sample 的數值，最常見的有8-bit、16-bit和32-bit，bit 數越多，能呈現聲音的範圍越大，聲音越準確。

C. Least Significant Bit

Least Significant Bit 簡稱 LSB。指的是在二進位的數字中，最小的位數。由於最小位數的更動，代表著更動後的值只會相差0或1，不容易被發現，因此經常被拿來當作隱寫術隱藏資料的位置。

D. Peak signal-to-noise ratio [8]

Peak signal-to-noise ratio 簡稱 PSNR，用來測量訊號的雜訊比，單位為分貝。其定義如(3)

$$PSNR = 10\log_{10}\left(\frac{MAX^2}{MSE}\right) \qquad (3)$$

以計算聲音的 PSNR 來說，MAX 值為 sample 的最大值。而 MSE（Mean Square Error）代表原始聲音與加入雜訊後的聲音之間的均方差，其定義如(4)

$$MSE = \sum_{i=1}^{n}(C_i - S_i)^2 \qquad (4)$$

其中 C 為原始聲音，S 代表加入雜訊後的聲音。我們用隱寫術藏進聲音中的這些 bit，在這公式中就被視為雜訊。

VI. 實驗架構

A. 決定參數

在此實驗架構中，將會有 server 和 client 兩種角色的架構。首先 server 需要先決定祕密 D，並與 client 約定好(k,n)的值，並由 Shamir Secret Sharing Scheme 的軟體 ssss [9]產生 n 份 sharing 後，等待 client 的連線。

B. server 錄音

當有 client 連上 server 時，server 則可以決定是否要傳送資料。當 server 決定要開始傳送資料時，server 會開始錄音。在本實驗中，將會以 sample rate 為8000、channel 為1，並用 sample format 為 unsinged int8來錄製5~10秒鐘的聲音，用來模擬一則語音訊息。

C. 隱寫術

錄完音後，會把 n 份 sharing 的其中一份藏進錄音檔裡。使用的方式是用 LSB 藏在 sample 裡，本實驗使用的 format 為 unsinged int8，所以每個 sample 就為一個 byte，也就是一個 byte 可以藏入一個 bit。如圖2所示，sharing 為**1-0bf53ca7dd**，長度為12 bytes。在隱藏資料前，我們需要先用一個 byte 來表示所藏資料的大小。假設共有12個字元，12用二進位表示為00001100。之後把00001100這8個 bits 藏進前8個 sample 中。接著把 sharing 用 ASCII 來表示，也就是一個英文字元用8個 bit 表示，所以一個字元會分別藏在8個 sample 裡，以上述的範例來說，則需要96個 sample 來藏。長度連同 sharing，全部共會用到8+96=104個 sample。

圖2. 隱藏範例

D. 包進 WAV 檔

把已經藏好的 sample，包成 WAV 檔，並使用 TCP 把此 WAV 檔送到 client。之後，server 又回到第二步，等待下一個 client 連線時送出下一份 sharing，直到所有 sharing 被領取完畢。

E. client 解析封包

client 收到 WAV 檔後，會先把所有的 sample 從 WAV 檔取出來，並從前八個 sample 取得 sharing 的長度。取得長度之後，就能知道總共要再取幾個 sample，以得出 sharing。

F. 取回祕密

在 n 個 client 中，他們手上都會有不同的 sharing。只要聚集 k 份以上的 sharing，就能透過 ssss [9]還原出祕密 D。

VII. 實驗結果

在實驗的步驟二裡，我們模擬了語音訊息傳送的情境，使用格式為 unsinged int8，channel 為1，並由 ssss

以(*k*,*n*)為(3,5)、產生 bytes 為12的 sharing 來藏入104個 sample 裡。以下 server 將分別測量錄音5~10秒，每次固定一個秒數，發送5個 sharing，並使用(3)與(4)比較它們之間的 MSE 和 PSNR。因為使用的 format 為 unsinged int8，所以 MAX 的值為255。結果如表 I、圖3和圖4所示

表 I
錄音秒數與 MSE 和 PSNR 對照

秒數	Sample 數	MSE	PSNR
5	40000	0.0012450	77.197667
6	48000	0.0011041	77.704725
7	56000	0.0009178	78.512879
8	64000	0.0008000	79.113535
9	72000	0.0007333	79.482942
10	80000	0.0006575	79.963869

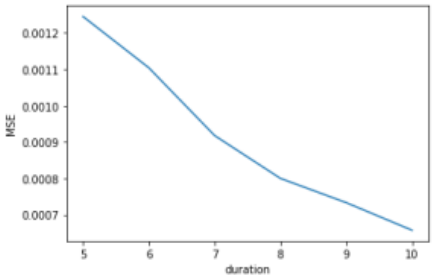


圖3. MSE 比較

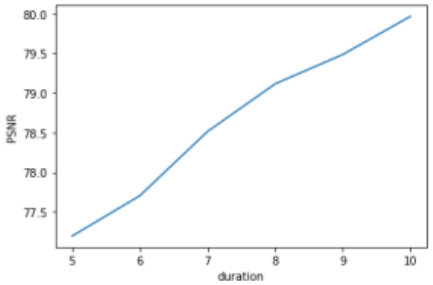


圖4. PSNR 比較

VIII. 結論

本實驗用來隱藏 sharing 的音檔皆是由 server 接收到 client 的連線後才開始錄音的，每次傳出去的音檔都是隨機的。因此，當有多個 sharing 要發送到不同地方時，每次發送的音檔都是不一樣，可用此來降低攻擊者的疑慮。

發送音檔的目的地皆不相同，因此要在多個接收端去攔截信號，且能夠偷到一定數量（*k*）以上的音檔，又為攻擊者加上一層阻礙。

IX. 未來展望

在聲音傳送的部分，用 TCP 來傳送仍不是最常見的形式，通常即時性的語音、視訊不會使用 TCP 來傳送，而是使用 UDP 搭配 RTP 來傳送。

但 UDP 和 RTP 並不像 TCP 一樣，能夠確保封包完整到達。如果在網路環境不好的情況下，封包傳送錯誤時，經常需要整段重傳，反而容易吸引攻擊者的注意。為了確保重要的 sharing 能夠傳達到目的地，未來考慮加入錯誤更正碼在 sharing 中，即使 server 在傳送過程中出現了錯誤，client 也能夠自動修正，從而還原出正確的 sharing。

參考文獻

[1] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[2] G. R. Blakley, "Safeguarding cryptographic keys," in *Managing Requirements Knowledge, International Workshop on*, 1979: IEEE Computer Society, pp. 313-318.

[3] F. Al-Shaarani and A. Gutub, "Securing matrix counting-based secret-sharing involving crypto steganography," *Journal of King Saud University-Computer and Information Sciences*, 2021.

[4] S. Porwal and S. Mittal, "A threshold secret sharing technique based on matrix manipulation," in *AIP Conference Proceedings*, 2020, vol. 2214, no. 1: AIP Publishing LLC, p. 020020.

[5] A. Gutub, N. Al-Juaid, and E. Khan, "Counting-based secret sharing technique for multimedia applications," *Multimedia Tools and Applications*, vol. 78, no. 5, pp. 5591-5619, 2019.

[6] H.-D. Yuan, "Secret sharing with multi-cover adaptive steganography," *Information Sciences*, vol. 254, pp. 197-212, 2014.

[7] H. Wang and S. Wang, "Cyber warfare: steganography vs. steganalysis," *Communications of the ACM*, vol. 47, no. 10, pp. 76-82, 2004.

[8] M. Tayel, A. Gamal, and H. Shawky, "A proposed implementation method of an audio steganography technique," in *2016 18th international conference on advanced communication technology (ICACT)*, 2016: IEEE, pp. 180-184.

[9] B. Poettering. "ssss" (https://linux.die.net/man/1/ssss)